ε-Strong Privacy Preserving Multiagent Planner by Computational Tractability

Jan Tožička, Antonín Komenda and Michal Štolba
Department of Computer Science, Czech Technical University in Prague,
Karlovo náměstí 13, 121 35, Prague, Czech Republic

Keywords: Automated Planning, Multiagent Systems, Privacy, Security.

Abstract: Classical planning can solve large and real-world problems, even when multiple entities, such as robots, trucks or companies, are concerned. But when the interested parties, such as cooperating companies, are interested in maintaining their privacy while planning, classical planning cannot be used. Although, privacy is one of the crucial aspects of multi-agent planning, studies of privacy are underepresented in the literature. A strong privacy property, necessary to leak no information at all, has not been achieved by any planner in general yet. In this contribution, we propose a multiagent planner which can get arbitrarily close to the general strong privacy preserving planner for the price of decreased planning efficiency. The strong privacy assurances are under computational tractability assumptions commonly used in secure computation research.

1 INTRODUCTION

A multiagent planning problem is a problem of finding coordinated sequeces of actions of a set of entities (or agents), so that a set of goals is fulfilled. If the environment and actions are deterministic (that is their outcome is unambiguously defined by the state they are applied in), the problem is a deterministic multiagent planning problem (Brafman and Domshlak, 2008). Furhtemore, if the set of goals is common to all agents and the agents cooperate in order to achieve the goals, the problem is a cooperative multiagent planning problem. 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 only the information necessary for their cooperation, but not the information about their inner processes. Such privacy constraints are respected by privacy preserving multiagent planners.

A number of privacy preserving multiagent planners has 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 of the mentioned planners claim to be privacy-preserving, proving such claims was rather scarce. The privacy of MAFS is discussed in (Nissim and Brafman, 2014) and expanded upon in (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).

Apart from a specialized privacy leakage quantification by (Van Der Krogt, 2009) (which is not practical as it is based on enumeration of all plans and also is not applicable to MA-STRIPS in general), the only rigorous definition of privacy so far was proposed in (Nissim and Brafman, 2014) and extended in (Brafman, 2015). The authors present two notions, weak and strong privacy preservation. Weak privacy preservation forbids only explicit communication of the private information, which is trivial to achieve and provides no security guarantees. The strong privacy preservation forbids leakage of any information allowing other agents to deduce any private information at all.

2 MULTI-AGENT PLANNING

The most common model for multiagent planning 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 and we also generalize the definition to multi-valued variables. Formally, for a set of agents $\mathcal{A}$, a problem $\mathcal{M} = \{\Pi_i\}_{i=1}^{|\mathcal{A}|}$

ToÅ¿iÄ■ka J., Komenda A. and Åătolba M.
Ît-Strong Privacy Preserving Multiagent Planner by Computational Tractability.
DOI: 10.5220/0006176400510057
In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*, pages 51-57
ISBN: 978-989-758-219-6

is a set of agent problems. An agent problem of agent $\alpha_i \in \mathcal{A}$ is defined as

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

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. $|\mathsf{dom}(V)| = 2$), 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 $\mathcal{V}^{\mathsf{pub}} \cap \mathcal{V}_i^{\mathsf{priv}} = \emptyset$. A complete assignment over $\mathcal{V}$ is a *state*, partial assignment over $\mathcal{V}$ is a partial state. We denote $s[V]$ as the value of V in a (partial) state s and $\mathsf{vars}(s)$ as the set of variables defined in s. The state s_I is the initial state and $s_\star$ is a partial state representing the goal condition, that is if for all variables $V \in \mathsf{vars}(s_\star)$, $s_\star[V] = s[V]$, s is a goal state.

The set O_i of actions comprises of a set O_i^{priv} of private actions of α_i, a set O_i^{pub} of public actions of α_i and a set O^{proj} of public projections of other agents' actions. O_i^{pub}, O_i^{priv}, and O^{proj} are pairwise disjoint. An action is defined as a tuple $a = \langle \mathsf{pre}(a), \mathsf{eff}(a) \rangle$, where $\mathsf{pre}(a)$ and $\mathsf{eff}(a)$ are partial states representing the precondition and effect respectively. 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 $a \circ s$, 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. As we often consider the planning problem from the perspective of agent α_i, we omit the index i.

