ε-Strong Privacy Preserving Multi-agent Planning

Antonín Komenda(✉), Jan Tožička, and Michal Štolba

Department of Computer Science, Faculty of Electrical Engineering,
Czech Technical University in Prague,
Karlovo náměstí 13, 121 35 Prague, Czech Republic
{antonin.komenda,jan.tozicka,michal.stolba}@fel.cvut.cz

Abstract. Multi-agent planning can solve various sequential decision problems comprising multiple entities. In contrast to classical planning, the agents are interested in maintaining privacy while planning with each other. Therefore they have to reason about what information they can share. Although privacy is one of the crucial aspects of multi-agent planning, formal and algorithmic treatment of privacy is rather sparse in literature. No domain-independent strong privacy preserving multi-agent planner was proposed so far. Moreover, our recent results indicate that an efficient variant of such planner may not exist at all. Such strong privacy preserving planner would not allow to leak any private information during planning neither directly nor indirectly. Especially the indirect leakage is hard to assess as it can be based on any possible deduction principle from the non-private information along the planning process.

Here, we propose a refined version of a multi-agent planning principle, based on our previous work published as the conference version of this paper. The planning principle is designed so that it can get arbitrarily close to the general strong privacy preserving planning for the price of decreased planning efficiency. We have tighten the bounds on the privacy leakage and proved the strong privacy can be achieved by a finite number of additional plans, in contrast to the previous algorithm, where the number had to be infinite in general. We newly illustrate the principle on an additional synthetic planning problem, which shows the general privacy leakage upper bound. As in the previous variant of the algorithm, the strong privacy assurances are under computational tractability assumptions commonly used in secure computation research.

Keywords: Automated planning · Multi-agent systems · Privacy
Security

1 Introduction

Multi-agent planning deals with a problem of finding a coordinated sequence of actions for a set of entities (or agents), such that the actions are applicable from

J. van den Herik et al. (Eds.): ICAART 2017, LNAI 10839, pp. 137–156, 2018.
https://doi.org/10.1007/978-3-319-93581-2_8

a predefined initial state and transform the environment to a state where predefined goals are fulfilled. If the environment and the actions are deterministic (that is their outcome is unambiguously defined by the state they are applied in), the problem is a deterministic multi-agent planning problem [3]. Furthermore, 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 multi-agent 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 multi-agent planners.

A number of privacy preserving multi-agent planners have been proposed in recent years, such as MAFS [10], FMAP [15], MADLA [19], PSM [16] and GPPP [8]. Although all of the mentioned planners claim to be privacy-preserving, formal proofs of such claims do not exist. The privacy of MAFS is discussed in [10] and expanded upon in [2], proposing Secure-MAFS, a version of MAFS with stronger privacy guarantees. These guarantees are proven for a family of planning problems, but does not hold generally. The approach was recently generalized in the form of Macro-MAFS [7], however without strengthening the claims about privacy.

Here, we propose a parameterized variant of strong privacy preserving planning using the definition of privacy in [10]. We show how the two extremities of the parameter lead to strong privacy preserving, but inefficient planner; or weak privacy preserving, but efficient planner.

This article is a reworked and extended version of the paper [17]. In this version, we have reformulated most of the definitions and proofs to improve readability and conciseness of the arguments. We provide tighter bounds on privacy leakage and propose a novel example illustrating the principle of the proposed planner. Moreover, the novel example provides a ground for novel claim about privacy leakage in multi-agent planning in general.

2 Multi-agent Planning

The most common model for multi-agent planning is MA-Strips [3] and derived models (such as MA-MPT [10] 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 $\mathcal{M} = \{\Pi_i\}_{i=1}^{|\mathcal{A}|}$ is a set of agent problems. An agent problem of agent $\alpha_i \in \mathcal{A}$ is defined as

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

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 of the agent α_i containing only $\mathcal{V}^{\mathsf{pub}}$ and $\mathcal{V}_i^{\mathsf{priv}}$ variable. $s_\star$ 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. Similarly, as in [9], we require all goals to be public, as private goals can be transformed into a public equivalent [16], i.e. $\mathsf{vars}(s_\star) \subseteq \mathcal{V}^{\mathsf{pub}}$.

