A

Quantifying Privacy Leakage in Multi-Agent Planning

MICHAL ŠTOLBA, Faculty of Electrical Engineering, Czech Technical University
JAN TOŽIČKA, Faculty of Electrical Engineering, Czech Technical University
ANTONÍN KOMENDA, Faculty of Electrical Engineering, Czech Technical University

Multi-agent planning using MA-STRIPS-related models is often motivated by the preservation of private information. Such a motivation is not only natural for multi-agent systems, but is one of the main reasons, why multi-agent planning (MAP) problems cannot be solved centrally. Although the motivation is common in literature, the formal treatment of privacy is often missing. In this paper, we expand on a privacy measure based on information leakage introduced in our recent work, where the leaked information is measured in terms of transition systems represented by the public part of the problem with regard to the information obtained during the planning process. Moreover, we present a general approach to computing privacy leakage of search-based multi-agent planners utilizing search-tree reconstruction and classification of leaked superfluous information about the applicability of actions. Finally, we present an analysis of the privacy leakage of two well know algorithms MAFS and Secure-MAFS both in general and on a particular example.

CCS Concepts: •**Computing methodologies** → **Multi-agent planning; Planning for deterministic actions;**•**Security and privacy** → Information-theoretic techniques;

Additional Key Words and Phrases: Multi-agent planning, deterministic domain-independent planning, privacy, security

ACM Reference Format:
Michal Štolba, Jan Tožička, and Antonín Komenda, 2016. Quantifying Privacy Leakage in Multi-Agent Planning. *ACM Trans. Embedd. Comput. Syst.* V, N, Article A (January YYYY), 20 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Multi-agent planning models the problems in which multiple entities (or agents) need to generate a coordinated sequence of actions in order to fulfill some specified goal, either common or respective to each agent. If the environment and actions are deterministic (that is, their outcome is unambiguously defined by the state they are applied in), we are talking about deterministic multi-agent planning. Here, we focus on a cooperative scenario, where the agents are coordinating their actions in order to fulfill a common goal. The reason the agents cannot simply feed their problem descriptions into a centralized planner typically lies in that, although the agents cooperate, they want to share the information necessary for their cooperation only, but not the information about their inner processes and data. An example of a multi-agent planning scenario with strong privacy concerns is a coalition surveillance mission, where Unmanned Aerial Vehicles (UAVs) survey an area and need to be refueled by coalition partners (a coalition base). The surveyed areas and the current state of the supplies of the base are private (secret).

This research was supported by the Czech Science Foundation (grant no. 15-20433Y) and by the Grant Agency of the Czech Technical University in Prague (grant no. SGS16/235/OHK3/3T/13).
Author's addresses: All authors, Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo náměstí 13, 121 35 Prague 2, Czech Republic.

DOI: http://dx.doi.org/10.1145/0000000.0000000

In general, the information that is considered private may be part of the global state (e.g., the location or state of the supplies of a military base), some actions performed by the agents (e.g., unloading of the supplies at the base) or some of the preconditions and effects of the actions interacting with other agents (e.g., the precise state of the supplies when refueling a coalition UAV). If any other agent (typically referred to as an adversary in the context of privacy) either directly receives any of the private information, or can indirectly deduce it from the communicated public information, we say that the private information has leaked.

Our focus is on domain-independent deterministic cooperative multi-agent planning with privacy, known as Privacy-Preserving Multi-Agent Planning (PP-MAP). A number of planners solving PP-MAP problems have been proposed in recent years, such as MAFS [Nissim and Brafman 2014], FMAP [Torreño et al. 2014], PSM [Tožička et al. 2015] and GPPP [Maliah et al. 2016b]. Although all the mentioned planners claim to be privacy-preserving, thorough formal treatment of such claims is rather scarce. Privacy properties of MAFS are discussed by [Nissim and Brafman 2014] and expanded upon by [Brafman 2015], proposing Secure-MAFS, a version of MAFS with stronger privacy guarantees. This approach was recently generalized in the form of Macro-MAFS [Maliah et al. 2016a] together with an experimental evaluation which was missing in the previous work. Despite the claimed privacy preservation, none of the mentioned planners has undergone either a theoretical or a practical quantitative analysis of the private information actually leaked during the execution. In the context of DisCSP, the authors in [Faltings et al. 2008] provide a formal definition of privacy and a number of techniques aimed at its preservation, whereas in [Greenstadt et al. 2006] the authors experimentally evaluate the privacy leakage on a particular example. Such an analysis is crucial for the meaningful comparison of privacy-preserving planners and for the evaluation of novel approaches. In this work, we aim to provide a general formal framework necessary for such an analysis of PP-MAP planners and apply it on two particular algorithms, MAFS and Secure-MAFS.

Our approach is based on research in the area of secure Multi-Party Computation (MPC) [Yao 1982], where the amount of privacy loss in general algorithms (or functions) has been studied as information leakage [Smith 2009; Braun et al. 2009]. In this paper, we take this general approach and apply it on the problem of PP-MAP. The information leakage is quantified based on the difference in the probability of guessing the right input of a function before and after the distributed computation. The high-level formula defined in [Smith 2009] is

$$\text{information leaked} = \text{initial uncertainty} - \text{remaining uncertainty}, \tag{1}$$

where the initial uncertainty is related to the probability of guessing the right input without any additional knowledge gained from the execution of the algorithm, whereas the remaining uncertainty is the probability of guessing the right input given the output of the particular execution.

In order to analyze privacy leakage in PP-MAP, we propose the following methodology: First, we define a privacy leakage measure based on secure MPC. The measure is defined in a bottom-up fashion, that is, we first identify the particular sources of privacy leakage applicable to PP-MAP in general and compute the exact amounts of privacy leaked by the identified sources and their combinations. Next, we provide a high-level algorithm for computing the privacy leakage based on the identified sources of leakage applied to the distributed state-space search. Finally, we analyze two prominent search-based PP-MAP algorithms (MAFS and Secure-MAFS) on a particular example.

This work expands on [Štolba et al. 2016a; 2016b], where the concept of information leakage based on the uncertainty about the agent's transition system was first intro-

duced. We extend the work by providing a more precise and formal definition, a general method of computing the leakage bounds, and analysis of the MAFS and Secure-MAFS algorithms on an example.

2. MULTI-AGENT PLANNING

As an example running throughout the paper, we use the most simple case of the coalition surveillance mission problem with one UAV and two secret locations (see Figure 1). We omit the movement actions for simplicity (movement between the surveyed locations would be private, movement to the coalition base would be public).

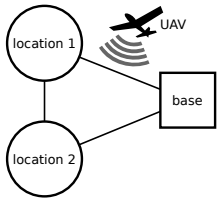


Fig. 1. UAV surveillance scenario example.

The most common model for PP-MAP is MA-STRIPS [Brafman and Domshlak 2008] and derived models (such as MA-MPT [Nissim and Brafman 2014] using multi-valued variables). We reformulate the MA-STRIPS definition in the following way (we also generalize the definition to multi-valued variables). Formally, for a set of agents $\mathcal{A}$, the PP-MAP problem $\mathcal{M} = \{\Pi_i\}_{i=1}^{|\mathcal{A}|}$ is a set of agent problems. An agent problem of agent $\alpha_i \in \mathcal{A}$ is defined as

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

where $\mathcal{V}_i$ is a set of variables s.t. each $V \in \mathcal{V}_i$ has a finite domain $\mathsf{dom}(V)$. If all variables are binary (i.e., $|\mathsf{dom}(V)| = 2$ for all V), the formalism corresponds to MA-STRIPS. The set of variables is partitioned into the set $\mathcal{V}^{\mathsf{pub}}$ of public variables (with all values public), common to all agents, and the set $\mathcal{V}_i^{\mathsf{priv}}$ of variables private to α_i (with all values private), such that $\mathcal{V}^{\mathsf{pub}} \cap \mathcal{V}_i^{\mathsf{priv}} = \emptyset$. The complete assignment over $\mathcal{V} = \bigcup_{i=1}^{|\mathcal{A}|} \mathcal{V}_i$ is the *state*, the partial assignment over $\mathcal{V}$ is the *partial state*. We denote $s[V]$ as the value of V in the (partial) state s and $\mathsf{vars}(s)$ as the set of variables which have a value assigned in the (partial) state s ($\mathsf{vars}(s) = \mathcal{V}$ for the non-partial state). The state s_I is the initial state and $s_\star$ is the partial state representing the goal condition, that is, s is the goal state iff for all variables $V \in \mathsf{vars}(s_\star)$, $s_\star[V] = s[V]$.

The set $\mathcal{O}_i$ of the actions is comprised of a set $\mathcal{O}_i^{\mathsf{priv}}$ of private actions of α_i, a set $\mathcal{O}_i^{\mathsf{pub}}$ of public actions of α_i and a set $\mathcal{O}^{\triangleright}$ of projected actions of other agents (the principle of the projection is described later on in this section). The sets $\mathcal{O}_i^{\mathsf{pub}}$, $\mathcal{O}_i^{\mathsf{priv}}$ and $\mathcal{O}^{\triangleright}$ are pairwise disjoint. An action is defined as a tuple $a = \langle \mathsf{pre}(a), \mathsf{eff}(a), \mathsf{lbl}(a) \rangle$, where $\mathsf{pre}(a)$ and $\mathsf{eff}(a)$ are partial states representing the precondition and effect, respectively, and $\mathsf{lbl}(a)$ is a unique label. An action a is public if it is affected by or affects a public variable, formally 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 a private action. Notice that a public action may also have private preconditions or effects. An action a is applicable in state s if $s[V] = \mathsf{pre}(a)[V]$ for all $V \in \mathsf{vars}(\mathsf{pre}(a))$ 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. When we are considering the planning problem from the perspective of a single agent α_i (the agent trying to hide the private information), we omit the index i whenever possible.