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

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

Finally, we define the solution to Π and $\mathcal{M}$. A sequence $\pi = (a_1, ..., a_k)$ of actions from 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. Such π does not have to be the global solution to $\mathcal{M}$, as the actions of other agents (O^{proj}) are used only as public projections and are missing private preconditions and effects of other agents. The public projection of π is defined as $\pi^\rhd = (a_1^\rhd, ..., a_k^\rhd)$ with ε actions omitted.

From the global perspective of $\mathcal{M}$ a public plan $\pi^\rhd = (a_1^\rhd, ..., a_k^\rhd)$ is a sequence of public projections of actions of various agents from $\mathcal{A}$ such that the actions are sequentially applicable with respect to $\mathcal{V}^{\mathsf{pub}}$ starting in $s_I^\rhd$ and the resulting state satisfies $s_\star^\rhd$. A public plan is α_i-extensible, if by replacing $a_{k'}^\rhd$ s.t. $a_{k'} \in O_i^{\mathsf{pub}}$ by the respective $a_{k'}$ and adding $a_{k''} \in O^{\mathsf{priv}}$ to required places we obtain a local plan (solution) to Π_i. According to (Tožička et al., 2015), a public plan $\pi^\rhd$ α_i-extensible by all $\alpha_i \in \mathcal{A}$ is a global solution to $\mathcal{M}$.

2.1 Privacy

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 only the information in $\Pi^\rhd$. Even if not communicated, the adversary may deduce the existence and values of private variables, preconditions and effects from the (public) information communicated.

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

2.2 Secure Computation

In general any function can be computed securely (Ben-Or et al., 1988; Yao, 1982; Yao, 1986). In this contribution, we focus on more narrow problem of *private set intersection* (PSI), where each agent has a private set of numbers and they want to securely compute the intersection of their private sets while not disclosing any numbers which are not in the intersection. The *ideal PSI* supposes that no knowledge is transfered between the agents (Pinkas et al., 2015).

Ideal PSI can be solved with trusted third party which receives both private sets, computes the intersection, and sends it back to agent. As long as the third party is honest, the computation is correct and no information leaks.

In literature (e.g., (Pinkas et al., 2015; Jarecki and Liu, 2010), we can find several approaches how the ideal PSI can be solved without trusted third

party. Presented solutions are based on several computational hardness assumptions, e.g., intractable large number factorization, DiffieHellman assumption (Diffie and Hellman, 1976), etc. All these assumptions break when an agent has access to unlimited computation power, therefore all the results hold under the assumption that $P \neq NP$, in other words by computational intractability of breaking PSI.

3 STRONG PRIVACY PRESERVING MULTIAGENT PLANNER

Multiagent planner fulfilling the strong privacy requirement forms the lower bound of knowledge exchanged between the agents. Agents do not leak any knowledge about their internal problems and thus their cooperation cannot be effective, nevertheless, a strong privacy preserving multiagent planner is an important theoretical result that could lead to better understanding of privacy preserving during multiagent planning and consequentially also to creation of more privacy preserving planners.

In this contribution, we present a planner that is not strong privacy preserving but can be arbitrarily close to it. We focus on planning using coordination space[1] search (Stolba et al., 2016) and thus we will define the terms in that respect. In the following definitions and proofs we suppose that there are two honest agents α_1 and α_2. We will consider the perspective of a curious agent α_1 trying to detect the private knowledge of α_2 for the simplicity of the presentation, but all holds for both curious agents and also for a larger group of agents. Similarly to (Brafman, 2015), we also assume that $O^{\text{priv}} = \emptyset$. This assumption can be stated WLOG as each sequence of private actions followed by a public action can be compiled as a single public action.

Definition 1 (Public plan acceptance)**.** Public plan acceptance $\phi(\pi)$ *is a probability known to agent* α_1 *whether plan* π *is* α_2*-extensible.*

When the algorithm starts, α_1 has some a priori knowledge $\phi^0(\pi)$ about the acceptance of plan π by agent α_2 (e. g. 50 % probability of acceptance of each plan in the case when α_1 knows nothing about α_2). At the end of the algorithm execution this knowledge changes to $\phi^\perp(\pi)$. Obviously, every agent knows that the solution π^* agents agreed on is extensible and thus it is accepted by every agent, i.e. $\phi^\perp(\pi^*) = 1$. The difference between the α_1's a priori knowledge and the final knowledge represents knowledge which leaked from α_2 during their communication.