The set $\mathcal{O}_i$ of actions comprises 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. A public projection of an action removes all its private parts. The set $\mathcal{O}^{\mathsf{proj}}$ contain public projections of other agents' actions. $\mathcal{O}_i^{\mathsf{pub}}$, $\mathcal{O}_i^{\mathsf{priv}}$, and $\mathcal{O}^{\mathsf{proj}}$ 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 a 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. A public action can be defined using a mixture of public and private preconditions and effects. A private action can be defined only over the private variables. 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 $s^\triangleright$, 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 $s^\triangleright = s'^\triangleright$. The public projection of 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 action noop. The public projection of Π is $\Pi^\triangleright = \langle \mathcal{V}^{\mathsf{pub}}, \{a^\triangleright | a \in \mathcal{O}^{\mathsf{pub}}\}, s_I^\triangleright, s_\star^\triangleright \rangle$.

We define the solution to Π 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. A local plan contains actions only of one agent, public, private or projected. Note that the actions in $\mathcal{O}^{\mathsf{proj}}$ are assumed to be of the particular agent as well.

Such local plan π does not have to be the global solution to $\mathcal{M}$, as the actions of other agents ($\mathcal{O}^{\mathsf{proj}}$) are used only as public projections and are missing private preconditions and effects of other agents. The public projection of π is defined as $\pi^\triangleright = (a_1^\triangleright, ..., a_k^\triangleright)$ with the noop actions omitted.

From the global perspective of $\mathcal{M}$ a public plan $\pi^\triangleright = (a_1^\triangleright, ..., a_k^\triangleright)$ 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 $s_I^\triangleright$ and the resulting state satisfies $s_\star^\triangleright$. A public plan is α_i-extensible, if by replacing $a_{k'}^\triangleright$ s.t. $a_{k'} \in \mathcal{O}_i^{\mathsf{pub}}$ by the respective $a_{k'}$ and adding $a_{k''} \in \mathcal{O}^{\mathsf{priv}}$ to required places we obtain a local plan (solution) to Π_i. According to [16], a public plan $\pi^\triangleright$ α_i-extensible by all $\alpha_i \in \mathcal{A}$ is a global solution to $\mathcal{M}$.

The nature of MA-MPT planning allows for plans containing repeated action sequences (in extreme, repeated infinitely many times). Such repetitions however does not provide transformation to a state not yet visited. Therefore the length of global *meaningful* plans is bounded by the number of all possible states

$$\prod_{V \in \mathcal{V}^{\mathsf{pub}} \cup \bigcup_{i=1}^{|\mathcal{A}|} \mathcal{V}_i^{\mathsf{priv}}} |\mathsf{dom}(V)|. \tag{1}$$

We define the sets of all meaningful global and local plans as $\textsc{sols}(\mathcal{M})$ and $\textsc{sols}(\Pi)$ respectively. Lengths of plans in both sets are limited by the presented bound on meaningful plans and therefore the sets are finite.

Since an agent can be required to repeatedly produce a particular value assignment of a variable, which is consumed (needed in precondition and changed in effect) by an action of another agent, we have to allow for meaningful repetitions in $\textsc{sols}(\Pi)$. Length of a local plan with such repetitions is however still limited by the maximal length of its related global plan solving $\mathcal{M}$. Therefore we use the same bound on length both for plans in $\textsc{sols}(\mathcal{M})$ and $\textsc{sols}(\Pi)$ where $\Pi \in \mathcal{M}$. $\textsc{sols}_l(\Pi)$ will denote sets of local plans of length l; $\textsc{sols}_{\leq l}(\Pi)$ and $\textsc{sols}_{>l}(\Pi)$ denote sets of local plans of length no more than l and longer than l respectively. Finally, $\textsc{seqs}(\Pi)$ will denote all possible sequences of actions (incl. non-plans) of the problem Π.