EXAMPLE. The problem in the running example consists of two agents $\mathcal{A} = \{\alpha_{\mathsf{UAV}}, \alpha_{\mathsf{base}}\}$ and can be described using following sets of variables:

Variables	Description	Variable	Values	s_I	$s_\star$
$\mathcal{V}^{\mathsf{pub}}$:	UAV has fuel	$\mathbf{f}$	T/F	F	-
	mission is complete	$\mathbf{c}$	T/F	F	T
$\mathcal{V}^{\mathsf{priv}}_{\mathsf{UAV}}$:	location 1 is complete	l1	T/F	F	-
	location 2 is complete	l2	T/F	F	-
$\mathcal{V}^{\mathsf{priv}}_{\mathsf{base}}$:	base has enough supplies	s	T/F	T	-

For further analysis, we use binary variables with T/F values, but all principles can be easily applied on general domain sizes. The problem consists of the following actions:

Actions (α_{UAV})	$\mathsf{lbl}(a)$	$\mathsf{pre}(a)$	$\mathsf{eff}(a)$
survey location 1	SL1	$\{\mathbf{f} = \mathsf{T}\}$	$\{\mathsf{l1} = \mathsf{T}, \mathbf{f} = \mathsf{F}\}$
survey location 2	SL2	$\{\mathbf{f} = \mathsf{T}\}$	$\{\mathsf{l2} = \mathsf{T}, \mathbf{f} = \mathsf{F}\}$
complete mission	C	$\{\mathsf{l1} = \mathsf{T}, \mathsf{l2} = \mathsf{T}\}$	$\{\mathbf{c} = \mathsf{T}\}$

Actions (α_{base})	$\mathsf{lbl}(a)$	$\mathsf{pre}(a)$	$\mathsf{eff}(a)$
refuel	R	$\{\mathbf{f} = \mathsf{F}, \mathsf{s} = \mathsf{T}\}$	$\{\mathbf{f} = \mathsf{T}, \mathsf{s} = \mathsf{F}\}$
refuel and resupply	RR	$\{\mathbf{f} = \mathsf{F}, \mathsf{s} = \mathsf{F}\}$	$\{\mathbf{f} = \mathsf{T}, \mathsf{s} = \mathsf{T}\}$

All actions in the problem are public (possibly with private preconditions or effects) for the ease of presentation, but private actions can be analyzed using the presented model as well. ⊠

We refer to the agent trying to hide the information as agent α (the agent). We model all other agents as a single agent β (the adversary), which is common in secure MPC, as all the agents can collude and combine their information in order to infer more information. The public part of the agent's problem Π which can be shared with the adversary is denoted as a public projection. The public projection of a (partial) state s is $s^{\triangleright}$, restricted only to variables in $\mathcal{V}^{\mathsf{pub}}$, that is $\mathsf{vars}(s^{\triangleright}) = \mathsf{vars}(s) \cap \mathcal{V}^{\mathsf{pub}}$. We say that s, s' are publicly equivalent states if $s^{\triangleright} = s'^{\triangleright}$ and that s, s' are publicly equivalent but distinct states if $s^{\triangleright} = s'^{\triangleright}$ and $s \neq s'$. The public projection of an action $a \in \mathcal{O}^{\mathsf{pub}}$ is $a^{\triangleright} = \langle \mathsf{pre}(a)^{\triangleright}, \mathsf{eff}(a)^{\triangleright} \rangle$ and of action $a' \in \mathcal{O}^{\mathsf{priv}}$ is an empty (no-op) action $\epsilon = \langle \emptyset, \emptyset \rangle$. The public projection of Π is

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

EXAMPLE. We analyze the example problem from perspective in which the agent α_{UAV} is trying to hide its private information from the agent α_{base}. Thus, Π is the problem of agent α_{UAV} and the part of Π which can be shared with the adversary α_{base} is the public projection $\Pi^{\triangleright}$:

Description	Var.	Val.	Actions in $\Pi^{\triangleright}$	$\mathsf{pre}(a)$	$\mathsf{eff}(a)$
			survey location	$\{\mathbf{f} = \mathsf{T}\}$	$\{\mathbf{f} = \mathsf{F}\}$
UAV has fuel	$\mathbf{f}$	T/F	complete mission	$\{\}$	$\{\mathbf{c} = \mathsf{T}\}$
mission complete	$\mathbf{c}$	T/F	refuel	$\{\mathbf{f} = \mathsf{F}\}$	$\{\mathbf{f} = \mathsf{T}\}$

⊠

In general, a single public projection $a^{\triangleright}$ can represent multiple actions a, a' such that $\mathsf{pre}(a)^{\triangleright} = \mathsf{pre}(a')^{\triangleright}$ and $\mathsf{eff}(a)^{\triangleright} = \mathsf{eff}(a')^{\triangleright}$. For the sake of simplicity with regards to the presentation of some concepts and the proofs in the later sections, we define a label-preserving projection $\Pi^{\trianglerighteq}$, where each projected action $a^{\trianglerighteq}$ corresponds to exactly one action a and the label of the action is preserved. Formally $a^{\trianglerighteq} = \langle \mathsf{pre}(a)^{\triangleright}, \mathsf{eff}(a)^{\triangleright}, \mathsf{lbl}(a) \rangle$.

Apart from this difference, $\Pi^{\trianglerighteq}$ is defined equally to $\Pi^{\triangleright}$. Later, we also show that the label preserving projection leaks a significant amount of information[1]. For disambiguation, we sometimes refer to $\Pi^{\triangleright}$ as a label non-preserving projection.

We define the solution to Π and subsequently to $\mathcal{M}$ as follows. A sequence $\pi = (a_1, ..., a_k)$ of actions from $\mathcal{O}$, s.t. a_1 is applicable in $s_I = s_0$ and for each $1 \leq i \leq k$, a_i is applicable in s_{i-1} and $s_i = a_i \circ s_{i-1}$, is a local s_k-plan, where s_k is the resulting state. If s_k is a goal state, π is a local plan that is a local solution to Π. Notice that such a π does not have to be the global solution to $\mathcal{M}$, as the actions of other agents ($\mathcal{O}^{\triangleright}$) are used as public projections only and may be missing preconditions and effects of other agents (in our context, the adversary). We define $\pi^{\triangleright} = (a_1^{\triangleright}, ..., a_k^{\triangleright})$ with all ϵ actions omitted to be the public projection of π. 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 agree on the public actions, formally $\pi_i^{\triangleright} = \pi_j^{\triangleright}$ for all i, j.

EXAMPLE. For further analysis, let us assume that the final global solution to the example problem together with its public projection is $\pi_{\mathsf{UAV}} = \{\mathsf{R}, \mathsf{SL1}, \mathsf{R}, \mathsf{SL2}, \mathsf{C}\}$, $\pi_{\mathsf{base}} = \{\mathsf{R}, \mathsf{SL}, \mathsf{RR}, \mathsf{SL}, \mathsf{C}\}$ and $\pi_{\mathsf{UAV}}^{\triangleright} = \pi_{\mathsf{base}}^{\triangleright} = \{\mathsf{R}, \mathsf{SL}, \mathsf{R}, \mathsf{SL}, \mathsf{C}\}$. ⊠

Finally, we explicitly define the transition system induced by the planning problem. The transition system of a problem Π is a tuple $\mathcal{T}(\Pi) = \langle S, L, T, s_I, s_\star \rangle$, where $S = \prod_{V \in \mathcal{V}} \mathsf{dom}(V)$ is a set of states, $L = \{\mathsf{lbl}(a) | a \in \mathcal{O}\}$ is a set of transition labels corresponding to the actions in $\mathcal{O}$ and $T \subseteq S \times L \times S$ is a transition relation of Π s.t. $\langle s, l, s' \rangle \in T$ if $a \in \mathcal{O}$ s.t. $\mathsf{lbl}(a) = l$ is applicable in s and $s' = a \circ s$. The state $s_I \in S$ is the initial state and $s_\star \subseteq S$ is the set of all goal states.

A public projection of $\mathcal{T}(\Pi)$ is $\mathcal{T}(\Pi)^{\triangleright} = \langle S^{\triangleright}, L^{\triangleright}, T^{\triangleright}, s_I^{\triangleright}, s_\star^{\triangleright} \rangle$ such that $S^{\triangleright}$ is S restricted to $\mathcal{V}^{\mathsf{pub}}$, $L^{\triangleright} = \{\mathsf{lbl}(a^{\triangleright}) | a \in \mathcal{O}^{\triangleright}\}$ and $\langle s^{\triangleright}, \mathsf{lbl}(a^{\triangleright}), s'^{\triangleright} \rangle \in T^{\triangleright}$ if $\langle s, \mathsf{lbl}(a), s' \rangle \in T$, where $\mathsf{lbl}(a^{\triangleright})$ is any label such that $\mathsf{lbl}(a^{\triangleright}) = \mathsf{lbl}(a'^{\triangleright}) \iff a^{\triangleright} = a'^{\triangleright}$. Notice that the public projection is, in fact, an abstraction of $\mathcal{T}(\Pi)$ [Haslum and Geffner 2000].

The complete transition system of the example global problem $\mathcal{M}$ is shown in Figure 2(a), together with the transition system of the public projection $\Pi^{\triangleright}$ in Figure 2(b).