Definition 2 (Leaked knowledge)**.** Leaked knowledge *during the execution of a multiagent planner leading to a solution* π^* *received by agent* α_1 *is*

$$\lambda = \sum_{\pi \neq \pi^*} \left| \phi^\perp(\pi) - \phi^0(\pi) \right|.$$

Definition of algorithm's *leaked knowledge* allows us to formally define *strong privacy* of coordination space algorithm.

Definition 3 (Strong privacy)**.** *Coordination space algorithm is* strong privacy preserving *if* $\lambda = 0$.

Definition 4 (ε-strong privacy preserving planner)**.** *For any given* $\varepsilon > 0$ *the algorithm can be tuned to leak less then* ε *knowledge, i.e.* $\lambda < \varepsilon$.

Our proposed algorithm (Algorithm 1) is based on the *generate-and-test* principle. All agents sequentially generate solutions to their problems at Line 4. How to generate new solutions of Π_i to achieve desired properties is discussed in respective proofs. Then the agents create public plans by making public projections of their solutions. Created public plans are then stored in a set Φ_i of α_i-extensible public plans. Agents need to continuously check whether there are some plans in the intersection of these sets. It is important to compute the intersection without disclosing any information about plans which do not belong to this intersection. Plans in the intersection are guaranteed to be extensible and thus the agents can extend them to local solutions. If the intersection is empty, no agent can infer any knowledge about the acceptance of proposed plans since it is possible that the other agent just did not generated the corresponding plans yet.

Algorithm 1: ε-Strong privacy preserving multiagent planner.

```
1  Function SecureMAPlanner(Π_i) is
2      Φ_i ← ∅;
3      loop
4          π ← generate new solution of Π_i;
5          Φ_i ← Φ_i ∪ {π^▷};
6          Φ ← secure ( ⋂_{α_j ∈ 𝒜} Φ_j );
7          if Φ ≠ ∅ then
8              return Φ;
9          end
10     end
11 end
```

[1] Coordination space contains all the solutions of public problem $\Pi^\triangleright$.

Theorem 1 (Soundness and completeness)**.** *Algorithm* `SecureMAPlanner()` *is sound and complete.*

Proof. Every public plan returned by the algorithm is extensible, because it is α_i-extensible by every agent, and thus it can be extended to valid local solutions.

A new plan is added to the plan set Φ_i under assumption that underlying planner generating new local solutions is complete. Let us suppose that π is a solution of length l of agent's local problem Π_i, such that $\pi^{\triangleright}$ is extensible. If the underlying planner is systematic, preferring shorter plans, (e.g. breadth-first search (BFS)) then it has to generate $\pi^{\triangleright}$ in finite time since there is only finite number of different plans of length at most l. Thus `SecureMAPlanner()` with systematic local planner ends in finite time when $\mathcal{M}$ has some solution. □

Theorem 2 (ε-strong privacy)**.** *Algorithm* `SecureMAPlanner()` *is* ε*-strong privacy preserving private when ideal PSI is used.*

Proof. Agents have to communicate only when computing the intersection of their plan sets.

Firstly, both agents encode public projections of their plans into a set of numbers using the same encoding. Then, they just need to compare two sets of numbers representing their sets of plausible public plans, in other words they need to compute ideal PSI (Pinkas et al., 2015; Jarecki and Liu, 2010).

No private knowledge leaks during ideal PSI in a single iteration. Nevertheless, there can be private knowledge leakage when the algorithm continues several for iterations when the agent uses systematic plan generation (which is in the completeness requirement).

Let us suppose both agents use BFS to solve local planning problem, but similar reasoning can be used for any systematic solver of Π_1. Agent α_1 adds its local solution π_1 to the Φ_1 in the first iteration. Agent α_2 also provides some Φ_2, but α_1 knows only that the intersection is empty. Then α_1 adds longer plan π'_1 to its Φ_1. Let us suppose that after few iterations the intersection is non-empty and contains π'_1 only. Then α_1 knows that agent α_2 does not accept π_1, because it would had to add it to Φ_2 before adding π'_1.

Agent α_2 could decrease the certainty of α_1 about the infeasibility of π_1 by interleaving the plans generated by BFS with other (possibly randomly) generated local solutions. Certainly, this would not breach the completeness as it would at most double the number of iterations before non-empty intersection is found. From the privacy perspective, α_1 cannot be sure which of the following cases happened: either *(i)* π'_1 has been generated by BFS and then, similarly to the previous case, α_2 does not accept π_1, or *(ii)* π'_1 has been generated by another unsystematic solver and it is still possible that α_2 would generate π_1 at some time in future and thus α_1 cannot deduce anything. Although α_1 cannot be sure which is the case, this information still changes the probabilities of possible private structures of α_2 problem and thus some information leaks.

Obviously, α_2 could further decrease the amount of leaked information by adding more (but still just finitely many) unsystematically generated plans between the plans generated systematically. This way the leaked information can be arbitrarily decreased and thus we just need to compute how many plans should be inserted to ensure that the total leakage is less than ε.

Having $\Theta^{\mathsf{REF}}_{\alpha_2}(l)$ representing a set of plans of length l proposed by α_1 and provably refused by α_2[2], we can estimate the leakage of the algorithm before finding a solution π^* as follows:

$$\lambda \leq \frac{\sum_{1 \leq l < |\pi^*|} |\Theta^{\mathsf{REF}}_{\alpha_2}(l)|}{k+1},$$

where k is number of unsystematically generated plans between two plans generated by BFS. There are two reasons why the agent α_2 cannot use this formula to calculate k to keep: $\lambda < \varepsilon$. Firstly, $\Theta^{\mathsf{REF}}_{\alpha_2}(l)$ is not known to α_2 as it cannot know how many plans of length l are acceptable for α_1. α_2 can overestimate this value as the number of all possible public plans of length l minus plans that are acceptable by itself, or further overestimate it by number of public plans of length l: $|\Theta^{\mathsf{ALL}}(l)| - |\Theta^{\mathsf{ACC}}_{\alpha_1}(l)| \leq |\Theta^{\mathsf{ALL}}(l)|$. Secondly, the length of the solution $|\pi^*|$ is also not known in advance. This problem can be easily avoided by allowing $\varepsilon/2$ leakage for the plans of length 1, $\varepsilon/4$ for plans of length 2, ..., $\varepsilon/2^l$ for plans of length l, which certainly yields in total leakage bellow ε.

$$\begin{aligned}
\lambda &< \varepsilon \\
\sum_{1 \leq l < |\pi^*|} \frac{|\Theta^{\mathsf{REF}}_{\alpha_2}(l)|}{k+1} &< \sum_{l < |\pi^*|} \frac{\varepsilon}{2^l} \\
\forall l : \frac{|\Theta^{\mathsf{ALL}}(l)|}{k+1} &< \frac{\varepsilon}{2^l} \\
\forall l : 2^l \cdot \frac{|\Theta^{\mathsf{ALL}}(l)|}{\varepsilon} &\leq k
\end{aligned}$$

Therefore, if there are at least $2^l \cdot \frac{|\Theta^{\mathsf{ALL}}(l)|}{\varepsilon}$ unsystematically generated plans between two systematically generated plans of length l, the total leakage will be less then ε and thus Algorithm 1 is ε-strong privacy preserving. □

[2]Provably refused plans are those that are guaranteed to be generated by systematic generation of plans.

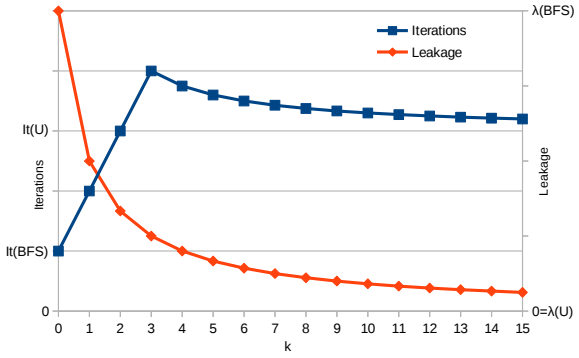

Figure 1: Trade-off between leaked knowledge λ(BFS/U) and number of iterations It(BFS/U) required to solve a problem with different numbers of unsystematically (U) generated plans inserted between two systematically generated plans (BFS).