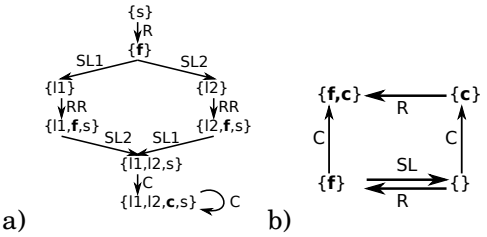


Fig. 2. a) Global transition system (its reachable part) of $\mathcal{M}$. The variables shown are set to true, all other variables are set to false. b) The transition system of the public projection.

This concludes the formal definition and the description of a running example which we use throughout the paper.

3. PRIVACY

Apart from the specialized privacy leakage quantification by [Van Der Krogt 2009], which is not practical, as it is based on enumeration of all plans, and is not applicable to MA-STRIPS in general, the only rigorous definition of privacy for PP-MAP so far

[1]The label-preserving projection is commonly assumed in PP-MAP planners and heuristics and was not identified as a source of significant loss of privacy in literature.

was proposed by [Nissim and Brafman 2014; Brafman 2015]. Here, we rephrase the definitions in order to build upon them in the next sections.

An algorithm is *strong privacy-preserving* if the adversary can deduce no information about a private variable and its values and private preconditions/effects of an action, beyond what can be deduced from the public input ($\Pi^{\triangleright}$) and the public output (projection of the solution plan $\pi^{\triangleright}$).

We say that an algorithm is *weak privacy-preserving* if, during the whole run of the algorithm, the agent does not openly communicate private parts of the states, private actions and private parts of the public actions. In other words, the agent openly communicates the information in $\Pi^{\triangleright}$ only. Even if not openly communicated, the adversary may deduce the existence and values of private variables, preconditions and effects from the public information communicated during the execution of the algorithm, beyond what can be deduced from the public parts of the input and output only. For example, if a state is not expanded by the agent using some particular action, the adversary may deduce that the state contains a private variable which prevents the applicability of that action.

In the following section we define the bottom-up privacy leakage measure based on secure MPC techniques.

3.1. Privacy Leakage

One of the threat models studied in the literature on secure MPC is one where the threat of an attack will allow the adversary to guess the private information of the agent. In the case of PP-MAP, this means that the adversary may be able to guess the actual transition system of the agent.

Based on [Smith 2009], let us have an algorithm which takes a private input H and produces a public output L. We are interested in how much information about H can be deduced by an adversary who sees the output L. We assume that there is an prior, publicly-known probability distribution of a random variable H with a finite space of possible values $\mathcal{H}$. We denote the prior probability that H has a value $h \in \mathcal{H}$ by $P[H = h]$, and we assume that each element h of $\mathcal{H}$ has nonzero probability. Similarly, we assume that L is a random variable with a finite space of possible values $\mathcal{L}$, and with probabilities $P[L = l]$. The output distribution can be obtained from input distribution and the algorithm which is known. We assume that each output $l \in \mathcal{L}$ is possible, in that it can be produced by some input $h \in \mathcal{H}$.

In the case of PP-MAP, the private input H the adversary is attempting to guess is the transition system $\mathcal{T}(\Pi)$ of the agent's problem. In agreement with the assumptions above (the known distribution and finite space of values), we assume that an upper bound on the size (number of states) of $\mathcal{T}(\Pi)$ is publicly known as n. This bound, together with the public projection $\mathcal{T}(\Pi)^{\triangleright}$ of the transition system and public projection $\pi^{\triangleright}$ of the final plan, limits the number of the transition systems possible with respect to $\mathcal{T}(\Pi)^{\triangleright}$ and $\pi^{\triangleright}$. We denote the number as t_{prior}. We also assume that all the possible transition systems are equally probable, which gives us a uniform distribution with the probability $P[H = \mathcal{T}(\Pi)] = 1/t_{\mathsf{prior}}$.

The public output consists of all information obtained from the agent during the planning process, that is, the public projection of the transition system $\mathcal{T}(\Pi)^{\triangleright}$, the resulting plan π and all public information communicated during the planning process, such as reachable states, possible transitions, etc. We will represent the information derived from the output as a mapping $\tau_{\mathsf{post}} : \mathcal{O} \to \mathbb{N}$ from the actions of the agent to the number of possible transition sub-systems it represents according to the obtained information.

By converting the probability to an information measure, we obtain a measure known as min-entropy $H_\infty(X) = \log \frac{1}{P[H=\mathcal{T}(\Pi)]}$ where $\log$ represents a base-2 logarithm (and will so further on in the text). In our (uniform) case, $H_\infty(H) = \log t_{\mathsf{prior}}$. We use $H_\infty(H)$ as the measure of the initial uncertainty. In our (uniform) case, the probability of guessing the transition system $\mathcal{T}(\Pi)$ given the output reflecting all the information revealed by the algorithm can be measured by the conditional min-entropy $H_\infty(H|L) = \log t_{\mathsf{post}}$.

In summary, in the case of a deterministic algorithm and uniform distributions, we obtain the information leakage measures based on the min-entropy $H_\infty(H)$ and conditional min-entropy $H_\infty(H|L)$ as:

initial uncertainty. $H_\infty(H) = \log t_{\mathsf{prior}}$
remaining uncertainty. $H_\infty(H|L) = \log t_{\mathsf{post}}$
information leaked. $H_\infty(H) - H_\infty(H|L) =$

$$\log t_{\mathsf{prior}} - \log t_{\mathsf{post}} = \log \frac{t_{\mathsf{prior}}}{t_{\mathsf{post}}} \tag{2}$$

According to [Smith 2009], the remaining uncertainty gives a security guarantee as the expected probability of guessing H, that is the transition system of the agent, given the public output of the planning algorithm L. The expected probability is

$$2^{-H_\infty(H|L)} = 2^{-\log t_{\mathsf{post}}} = 1/t_{\mathsf{post}} \tag{3}$$

where, again, t_{post} is the number of possible transition systems given the public output. Thanks to the determinism and uniform distributions, the result conveys, with intuition, that the privacy preservation decreases by lowering the number of possible transition systems. In the next section, we will focus on how to estimate t_{prior} and t_{post}.

3.2. Leakage Quantification in PP-MAP

In the previous section, we have placed an assumption that an upper bound n on the total number of the states of the agent's transition system is publicly known. This results in n^2 possible transitions (including loops) and 2^{n^2} possible transition systems. Moreover, we assume, that for all $V \in \mathcal{V}^{\mathsf{priv}}$ there are bounds p and d on the number of private variables $|\mathcal{V}^{\mathsf{priv}}| \leq p$ and on the size of the private variable domains $|\mathsf{dom}(V)| \leq d$, respectively, and that the bounds p, d are publicly known also.

Similarly to [Brafman 2015], for the simplicity of presentation, we assume that $\mathcal{O}^{\mathsf{priv}} = \emptyset$. This assumption can be stated without loss of generality as each sequence of private actions followed by a public action can be compiled as a single public action (with a potential exponential blow up in the number of public actions). From the perspective of privacy, it is clear that when adhering to the weak privacy, private action is never communicated and, thus, can never leak. Any information about private action always leaks in the sense of the described compilation, that is, it appears as if each new public action created by the compilation had some additional private preconditions or effects.

Let us first consider the public projection of the agent's transition system $\mathcal{T}(\Pi)^\triangleright$. Based on the above bounds, the number of private states $s \in S$ represented by a single public state $s^\triangleright \in S^\triangleright$ is d^p. The number of possible private transitions $\langle s, l, s' \rangle \in T$ represented by a single public transition $\langle s^\triangleright, l^\triangleright, s'^\triangleright \rangle \in T^\triangleright$is $(d^p)^2$, that is, from each private state s there is a transition to a private state s'.

For a single variable ($p = 1$), there are d private states represented by each public state and for a single action a, the respective public transition $\langle s^\triangleright, \mathsf{lbl}(a^\triangleright), s'^\triangleright \rangle \in T^\triangleright$ represents d^2 possible private transitions between two public states. As demonstrated in Figure 3, for each of the left d private states there either is or is not a transition

to each of the right d private states, based on the private preconditions and effects of a. This set of transitions represents a transition system of action a where the left to right direction represents the application of a. The upper part represents $V = \mathsf{T}$ before (left) and after (right) the application of the action and the lower part represents $V = \mathsf{F}$ analogously. For example, an action represented as $\rightarrow$ is applicable for both $V = \mathsf{T}, V = \mathsf{F}$ with no effect, as both arrows start and end in the same upper and lower parts, i.e. values of V.

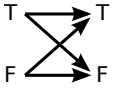


Fig. 3. Possible transitions for a single binary variable with T/F values. Possible transition systems represented by a single public action are all subsets of the depicted transitions. The left side represents a precondition, the right side an effect.

An upper bound on the number of all such transition systems for a is a constant $t_a = 2^{d^2}$, that is, the number of subsets of the d^2 transitions (subsets of arrows in Figure 3). A public transition encodes the existing transitions only (actions applicable in, at least, one private state). This means that an empty transition system is not an option, thus, $t_a = 2^{d^2} - 1$. For a binary variable, d equals to 2, therefore $t_a = 2^{2^2} - 1 = 15$. As the variables are independent, for p variables, we obtain $t_a = (2^{d^2} - 1)^p$, or $t_a = 15^p$ if $d = 2$.