The ε, and consequentially also k, acts as a trade-off parameter between security and efficiency. If α_2 generates all plans unsystematically, then no knowledge about not yet generated public plans could be deduced and thus it would imply the strong privacy. Nevertheless, this would breach the completeness of the algorithm `SecureMAPlanner()`, as only the systematically generated plans ensure complete search of all possible plans. This trade-off is illustrated by Figure 1 showing knowledge leakage λ(BFS/U) and number of iterations It(BFS/U) required to solve a problem for different values k. We can see, that in this particular case, this problem is solved by diluted BFS to up to three unsystematically generated plans inserted between plans generated by BFS. Then, unsystematical plan generator finds the solution faster than BFS, and from this point more unsystematically generated plans imply both increased efficiency and reduced knowledge leakage. Obviously, in different cases, both curves would cross at different k value.

In PSM planner (Tožička et al., 2015), each agent stores generated plans in a form of planning state machines, special version of finite state machines. (Guanciale et al., 2014) presents an algorithm for secure intersection of finite state machines which can be used for secure intersection of planning state machines too.

In the case of different representation of public plans, more general approach of generic secure computation can be applied (Ben-Or et al., 1988; Yao, 1982; Yao, 1986).

4 EXAMPLE

Let us consider a simple logistics scenario to demonstrate how private knowledge can leak for $k = 0$ and how it decreases with larger k values.

In this scenario, there are two transport vehicles (`plane` and `truck`) operating in three locations (`prague`, `brno`, and `ostrava`). A `plane` can travel from `prague` to `brno` and back, while a `truck` provides connection between `brno` and `ostrava`. The goal is to transport the `crown` from `prague` to `ostrava`.

This problem can be expressed using MA-STRIPS as follows. Actions $\texttt{fly}(\mathit{loc}_1, \mathit{loc}_2)$ and $\texttt{drive}(\mathit{loc}_1, \mathit{loc}_2)$ describe movement of `plane` and `truck` respectively. Actions $\texttt{load}(\mathit{veh}, \mathit{loc})$ and $\texttt{unload}(\mathit{veh}, \mathit{loc})$ describe loading and unloading of `crown` by a given vehicle at a given location.

We define two agents Plane and Truck. The agents are defined by sets of executable actions as follows.

$$\begin{aligned}
\mathit{Plane} = \{ & \\
& \texttt{fly(prague,brno)}, \texttt{fly(brno,prague)}, \\
& \texttt{load(plane,prague)}, \texttt{load(plane,brno)}, \\
& \texttt{unload(plane,prague)}, \texttt{unload(plane,brno)} \ \} \\
\mathit{Truck} = \{ & \\
& \texttt{drive(brno,ostrava)}, \texttt{drive(ostrava,brno)}, \\
& \texttt{load(truck,brno)}, \texttt{load(truck,ostrava)}, \\
& \texttt{unload(truck,brno)}, \texttt{unload(truck,ostrava)} \ \}
\end{aligned}$$

Aforementioned actions are defined using facts $\texttt{at}(\mathit{veh}, \mathit{loc})$ to describe possible vehicle locations, and facts $\texttt{in}(\texttt{crown}, \mathit{loc})$ and $\texttt{in}(\texttt{crown}, \mathit{veh})$ to describe positions of `crown`. We omit action ids in examples when no confusion can arise. For example, we have the following.

$$\begin{aligned}
\texttt{fly}(\mathit{loc}_1, \mathit{loc}_2) = \langle & \\
& \{\texttt{at}(\texttt{plane}, \mathit{loc}_1)\}, \\
& \{\texttt{at}(\texttt{plane}, \mathit{loc}_2)\}, \\
& \{\texttt{at}(\texttt{plane}, \mathit{loc}_1)\}\rangle \\
\texttt{load}(\mathit{veh}, \mathit{loc}) = \ \langle & \\
& \{\texttt{at}(\mathit{veh}, \mathit{loc}), \texttt{in}(\texttt{crown}, \mathit{loc})\}, \\
& \{\texttt{in}(\texttt{crown}, \mathit{veh})\}, \\
& \{\texttt{in}(\texttt{crown}, \mathit{loc})\}\rangle
\end{aligned}$$

The initial state and the goal are given as follows.