In the case of a label preserving projection where $\mathsf{lbl}(a) = \mathsf{lbl}(a^{\triangleright})$ all actions $a \in \mathcal{O}^{\triangleright}$ have unique labels, because each public transition $\langle s^{\triangleright}, l^{\triangleright}, s'^{\triangleright} \rangle \in T^{\triangleright}$ represents exactly one action $a \in \mathcal{O}^{\triangleright}$. As a single (deterministic STRIPS) action can never produce multiple states when applied in a single state, the number of possible transition systems t_a is significantly reduced. For a single variable with d values, we get the number of partial functions between two sets of size d, which is $t_a = (1 + d)^d - 1$ (again -1 for an empty transition system), for STRIPS, where $d = 2$ we obtain $t_a = (1 + 2)^2 - 1 = 8$. If we assume no conditional effects, a single action cannot have a different effect depending on the state and, thus, the transition system where the application of a in T results in F and the application in F results in T is not possible and we are left with $t_a = 7$. Again, as the variables are independent, the numbers can be multiplied for each variable. These 7 transition systems for a single variable V s.t. $\mathsf{dom}(V) = \{\mathsf{T}, \mathsf{F}\}$ are schematically shown to be the following

$$\rightarrow, \nearrow, \searrow, \rightarrow, \rightarrow, \nearrow, \searrow, \tag{4}$$

each representing one possible transition system of an action a. As in Figure 3, the upper arrow represents the application of a when $V = \mathsf{T}$, the lower arrow when $V = \mathsf{F}$. A horizontal arrow represents unchanged value, a diagonal arrow represents a change from $V = \mathsf{T}$ to $V = \mathsf{F}$ ($\searrow$) or from $V = \mathsf{F}$ to $V = \mathsf{T}$ ($\nearrow$). A missing arrow means that a is not applicable when $V = \mathsf{T}$ (upper arrow missing) or $V = \mathsf{F}$ (lower arrow missing).

In the case of the label non-preserving projection, the number of transition systems for one variable is 15. This number is obtained by adding the non-deterministic transitions to Equation 4, which means, for example, adding $\searrow$ to $\rightarrow$ resulting in $\searrow$, meaning that the multiple actions with the same public projection can be applied in $V = \mathsf{T}$ and the result can be both $V = \mathsf{T}$ and $V = \mathsf{F}$, depending on the actual action applied.

Based on the above explanation, a complete public projection $\mathcal{T}(\Pi)^{\triangleright}$ of the agent's transition system restricts the number of possible private transition system to $t_\Pi = {t_a}^{|T^{\triangleright}|}$, where t_a is the number of transition systems represented by a single transition (again, we can multiply it, as the actions are independent).

EXAMPLE. The example problem has binary variables, two of which are private, that is $p = 2$ and the number of transitions representing public actions is 2. For $\Pi^{\triangleright}$ we get

$t_\Pi = 15^{2p} = 15^4 \cong 5 \times 10^4$ possible transition systems. For $\Pi^{\trianglerighteq}$ we get $t_\Pi = 7^{2p} = 7^4 \cong 2 \times 10^3$. Notice the order-of-magnitude difference between the two variants. ⊠

3.3. Sources of Leakage

Before we analyze complex algorithms, let us first focus on elements which have a major impact on the privacy leakage. From Equation 1 and its particular instance, Equation 2, it follows that the source we need to focus on is the public information which is communicated in addition to the initial information and the solution (i.e., the projected problem $\Pi^{\triangleright}$ and the projected plan $\pi^{\triangleright}$). If some private information can be deduced from such communicated public information, we refer to it as *superfluous* information. In particular, the sources of information we focus on are superfluous distinct states and superfluous action applicability information.

There are other possible sources of information leakage and their combinations (e.g., partial plans), but we base our analysis on these two prominent sources, thus, providing a lower bound on the information leaked. We use label-preserving projection for the ease of presentation, but in the further analysis we focused mainly on label non-preserving projection, as it is more reasonable. In the following propositions, we assume a STRIPS model with p private binary variables. We start with the definition of the superfluous distinct states.

Definition 3.1. Two states s, s' such that $s \neq s'$ are said to be *superfluous distinct states* iff s, s' are publicly equivalent, that is, $s^{\triangleright} = s'^{\triangleright}$ and $s \neq s'$ is revealed to the adversary agent β.

The possibilities how superfluous distinct states can leak are algorithm-dependent and will be discussed in-detail later. The superfluous distinct state information itself does not reduce the number of possible transition systems, but can be used for further deduction of the superfluous action applicability.

The first case of superfluous action applicability we focused on is the situation, where an action is applicable in the initial state s_I, or a state created from s_I by a sequence of actions of the adversary β.

Definition 3.2. Let $\pi_{\mathsf{sup}} = (a_1, ..., a_k)$ be an s_k-plan such that π_{sup} is not a prefix of the final plan π and π_{sup} is revealed to the adversary agent β. Let a_l be the first action of agent α in π_{sup}. Then a_l is an *init-applicable* action.

PROPOSITION 3.3. *Let $\pi_{\mathsf{sup}} = (a_1, .., a_l, ..., a_k)$ be an s_k-plan and let a_l be an init-applicable action. The number of transition systems represented by a label-preserving projection $a_l^{\trianglerighteq}$ is $t_a^{\mathsf{ia}} = 5^p$ and by a label non-preserving projection $a_l^{\triangleright}$ is $t_a^{\mathsf{ia}} = 12^p$.*

PROOF. Let $\pi_{\mathsf{sup}} = (a_1, .., a_l, ..., a_k)$ be an s_k-plan and let a_l be an init-applicable action. From the definition of an s_k-plan, a_1 is applicable in the initial state s_I. Without the loss of generality, we can assume that, in s_I, all variables have a particular value (e.g., T for STRIPS), as the values can be arbitrarily renamed. The values of s_I private to the agent α are not changed by the actions of the adversary β. Let a_l be the first action of α in π_{sup}, applicable in state s_{l-1}. From the above, s_{l-1} has the same particular values of variables private to α. Again, without loss of generality, we can assume that the value is T. This fixes the uncertainty about a_l in that it is, now, not possible that a_l is not applicable in s_{l-1}. But since we know that $s_{l-1}[V] = \mathsf{T}$, for a single binary variable V this means that a_l cannot have a precondition in the form of $V = \mathsf{F}$. In Equation 4, this corresponds to $\rightarrow$ and $\nearrow$ and, thus, only the remaining 5 transition systems are possible. The same holds for each variable independently. The resulting number of transition systems is, thus, $t_{a_l} = 5^p$ for p variables. For the label non-preserving projection, the combination of $\rightarrow$ and $\nearrow$ is not possible also, which

results in removing 3 transition systems out of the 15 total for each variable, thus, $t_{a_l} = 12^p$ for p variables. □

In the rest of this section, we focus on the information about the applicability of an action a on two superfluous distinct states s, s'. We define two types of the superfluous action applicability information.

Definition 3.4. Action $a \in \mathcal{O}$ is a *privately-dependent* action iff $\mathsf{vars}(\mathsf{pre}(a)) \cap \mathcal{V}^{\mathsf{priv}} \neq \emptyset$, that is, a has some private preconditions.

PROPOSITION 3.5. *Let $a \in \mathcal{O}$ be a privately-dependent action. The number of transition systems represented by a label-preserving projection $a^{\trianglerighteq}$ is $t_a^{\mathsf{pd}} = 7^p - 3^p$ and by a label non-preserving projection $a^{\triangleright}$ is $t_a^{\mathsf{pd}} = 15^p - 9^p$.*

PROOF. Let us first consider the label-preserving projection $\Pi^{\trianglerighteq}$. For $p = 1$, the following transition systems in Equation 4 $\rightarrow, \nearrow, \searrow$ are not possible to be represented by $a^{\trianglerighteq}$, because such transition systems do not depend on the variable, leaving only four, in particular $\not\rightarrow, \searrow, \rightarrow, \not\nearrow$. For p variables, such a situation must occur for at least one variable. We can formulate the total number of possible transitions, by taking $t_a = 7^p$ and subtracting the number of transition systems which does not satisfy the condition, that is, for all p variables, the action is applicable in both values, as in $\rightarrow, \nearrow, \searrow$. By subtracting that 3^p we obtain $t_a^{\mathsf{pd}} = 7^p - 3^p$. The result for a label non-preserving projection is achieved by the same reasoning applied on all of the 15 possible transition systems. □

EXAMPLE. In the running example, a privately-dependent action is C, because it depends on both the private variables l1 and l2. This means, that the number of possible transition systems represented by C is reduced from $7^2 = 49$ to $7^2 - 3^2 = 40$ in the case of label-preserving projection and from $15^2 = 225$ to $15^2 - 9^2 = 144$ in the case of label non-preserving projection. ⊠

A somewhat analogous situation arises, when a single action a is applicable in two publicly equivalent, but superfluous distinct states s, s'. In this case, the information learned is that the action does not depend on the private variable which distinguishes s and s', thus, reducing the number of possible transition systems for a.

Definition 3.6. Action $a \in \mathcal{O}$ is a *privately-independent* action iff $\mathsf{vars}(\mathsf{pre}(a)) \cap \mathcal{V}^{\mathsf{priv}} \neq \mathcal{V}^{\mathsf{priv}}$, that is, there is a private variable $V \in \mathcal{V}^{\mathsf{priv}}$ on which a does not have a precondition.

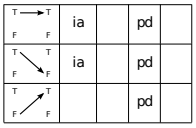
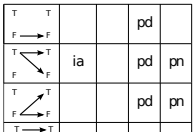
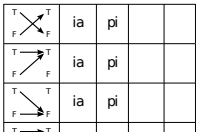
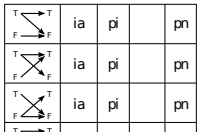
PROPOSITION 3.7. *Let $a \in \mathcal{O}$ be a privately-independent action. The number of transition systems represented by a label-preserving projection $a^{\trianglerighteq}$ is $t_a^{\mathsf{pi}} = 7^p - 4^p$ and by a label non-preserving projection $a^{\triangleright}$ is $t_a^{\mathsf{pi}} = 15^p - 6^p$.*

PROOF. As in Proposition 3.5, for the label-preserving projection $\Pi^{\trianglerighteq}$, let $p = 1$. The transition systems $\rightarrow, \not\rightarrow, \not\nearrow, \searrow$, where a is applicable in one of the states only, are not possible. Thus, for $p \geq 1$, the resulting number is $t_a^{\mathsf{pi}} = 7^p - 4^p$ for label preserving projection and $t_a^{\mathsf{pi}} = 15^p - 6^p$ for label non-preserving projection. □

EXAMPLE. In the running example, such an action is SL2, because it does not depend on any of the variables l1 and l2. This means that the number of possible transition systems represented by SL2 is reduced from $7^2 = 49$ to $7^2 - 4^2 = 33$ in the case of label-preserving projection and from $15^2 = 225$ to $15^2 - 6^2 = 189$. ⊠

In the case of label non-preserving projection, there is one more possible leakage. The application of a projected action $a^{\triangleright}$ (possibly representing multiple STRIPS actions) may result in two publicly equivalent, but distinct states.

Table I. All 15 transition systems possible, represented by $a^{\triangleright}$. The tags describe which of them are possible for a given type of action.

	ia		pd					pd			ia	pi				ia	pi		pn
	ia		pd			ia		pd	pn		ia	pi				ia	pi		pn
	ia		pd					pd	pn		ia	pi				ia	pi		pn
			pd			ia	pi				ia	pi		pn		ia	pi		pn

Types: ia – init-applicable action, pd – privately-dependent action, pi – privately-independent action, pn – privately-nondeterministic action.

Definition 3.8. Let $a \neq a' \in \mathcal{O}$ be two actions such that $a^{\triangleright} = a'^{\triangleright}$. Then $a^{\triangleright}$ is a *privately-nondeterministic* action iff there is a private variable $V \in \mathcal{V}^{\mathsf{priv}}$ such that $\mathsf{eff}(a)[V] \neq \mathsf{eff}(a')[V]$, that is a and a' differ in a private effect.

PROPOSITION 3.9. *Let $a^{\triangleright} \in \mathcal{O}^{\triangleright}$ be a privately-nondeterministic action. The number of transition systems represented by $a^{\triangleright}$ is $t_a^{\mathsf{pn}} = 15^p - 8^p$.*

PROOF. From Definition 3.8, $a^{\triangleright}$ represents at least two actions a, a' which differ in the private effects, that is, $\mathsf{eff}(a) \neq \mathsf{eff}(a')$. Now for at least one variable, the transition system represented by $a^{\triangleright}$ must contain both $\rightarrow, \nearrow$ or $\overrightarrow{\ }, \searrow$ transitions. This is not satisfied by 8 out of the 15 possible transition systems (see Table I, rows without pn), thus $t_a^{\mathsf{pn}} = 15^p - 8^p$. □

EXAMPLE. In the running example, such an action is SL representing both SL1 and SL2. This means that the number of possible transition systems represented by SL is reduced from $15^2 = 225$ to $15^2 - 8^2 = 161$. ⊠

Even more information can leak from the combination of the properties. We know, that an action cannot be both privately-dependent and privately-independent for a single variable, but other combinations are possible. Interesting combinations may especially arise from the init-applicable actions, where the "base" number of transition systems is lowered to 12^p (Proporsition 3.3) and other applicability properties, which reduce the possible transition systems for a single variable.

PROPOSITION 3.10. *Let $a^{\triangleright} \in \mathcal{O}^{\triangleright}$ be a first action in a superfluous plan π_{sup} and let $a^{\triangleright}$ be privately-independent. The number of transition systems represented by a label non-preserving projection $a^{\triangleright}$ is $t_a^{\mathsf{ia}\times\mathsf{pi}} = 12^p - 3^p$.*

PROOF. By counting the possible transition systems in Table I of a privately-independent action restricted to those applicable on $s_I[V] = \mathsf{T}$ for all $V \in \mathcal{V}^{\mathsf{priv}}$. □

PROPOSITION 3.11. *Let $a^{\triangleright} \in \mathcal{O}^{\triangleright}$ be a first action in a superfluous plan π_{sup} and let $a^{\triangleright}$ be privately-nondeterministic. The number of transition systems represented by a label non-preserving projection $a^{\triangleright}$ is $t_a^{\mathsf{ia}\times\mathsf{pn}} = 12^p - 6^p$.*

PROOF. The same technique as in the proof of Proposition 3.10. □

EXAMPLE. In particular, both cases hold in the running example for SL, in the first case, the result is $12^2 - 3^2 = 135$ possible transition systems and in the second case, $12^2 - 6^2 = 108$ possibilities. ⊠

All transition systems corresponding to a particular type of action are shown in Table I, which can be used to infer such information.

3.4. Leakage Estimate

Let the number of transition systems represented by a single action a without any information leaked be a constant t_a. The particular value of t_a depends on the num-

ber of variables, their domain size and other factors, such as the possibility of nondeterministic transitions. Let $\mathcal{O}^{\mathsf{ia}}$ denote the set of init-applicable actions and t_a^{ia} the number of transition systems represented by a single action a reduced by the information learned from its applicability. Similarly, we define the set $\mathcal{O}^{\mathsf{pd}}$ of privately-dependent actions, the set $\mathcal{O}^{\mathsf{pi}}$ of privately-independent actions and the set $\mathcal{O}^{\mathsf{pn}}$ of privately-nondeterministic actions. The respective number of transition systems represented by each such action a is t_a^{pd}, t_a^{pi} and t_a^{pn}, respectively (also the combined information $t_a^{\mathsf{ia}\times\mathsf{pi}}, t_a^{\mathsf{ia}\times\mathsf{pn}}$ may be used), each reduced according to the revealed information (the particular numbers were discussed in Section 3.3). Thus for each action $a \in \mathcal{O}^{\mathsf{pub}}$, we can define the number of transition systems it represents as

$$\tau_{\mathsf{post}}(a) = \min \begin{cases} t_a & \text{always} \\ t_a^{\mathsf{ia}} & \text{if } a \in \mathcal{O}^{\mathsf{ia}} \\ t_a^{\mathsf{pd}} & \text{if } a \in \mathcal{O}^{\mathsf{pd}} \\ \dots & \end{cases} \tag{5}$$

that is the minimum of the number of transition systems represented by a based on its membership in the $\mathcal{O}^{\mathsf{ia}}, \mathcal{O}^{\mathsf{pd}}, \mathcal{O}^{\mathsf{pi}}$ and $\mathcal{O}^{\mathsf{pn}}$ sets. The knowledge obtained about the actions can be combined to compute the total number of possible transition systems

$$t_{\mathsf{post}} = \prod_{a \in \mathcal{O}^{\mathsf{pub}}} \tau_{\mathsf{post}}(a) \tag{6}$$

giving us an upper bound on the remaining uncertainty as

$$H_\infty(H|L) \ = \log(t_{\mathsf{post}}) = \log \textstyle\prod_{a \in \mathcal{O}^{\mathsf{pub}}} \tau_{\mathsf{post}}(a) \ . \tag{7}$$

In the above formula, the number of possible transition systems include, not only the superfluous (and thus leaked) information, but also the a priori known information. The a priori formula for the initial uncertainty can be constructed similarly, but using the information from the projected problem only (that is t_a) and the projection $\pi^{\triangleright} = (a_1^{\triangleright}, ..., a_l^{\triangleright}, ..., a_k^{\triangleright})$ of the solution plan, where a_l is the init-applicable action and, thus, $\mathcal{O}^{\mathsf{ia}} = \{a_l\}$ resulting in

$$\tau_{\mathsf{prior}}(a) = \min \begin{cases} t_a & \text{always} \\ t_a^{\mathsf{ia}} & \text{if } a^{\triangleright} \in \mathcal{O}^{\mathsf{ia}}. \end{cases}$$

The final formula for the leaked information is obtained as the difference of the two, that is

$$H_\infty(H) - H_\infty(H|L) \ = \sum_{a \in \mathcal{O}^{\mathsf{pub}}} (\log \tau_{\mathsf{prior}}(a) - \log \tau_{\mathsf{post}}(a)) \tag{8}$$

where clearly, for the actions with no additional information revealed, we obtain $\log \tau_{\mathsf{prior}}(a) - \log \tau_{\mathsf{post}}(a) = 0$, and for actions with leaked information, we obtain $\log \tau_{\mathsf{prior}}(a) - \log \tau_{\mathsf{post}}(a) > 0$. Since $\tau_{\mathsf{prior}}(a) \geq \tau_{\mathsf{post}}(a)$ always as no information can be lost, the number of possible transition systems can only decrease. This is important, because it shows that the more actions are revealed by the superfluous plans or superfluous applicability, the more private information is leaked.

4. ANALYSIS OF PRIVACY LEAKAGE IN PP-MAP ALGORITHMS

In this section, we take a closer look on how the selected algorithms behave from the point of view of privacy leakage analysis. We focus on the Multi-Agent Forward Search (MAFS) [Nissim and Brafman 2014] family of algorithms, that is the optimal variant Multi-Agent Distributed A* (MAD-A*) [Nissim and Brafman 2012], a secure variant

Secure-MAFS [Brafman 2015] and Macro-MAFS [Maliah et al. 2016a] and other similar planners. The reason of restricting our attention to this particular family of algorithms is twofold: First, the algorithms share many common principles, which simplifies the presentation and keeps the analysis focused and second, most of the work focusing on reducing privacy leakage has been done in this family of algorithms and, thus, the comparison is most relevant and informative. We start with a general high-level outline and subsequently provide an in-depth analysis of their behavior on one particular problem we have introduced in this article and used throughout.

In this work, we aim to compute a lower bound on the privacy leakage. This means the amount of information that leaks during the execution of the algorithm. We place a number of assumptions on the agents, the environment and the algorithms used.