$$\begin{aligned}
I = \{ & \ \texttt{at(plane,prague)}, \texttt{at(truck,brno)}, \\
& \texttt{in(crown,prague)}\} \\
G = \{ & \ \texttt{in(crown,ostrava)}\}
\end{aligned}$$

In our running example, the only fact shared by the two agents is $\texttt{in}(\texttt{crown},\texttt{brno})$. As we require $G \subseteq \mathcal{V}^{\mathsf{pub}}$ we have the following facts classification.

$$\begin{aligned}
\mathcal{V}^{\mathsf{pub}} &= \{\texttt{in(crown,brno)}, \\
&\qquad \texttt{in(crown,ostrava)}\} \\
\mathcal{V}^{\mathsf{priv}}_{\mathit{Plane}} &= \{\texttt{at(plane,prague)}, \texttt{at(plane,brno)}, \\
&\qquad \texttt{in(crown,prague)}, \texttt{in(crown,plane)}\}
\end{aligned}$$

$$\texttt{load(truck,brno)}^{\triangleright} = \langle\{\texttt{in(crown,brno)}\},\emptyset,\{\texttt{in(crown,brno)}\}\rangle$$
$$\texttt{unload(truck,ostrava)}^{\triangleright} = \langle\emptyset,\{\texttt{in(crown,ostrava)}\},\emptyset\rangle$$

All the actions arranging vehicle movements are internal. Public are only the actions providing package treatment at public locations (brno, ostrava). Hence the set $O^{\mathsf{pub}}_{\texttt{plane}}$ contains only actions load(plane,brno) and unload(plane,brno) while O^{pub}_{Truck} is as follows.

{ load(truck,brno), unload(truck,brno), load(truck,ostrava), unload(truck,ostrava) }

Agent plane systematically generates possible plans using BFS and thus it sequentially generates following public plans:

$\pi_1^{\texttt{plane}} = \langle$ unload(truck,ostrava) $\rangle$
$\pi_2^{\texttt{plane}} = \langle$ unload(plane,brno), unload(truck,ostrava) $\rangle$
$\pi_3^{\texttt{plane}} = \langle$ unload(truck,brno), unload(truck,ostrava) $\rangle$
$\pi_4^{\texttt{plane}} = \langle$ unload(plane,brno), unload(truck,ostrava) $\rangle$
...
$\pi_i^{\texttt{plane}} = \langle$ unload(plane,brno), load(truck,brno), unload(truck,ostrava) $\rangle$

Note, that actually any locally valid sequence of action containing action unload(truck,ostrava) seems to be a valid solution to plane agent. In this example,$\pi_i^{\texttt{plane}}$ is the first extensible plan generated by plane and it is generated in i-th iteration of the algorithm.

Similarly, agent truck sequentially generates following public plans:

$\pi_1^{\texttt{truck}} = \langle$ unload(plane,brno), load(truck,brno), unload(truck,ostrava) $\rangle$
$\pi_2^{\texttt{truck}} = \langle$unload(plane,brno), unload(plane,brno), load(truck,brno), unload(truck,ostrava) $\rangle$
...

We can see that truck generates an extensible plan as the first one and plane generated equivalent solution at i-th iteration. Thus, once both agents agree on a solution, agent plane can try to deduce something about truck private knowledge. Since all plans $\pi_1^{\texttt{plane}},\ldots,\pi_4^{\texttt{plane}}$ are strictly shorter than the accepted solution $\pi_i^{\texttt{plane}}$ and there were not generated by truck, that implies that these plans are not acceptable by truck, i. e. for example $\phi^{\perp}(\pi_1^{\texttt{plane}}) = 0$. More specifically, plane can deduce following about truck's private knowledge:

- unload(truck,ostrava) has to contain some private precondition, otherwise $\pi_1^{\texttt{plane}}$ would be generated also by truck before $\pi_1^{\texttt{truck}}$ because it is shorter.
- Private precondition of unload(truck,ostrava) certainly depends on private fact (possibly indirectly) generated by load(truck,brno), otherwise $\pi_2^{\texttt{plane}}$ would be generated before $\pi_1^{\texttt{truck}}$.