Honest but curious agents. This is an assumption often used in secure MPC, stating, that the agents do not diverge from the algorithm and communication protocol in order to exploit it, but deduce as much information as possible from the execution and from the communicated data.
The adversary knows the algorithm and the communication protocol. This means that the adversary can place assumptions based on the algorithm itself, such as that for each expanded state, all its successors are created and sent before expanding any further state.
Messages are received in-order. That is in the order in which they were sent.

Removing the first assumption would increase the information leakage, as the adversary could, for example, force the agent to explore the complete search space. Removing the other two assumptions would decrease the information leakage by increasing the number of possible interpretations of the received information.

In the following sections we analyze which parts of the search space are explored in the actual execution of particular algorithms and what effect do they have on the leakage of the private information.

4.1. General Method for Search-based Algorithms

Before we get into more details, let us describe the basic principle of multi-agent best-first search. In short, in the MAFS algorithm, each agent searches the state space using its own actions $a \in \mathcal{O}_i^{\mathsf{pub}} \cup \mathcal{O}_i^{\mathsf{priv}}$, an ordered open list and a closed list, as in a textbook best-first search. If a public action $a \in \mathcal{O}_i^{\mathsf{pub}}$ is used to expand a state s, the resulting state $s' = a \circ s$ is sent to all other agents $\alpha_{j \neq i}$ which have some relevant action ($a' \in \mathcal{O}_j^{\mathsf{pub}}$ s.t. $\mathsf{pre}(a')$ is satisfied in s'). The state s' is sent in the form of public projection $s'^{\triangleright}$, together with its heuristic value and an encrypted[2] private part of s' for each agent, that is, $\mathsf{enc}_\alpha(s')$ for the agent and $\mathsf{enc}_\beta(s')$ for the adversary.

When a state message is received, the encrypted information is used to reconstruct the private part of the received state respective to the receiving agent α_i. Moreover, the heuristic is recomputed from the point of view of agent α_i and the maximum is used (this step may be omitted in the case of a distributed or an inadmissible heuristic).

More formally (based on [Brafman 2015]), we say that s_0 is an i-parent of s_k if s_k is obtained from s_0 by the application of a sequence $a_1, ..., a_l$ of actions of agents other than α_i and the last action in this sequence, a_l, is public. This means that s_k is a state that would be sent by some agent to α_i, and α_i was not involved in the generation of any state between s_0 and s_k.

[2]Note that there are no implementation details on the encryption algorithm in the description of any of the planners, as none of them internally implements any encryption.

ALGORITHM 1: Privacy leakage for MAFS-based algorithms

```
1 Algorithm computePrivacyLeakage(M)
2     Reconstruct the search tree
3         Determine possible parent states
4         Determine possible applied actions
5         Determine distinct states
6     Determine superfluous action applicability and Assign actions to O^ia, O^pi, O^pd, or O^pn
7     Compute the leakage
```

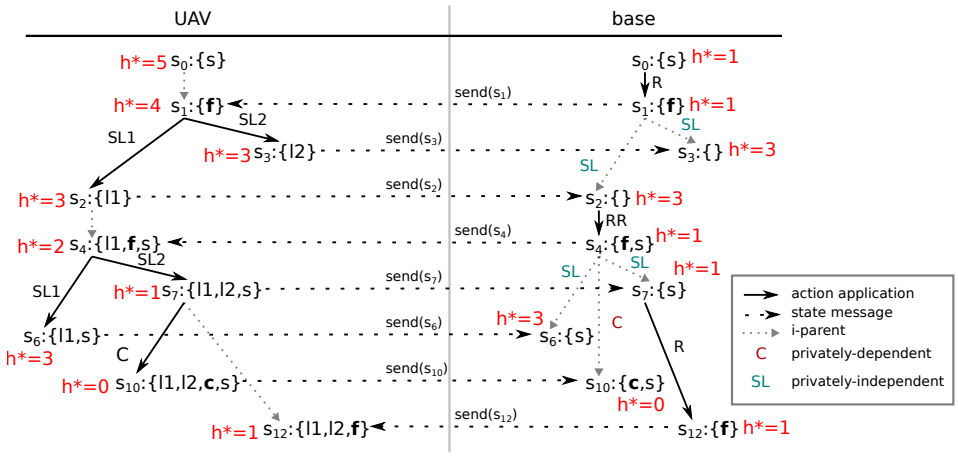


Fig. 4. Search tree of a particular run of MAFS (or MAD-A*) using the optimal projected heuristic h^*.

Here, we present a high-level method for computing the privacy leakage of MAFS-based algorithms, shown in Algorithm 1. We focus on the particular instantiation of the steps in the following sections. In general, the first step is to attempt to reconstruct the search tree of the particular execution of the search-based algorithm (an example of the search tree of MAD-A* is shown in Figure 4). Reconstructing the search tree means that the parent states and applied actions are identified. Moreover, it is important to determine as much publicly equivalent but distinct states as possible in order to be able to later identify privately-dependent, privately-independent and privately-nondeterministic actions and compute the leaked information based on the computation in Section 3.

4.2. MAFS and MAD-A*

According to [Nissim and Brafman 2014], MAFS is at least weak privacy-preserving, as no private information is explicitly sent to other agents. But MAFS is not strong privacy-preserving as some additional information can be deduced from the search. Here, we analyze which information it is.

EXAMPLE. For the analysis of the example, we use the MAD-A* algorithm. As MAD-A* is a heuristic search, we need to assign a heuristic value to each state. We will use the optimal projected heuristic, that is, h^* computed on the projected problem (of each agent). Figure 4 shows a portion of the search space explored by the algorithm using this heuristic estimate (shown as h^* in the figure). ⊠

Let us now focus on the individual steps of Algorithm 1 and their relation to the results of Section 3.3.

Search tree reconstruction in MAFS. In MAFS and MAD-A*, it is more important to reconstruct and analyze the search tree than to analyze the (partial) plans. The search tree is a tree determined by the parent/child relation of the states during the particular search execution, as shown if Figure 4.

The important information in the search tree is which state s_{k-1} is the parent of the state s_k and what action a was applied by the agent to generate s_k from s_{k-1}. As both the agent and the adversary are using the same algorithm (e.g., MAFS), each state s_k also has a private part of the adversary β, that is $\mathsf{enc}_\beta(s_k)$. This part can be a pointer to the i-parent state s_0, which contains the right private part (possibly encrypted). If a sequence of (public) actions was applied by the agent α between s_0 and s_k, then $s_0 \neq s_{k-1}$. To determine the actual parent, the following proposition holds:

PROPOSITION 4.1. *For an agent α and an adversary β, let s_k be a state sent by α and received by β and let s_0 be the i-parent of s_k known to β. Then the parent state s_{k-1} of s_k was received by β before s_k and after s_0 which is also an i-parent of s_{k-1}.*

PROOF. Trivial if $s_{k-1} = s_0$. Otherwise a sequence $a_1, ..., a_k$ of actions of agent α must have been applied by α on s_0 to generate s_k. The state generated by action a_{k-1} is the parent s_{k-1} of s_k. As we assume public actions only, all intermediate states between s_0 and s_k are also sent to β and, thus, the same holds for the parent s_{k-1}. Based on the MAFS algorithm, a state is fully expanded before expanding the next state from the open list. As we assume all messages retain the order in which they were sent, then clearly s_{k-1} was sent before s_k and, thus, also received before s_k. □

Based on Proposition 4.1, the parent s_{k-1} of the state s_k is some of the states received before s_k such that s_0 is an i-parent of s_{k-1}. Thus, the parent state might not be determined unambiguously, but instead the adversary knows a set of potential parent states of s_k.

Determining the applied action is even trickier. For two states s', s, where s' is a (potential) parent of s, we need to find a projected action $a^\triangleright \in O^\triangleright$ for which $\mathsf{pre}(a^\triangleright)$ holds in s' and $\mathsf{eff}(a^\triangleright)$ is consistent with s. This means that we can only determine the public projection $\langle \mathsf{pre}(a^\triangleright), \mathsf{eff}(a^\triangleright) \rangle$ of the used action, which corresponds to the label non-preserving projection. In the case where we know a set of possible parent states, the result is also a set of possibly applied actions and the exact parent state might not be known (for each possible action there is a subset of candidate parent states).

EXAMPLE. In the example in Figure 4, the public projections of actions of agent α_{UAV} are $\langle \{\mathsf{f} = \mathsf{T}\}, \{\mathsf{f} = \mathsf{F}\} \rangle$ and $\langle \{\}, \{\mathsf{c} = \mathsf{T}\} \rangle$. As $\mathsf{c} = \mathsf{F}$ holds in state s_3, the second projection is not applicable. The first projection is applicable in s_1 and the effect holds in s_3. The corresponding projected action is SL. The same holds for the parent s_1 and child s_2 and similarly for s_4 and s_7, s_6. The i-parent of state s_{10} is s_4 and the potential parent states are, thus, s_4, s_6, s_7. As we know, there is no single action which would change $\{\mathsf{f}\}$ to $\{\mathsf{c}\}$, then clearly s_4 is not the parent. For the parents s_7, s_6 and child s_{10}, only the projection $\langle \{\}, \{\mathsf{c} = \mathsf{T}\} \rangle$ corresponding to the action C is applicable. The only remaining uncertainty is which of the two states is the true parent state. ⊠

Superfluous distinct states in MAFS. The goal of encryption in MAFS and MAD-A* is twofold: The first is to hide the parent of the state and the second is to disallow the adversary to determine whether two states are globally equal or not. The first goal is achievable without any effect on the algorithm performance. Parents of states are necessary only for the plan reconstruction process and do not have to be shared to do so. To achieve the second goal is much trickier. In order to maintain soundness, the agent has to be able to reconstruct the private part of each state unambiguously, this means that two different private parts of a state can never result in the same encrypted value $\mathsf{enc}_\alpha(s) = \mathsf{enc}_\alpha(s')$.