In this example, we have shown how systematic generation of plans can cause private knowledge leakage. Let us now consider a case when both agents add one unsystematically generated plan after each systematically generated one, i. e. $k = 1$. For the simplicity, we will consider previous sequence of plans, where $\pi_i^{\texttt{plane}}$ is unsystematically generated after $\pi_3^{\texttt{plane}}$. In this case, the amount of leaked knowledge is much smaller. plane can still deduce that $\phi^{\perp}(\pi_1^{\texttt{plane}}) = 0$ but cannot deduce the same about other plans. plane could deduce that truck accepts no plan of length 2, only once it is sure that all of them have been systematically generated. But thanks to the adding of unsystematically generated plans, this would take twice as long and there is also some probability that the solution is found using unsystematically generated plans.

Obviously $k = 1$ decreases the leaked knowledge only minimally. To decrease the private knowledge leakage significantly, k has to grow exponentially with the length systematically generated solutions as we showed in proof of Theorem 2.

5 CONCLUSIONS

We have proposed a straightforward application of the private set intersection (PSI) algorithm to privacy preserving multiagent planning using intersection of sets of plans. As the plans are generated as extensible to a global solution provided that all agents agree on a selection of such local plans, the soundness of the planning approach is ensured. The intersection process can be secure in one iteration by PSI, but some private knowledge can leak during iterative generation of the local plans, which is the only practical way how to solve generally intractable planning problems. In more iterations, plans which are extensible by some agents but not extensible by all agents can leak private information about private dependencies of actions within the plans. In other words if an agent says the proposed solution can be from its perspective used as a solution to the planning problem, but it cannot be used as a solution by another agent, the first one learns, that the other one needs to use some private ac-

tions which obviate usage (extensibility) of the plan to a global solution. We have shown that this privacy leakage can be arbitrarily "diluted" by randomly generated local plans, however never fully averted provided the completeness of the planning process has to be ensured.

ACKNOWLEDGEMENTS

This research was supported by the Czech Science Foundation (no. 15-20433Y).

REFERENCES

Ben-Or, M., Goldwasser, S., and Wigderson, A. (1988). Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 1–10, New York, NY, USA. ACM.

Brafman, R. I. (2015). A privacy preserving algorithm for multi-agent planning and search. In Yang, Q. and Wooldridge, M., editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1530–1536. AAAI Press.

Brafman, R. I. and Domshlak, C. (2008). From one to many: Planning for loosely coupled multi-agent systems. In *Procs. of the ICAPS'08*, pages 28–35.

Diffie, W. and Hellman, M. (1976). New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654.

Guanciale, R., Gurov, D., and Laud, P. (2014). Private intersection of regular languages. In Miri, A., Hengartner, U., Huang, N., Jøsang, A., and García-Alfaro, J., editors, *2014 Twelfth Annual International Conference on Privacy, Security and Trust, Toronto, ON, Canada, July 23-24, 2014*, pages 112–120. IEEE.

Jarecki, S. and Liu, X. (2010). Fast secure computation of set intersection. In Garay, J. A. and Prisco, R. D., editors, *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*, volume 6280 of *Lecture Notes in Computer Science*, pages 418–435. Springer.

Maliah, S., Shani, G., and Brafman, R. (2016a). Online macro generation for privacy preserving planning. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling, ICAPS'16*.

Maliah, S., Shani, G., and Stern, R. (2016b). Collaborative privacy preserving multi-agent planning. *Procs. of the AAMAS'16*, pages 1–38.

Nissim, R. and Brafman, R. I. (2014). Distributed heuristic forward search for multi-agent planning. *JAIR*, 51:293–332.

Pinkas, B., Schneider, T., Segev, G., and Zohner, M. (2015). Phasing: Private set intersection using permutation-based hashing. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 515–530, Washington, D.C. USENIX Association.

Stolba, M., Tozicka, J., and Komenda, A. (2016). Secure multi-agent planning algorithms. In *ECAI 2016*, pages 1714–1715.

Torreño, A., Onaindia, E., and Sapena, O. (2014). FMAP: distributed cooperative multi-agent planning. *AI*, 41(2):606–626.

Tožička, J., Jakubův, J., Komenda, A., and Pěchouček, M. (2015). Privacy-concerned multiagent planning. *KAIS*, pages 1–38.

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

Yao, A. C. (1982). Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, Washington, DC, USA. IEEE Computer Society.

Yao, A. C.-C. (1986). How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 162–167, Washington, DC, USA. IEEE Computer Society.