The simplest approach is to have the private part of each state s encrypted as a unique value $\mathsf{enc}_\alpha(s)$. In that case, no information is leaked as the adversary cannot determine which states are the same. But this approach may seriously influence the algorithm performance or even make it incomplete by introducing infinite loops, as the other agent is not able to close already visited states. As in our analysis, we are aiming for a lower bound of the privacy leakage, we can safely assume that the agents use the unique encryption method and no information about superfluous distinct states leaks. By bounding the encryption values, the leakage can only increase.

Both MAFS and MAD-A* expand the states in the open list in the order defined by a function $f(s)$. In the case of MAD-A*, the function is defined as $f(s) = g(s) + h(s)$, where $g(s)$ is the distance (number of actions) from the initial state s_I to s and $h(s)$ is an admissible heuristic estimate of the remaining distance from s to the nearest goal state. In the case of MAFS, the $g(s)$ function can be omitted to obtain a Greedy Best-First Search (GBFS), where $f(s) = h(s)$, so that the states are ordered solely by the heuristic information. Based on the defined functions, we can use the following to infer different states.

PROPOSITION 4.2. *Let s, s' be two states and $h(s)$ a deterministic heuristic function. Then $f(s) \neq f(s') \vee g(s) \neq g(s') \vee h(s) \neq h(s') \implies s \neq s'$.*

PROOF. The computation of a heuristic function, the distance from the initial state, or their sum, never results in a different value for the same state. □

Also, the set of successor states can be used to determine equivalence.

PROPOSITION 4.3. *Let S, S' be the sets of all successors of the states s, s' respectively. If $S \neq S'$ then $s \neq s'$.*

PROOF. Let us assume $s = s'$, but $S \neq S'$. But each state was expanded using the same set of actions and, thus, $S = S'$. □

EXAMPLE. In the example in Figure 4, s_6 and s_7 differ based on Proposition 4.2 as $h(s_6) \neq h(s_7)$ and also s_3 and s_7 as again $h(s_3) \neq h(s_7)$. But based on the information that $s_3 \neq s_7$ and Proposition 4.3, we can infer that $s_4 \neq s_1$ as their child states are different. ⊠

Superfluous action applicability in MAFS. MAFS and MAD-A* use a label non-preserving projection, thus, each action a (with a distinct public projection) represents 15^p transition systems, where p is the number of (binary) private variables (see Section 3.3 for details). In our example with two private variables, the number is $15^2 = 225$.

An action, which is known to be applied in the initial state s_I (or a state s_l resulting from the application of a sequence of actions of the adversary actions on s_I) reveal the information that it is applicable in some particular state where we can assume a fixed value of all variables. Each such init-applicable action ($\mathcal{O}^{\mathsf{ia}}$) represents, at most, $t_a^{\mathsf{ia}} = 12^p$ transitions (Proposition 3.3).

An action, which is known to be applied on two states s, s' known to be distinct, is privately-independent (Definition 3.6) in, at least, one variable, as it does not depend on the variable in which s and s' differ. Such an action represents $t_a^{\mathsf{pi}} = 15^p - 6^p$ transitions (Proposition 3.7). Conversely, an action, which is known to be applied on state s and not on state s' such that s, s' are publicly equivalent, but known to be distinct is a privately-dependent action (Definition 3.4) as it depends on the variable in which s and s' differ. Such an action represents $t_a^{\mathsf{pd}} = 15^p - 9^p$ transitions (Proposition 3.5).

Furthermore, if we observe, that the application of a single action on a single state results in two distinct states, we found a privately-nondeterministic action (Definition 3.8), that is, a projected action which represents at least two actions with private

effects different in, at least, one variable. Such an action represents $t_a^{\mathsf{pn}} = 15^p - 8^p$ transitions (Proposition 3.9).

EXAMPLE. In the example in Figure 4, the action SL is applicable in the state s_1 and, thus, is init-applicable and represents $\tau(\mathsf{SL}) = t_a^{\mathsf{ia}} = 12^2 = 144$ transitions. We can also see that the action SL can be applied in both s_1 and s_4, which are known to be different states and, thus, is privately-independent in, at least, one variable. Moreover, we can observe, that the action C is applied on state s_6 or s_7, but is never applied on state s_2, which is publicly equivalent and for which all child states were already generated (that is only s_4). This can only mean that either s_6 or s_7 (or both) are distinct from s_2 and this distinction affects the applicability of C. Thus, C is privately-dependent. Finally, the action SL is applied in s_4 and results in two distinct states s_6 and s_7 and, thus, is privately-nondeterministic. ⊠

Based on the above information and on the fact that SL is the first action of agent α_{UAV} in the resulting plan, the combined knowledge of the action properties can be used and the final number of transition systems represented by the actions SL and C computed based on Equation 5 as

$$\begin{aligned}\tau_{\mathsf{post}}(\mathsf{SL}) &= \min(t_a, t_a^{\mathsf{ia}}, t_a^{\mathsf{pi}}, t_a^{\mathsf{pn}}, t_a^{\mathsf{ia}\times\mathsf{pi}}, t_a^{\mathsf{ia}\times\mathsf{pn}}) = \min(144, 189, 161, 135, 108) = 108\\ \tau_{\mathsf{post}}(\mathsf{C}) &= \min(t_a, t_a^{\mathsf{pi}}) = 144\end{aligned}$$

Information leakage in MAFS. Now, we can combine the results to obtain the total information leaked. Based on Equation 5 and Equation 6, we conclude that $t_{\mathsf{prior}} = 12^p 15^p = 32400$ and because the only information learned from the projection $\Pi^{\triangleright}$ and the plan $\pi^{\triangleright}$ is that action SL is init-applicable, the posterior information is

$$t_{\mathsf{post}} = \prod_{a \in \mathcal{O}^{\mathsf{pub}}} \tau_{\mathsf{post}}(a) = \tau_{\mathsf{post}}(\mathsf{SL})\tau_{\mathsf{post}}(\mathsf{C}) = 108 \cdot 144 = 15552$$

Thus, the lower bound on the information leakage of this particular execution of the MAFS algorithm on this particular example is

$$H_\infty(H) - H_\infty(H|L) = \log \frac{t_{\mathsf{prior}}}{t_{\mathsf{post}}} \cong \log(2) = 1$$

bits of information. Based on Equation 3, we obtain the expected probability of guessing the correct transition system as

$$2^{-H_\infty(H|L)} = 2^{-\log t_{\mathsf{post}}} = \frac{1}{t_{\mathsf{post}}} = \frac{1}{15552} \cong 6 \times 10^{-5}$$

Notice that in this particular example we were able to identify the applicability of all actions and thus all the information according to Section 3.3 has leaked.

4.3. Secure-MAFS

The theoretical Secure-MAFS [Brafman 2015] algorithm and its (generalized) implementation Macro-MAFS [Maliah et al. 2016a] are variants of the MAFS algorithm aimed at improving the privacy of the planning process.

The main difference of Secure-MAFS in comparison to MAFS is that it sends only states, which differ from previously sent states in the private part of other agents. In other words, each state with a unique public part and other agents' private parts is sent, at most, once. The transitions caused by the actions of other agents are stored (in the form of action macros) and applied instead of sending the state.

EXAMPLE. An execution of the Secure-MAFS algorithm on the running example is shown in Figure 5. When the agent α_{UAV} receives the state s_1, it stores the transition $\{\mathsf{f} = \mathsf{F}\} \rightarrow \{\mathsf{f} = \mathsf{T}\}$ together with the information about the private part of α_{base}. The

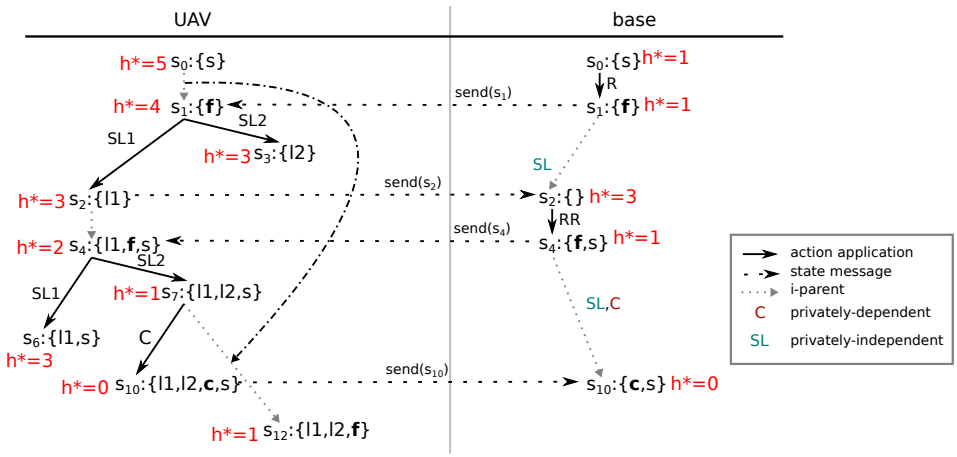


Fig. 5. Search tree of the Secure-MAFS algorithm.

learned (macro) transition can be later applied on any state which satisfies the left side and the α_{base} private part condition. Any further state equals to $\{\text{s}\}$ (the private part of α_{base} is encrypted) does not have to be sent, the same holds for $\{\}$ later. This significantly reduces the need of communication to only one more state s_2 together with the goal state $s_{10} = \{\mathbf{c}, \text{s}\}$. Now let us analyze what effect the reduction of communication has on the privacy leakage. ⊠

Superfluous distinct states in Secure-MAFS. One major difference of security handling from MAFS in Secure-MAFS is the role of encryption of the private states. In order to be able to determine that two states s, s' do not differ in the public part and the private part of other agents, the encryption of the private parts must be bijective. If it is not so, the effect of the Secure-MAFS principle is lessened. Therefore, any two states, which are publicly equivalent, but differ on the private part of some agent can be distinguished.

On the contrary, no two states s, s' which differ in the private part of an agent α_i only are sent to other agents. This means that even though the information about state equality is revealed, its effect for the information leakage is significantly reduced.

EXAMPLE. In the example in Figure 5, only the publicly equivalent states which differ in the private part of the agent α_{base} are sent, in particular s_0 and s_2. ⊠

Search tree reconstruction in Secure-MAFS. As described above, the search tree reconstruction is simplified by the fact that the equality of states is revealed, but is complicated by the fact that significant parts of the search tree are not communicated. Also, as the received state s' might be a result of application of a sequence of multiple public actions on the (known) parent state s, thus, Proposition 4.1 does not hold for Secure-MAFS. This renders inference of the applied action (or, in fact, a sequence of actions) significantly more complicated (and computationally more intensive).

EXAMPLE. In the example in Figure 5, the only possible sequence of actions responsible for the transitions are $\langle \text{SL} \rangle$ and $\langle \text{SL}, \text{C} \rangle$, respectively. ⊠

Superfluous action applicability in Secure-MAFS. In general, it may not even be possible to infer the sequence of actions used to generate the transitions. The action applicability inference is significantly complicated as well. As no two states which differ in the private part of the agent α only are ever communicated, a privately-independent action can be determined only in the case of two states which differ in both private parts of

the agent and of the adversary. To do so, the adversary would need to observe the action applied on at least two such states.

EXAMPLE. In the example, the candidate states are s_1 and s_4. As the heuristic estimate is equal for both states, the difference of s_1, s_4 can be determined only if the uniqueness of their private parts is retained. Assuming so, the action SL can be determined to be privately-independent. ⊠

A similar restriction holds for privately-dependent actions. The adversary would need to see such an action applied on state s and never applied on a publicly equivalent but distinct state s', where s and s' differ in the private part of the adversary.

EXAMPLE. In the example in Figure 5, the adversary only knows that C is applied on some state s'_4 resulting from the application of SL on s_4. There is no information about its applicability on another state and, thus, its private dependency is not revealed. ⊠

THEOREM 4.4. *Let $a^{\triangleright}$ be a privately-nondeterministic action of agent α. In the Secure-MAFS algorithm, this information never leaks.*

PROOF. From Definition 3.8, the privately-nondeterministic action $a^{\triangleright}$ represents two actions a', a'' such that $\mathsf{eff}(a') \neq \mathsf{eff}(a'')$. Thus, the application of such an action $a^{\triangleright}$ on state s results in two publicly equivalent but distinct states $s' = a' \circ s$ and $s'' = a'' \circ s$. As a', a'' do not influence the private parts of other agents, only the different parts of s' and s'' are the parts private to the agent α. But, this means that in Secure-MAFS, only one of s', s'' is sent by α and, thus, the information about $a^{\triangleright}$ is never revealed. □

COROLLARY 4.5. *Let $\mathcal{M}$ be a multi-agent planning problem and $a^{\triangleright}$ a privately-nondeterministic action. If $a^{\triangleright}$ leaks that information in every execution of the MAFS algorithm, then Secure-MAFS preserves more privacy than MAFS on $\mathcal{M}$.*

Information leakage in Secure-MAFS. Based on the above analysis, we can compute the total information leaked. Similarly to Section 4.2, we first compute the transition systems represented by each action based on the leaked (and prior) information as $\tau_{\mathsf{post}}(\mathsf{SL}) = \min(t_a, t_a^{\mathsf{pi}}, t_a^{\mathsf{ia}}, t_a^{\mathsf{ia}\times\mathsf{pi}}) = \min(189, 144, 135) = 135$ and $\tau_{\mathsf{post}}(\mathsf{C}) = \min(t_a) = 225$. Notice that action C now leaks no information. The prior information is the same as in Section 4.2 that is $t_{\mathsf{prior}} = 12^p 15^p = 32400$. The posterior information is

$$t_{\mathsf{post}} = \prod_{a \in \mathcal{O}^{\mathsf{pub}}} \tau_{\mathsf{post}}(a) = \tau_{\mathsf{post}}(\mathsf{SL})\tau_{\mathsf{post}}(\mathsf{C}) = 135 \cdot 225 = 30375$$

Thus, the lower bound on the information leakage of Secure-MAFS is

$$H_\infty(H) - H_\infty(H|L) = \log \frac{t_{\mathsf{prior}}}{t_{\mathsf{post}}} \cong \log(1.1) = 0.1$$

bits. The expected probability of guessing the correct transition system is

$$2^{-H_\infty(H|L)} = 2^{-\log t_{\mathsf{post}}} = \frac{1}{t_{\mathsf{post}}} = \frac{1}{30375} \cong 3 \times 10^{-5}$$

5. CONCLUSION AND DISCUSSION OF THE RESULTS

In terms of the sources of leakage we focused on, the Secure-MAFS algorithm leaks less (or equal to) information than MAFS in general, as it does not leak information about the privately-nondeterministic actions. We have also seen that the Secure-MAFS algorithm may prevent leakage of some information about action applicability.

In the particular example, the Secure-MAFS algorithm leaked some information (the applicability of the SL action), but this leakage is very low in the final computation result. Notice that if the adversary α_{base} had no private variables, the algorithm would leak no information at all. We have shown, that the tools which were able to deduce

a significant amount of information from the execution of plain MAFS are much less powerful in the case of Secure-MAFS, but strong enough to show nonzero leakage.

To evaluate the actual scale and importance of the computed results, we need to compute the maximal possible leakage, that is what is the leakage if all private information has leaked. We obtain the maximal possible leakage by putting $t_{\mathsf{post}} = 1$, in which case the adversary knows the exact transition system of the agent's problem. In the case of the example, the result is $H_\infty(H) - H_\infty(H|L) = \log \frac{t_{\mathsf{prior}}}{1} = \log(32400) \cong 15$. The leakage of the MAFS algorithm is 1 bit out of 15, which is not particularily high, but is significant (approximately 7% of the information has leaked), whereas in the case of the Secure-MAFS algorithm, the leakage of 0.1 bits is rather insignificant ($<$1%).

Our approach is able to identify the lower-bound on the leakage, which can be tightened by improving the algorithm and including more sources of leakage. Nevertheless, Theorem 4.4 provides us with an upper bound in the sense that the privately-nondeterministic actions never leak in Secure-MAFS. Overall, this work provides the first and crucial steps for formal treatment of privacy leakage in PP-MAP, ready to be built-on and improved by future tightening the upper and lower bounds.

References

Ronen I. Brafman. 2015. A Privacy Preserving Algorithm for Multi-Agent Planning and Search. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*. 1530–1536.

Ronen I. Brafman and Carmel Domshlak. 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*. 28–35.

Christelle Braun, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2009. Quantitative Notions of Leakage for One-try Attacks. *Electr. Notes Theor. Comput. Sci.* 249 (2009), 75–91.

Boi Faltings, Thomas Léauté, and Adrian Petcu. 2008. Privacy Guarantees through Distributed Constraint Satisfaction. In *Proceedings of the International Conference on Intelligent Agent Technology*. 350–358.

Rachel Greenstadt, Jonathan P. Pearce, and Milind Tambe. 2006. Analysis of Privacy Loss in Distributed Constraint Optimization. In *Proceedings of The Twenty-First National Conference on Artificial Intelligence (AAAI)*. 647–653.

Patrik Haslum and Hector Geffner. 2000. Admissible Heuristics for Optimal Planning. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems, ICAPS*. 140–149. http://www.aaai.org/Library/AIPS/2000/aips00-015.php

Shlomi Maliah, Guy Shani, and Ronen Brafman. 2016a. Online Macro Generation for Privacy Preserving Planning. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling*.

Shlomi Maliah, Guy Shani, and Roni Stern. 2016b. Collaborative privacy preserving multi-agent planning. *Autonomous Agents and Multi-Agent Systems* (2016), 1–38.

Raz Nissim and Ronen I. Brafman. 2012. Multi-agent A* for parallel and distributed systems. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, AAMAS'12*. 1265–1266.

Raz Nissim and Ronen I. Brafman. 2014. Distributed Heuristic Forward Search for Multi-agent Planning. *J. Artif. Intell. Res. (JAIR)* 51 (2014), 293–332.

Geoffrey Smith. 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.

Michal Štolba, Jan Tožička, and Antonín Komenda. 2016a. Secure Multi-Agent Planning. In *Proceedings of the 1st International Workshop on AI for Privacy and Security*. ACM, 11.

Michal Štolba, Jan Tožička, and Antonín Komenda. 2016b. Secure Multi-Agent Planning Algorithms. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI'16)*. 1714–1715.

Alejandro Torreño, Eva Onaindia, and Oscar Sapena. 2014. FMAP: Distributed cooperative multi-agent planning. *Appl. Intell.* 41, 2 (2014), 606–626.

Jan Tožička, Jan Jakubův, Antonín Komenda, and Michal Pěchouček. 2015. Privacy-concerned multiagent planning. *Knowledge and Information Systems* (2015), 1–38 (pre–print).

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

Andrew C. Yao. 1982. Protocols for Secure Computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, SFCS*. 160–164.