2.1 Privacy

First definition of privacy leakage quantification was proposed in [18]. It was based on enumeration of all plans, which we used as the underlying principle for measuring the privacy leakage in this work. However, we do not explicitly require enumeration of the particular plans as in looser form, our bounds work with all possible action sequences. Note that the work in [18] is also not easily applicable to MA-Strips and MA-MPT.

The only rigorous definition of privacy for MA-Strips and MA-MPT so far was proposed in [10] and extended in [2]. The authors present two notions, weak and strong privacy preservation:

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 $\Pi^{\triangleright}$. 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 $\Pi^{\triangleright}$ and the public projection of the solution plan $\pi^{\triangleright}$.

2.2 Secure Computation

In general, any function can be computed securely [1,20,21], however it is not known how to encode the whole planning process into one function [14]. 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 information is transferred between the agents [11].

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., [6,11]), 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 [4], 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 ε-Strong Privacy Preserving Multi-agent Planner

Multi-agent planner fulfilling the strong privacy requirement forms the lower bound of information exchanged between the agents. Agents do not leak any information about their internal problems and thus their cooperation cannot be effective [14], nevertheless, a strong privacy preserving multi-agent planner is an important theoretical result that could lead to better understanding of privacy preservation during multi-agent 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 projected plan-space[1] search [13] and thus we will define the terms in that respect. In the following definitions and proofs we suppose that there are two semi-honest (honest but curious) agents α_- and α_+. We will consider the perspective of the agent α_- trying to detect the private information of α_+ for the simplicity of the presentation, but all holds for both agents and also for a larger group of agents. Similarly to [2], we also assume that $\mathcal{O}^{\mathsf{priv}} = \emptyset$. This assumption can be stated WLOG as each sequence of private actions followed by a public action can be compiled to a single public action.

Definition 1 (Public Plan Acceptance). Public plan acceptance $P(\pi^{\triangleright})$ *is a probability known to agent* α_- *whether a plan* $\pi^{\triangleright}$ *is* α_+*-extensible.*

When the algorithm starts, α_- has some *a priori* information $P^0(\pi^{\triangleright})$ about acceptance of plan $\pi^{\triangleright}$ by agent α_+ (e.g., 0.5 probability of acceptance of each

[1] Projected plan-space contains all the solutions of the projected public problem $\Pi^{\triangleright}$.

plan in the case when α_- knows nothing about α_+). At the end of execution of the algorithm, this information changes to $P^*(\pi^\triangleright)$. Obviously, every agent knows that the solution public plan π^* the agents agreed on is extensible and thus it is accepted by every agent, i.e. $P^*(\pi^*) = 1$. The difference between the α_-'s a priori information and the final information represents information which leaked from α_+ during their communication. Whether an agent is *certain* about acceptance of a plan can be expressed as $|1 - 2P(\pi^\triangleright)|$, normalized to an interval $\langle 0, 1\rangle$, where 0 means not knowing anything about acceptance of the plan ($P(\pi^\triangleright) = 0.5$) and 1 means certainty ($P(\pi^\triangleright) = 1$ or $P(\pi^\triangleright) = 0$).

Definition 2 (Leaked Information). Leaked information *from perspective of one agent during execution of a multi-agent planner leading to a solution* π^* *is a sum of changes in certainty about acceptance of the plan from the beginning* $P^0(\pi^\triangleright)$ *to the end* $P^*(\pi^\triangleright)$ *of planning excluding the solution plan* π^*

$$\lambda = \sum_{\pi^\triangleright \in \{\pi^\triangleright \mid \pi \in \text{SOLS}(\Pi)\} \setminus \{\pi^*\}} \left|1 - 2P^*(\pi^\triangleright)\right| - \left|1 - 2P^0(\pi^\triangleright)\right|. \tag{2}$$

As we do not assume the agents intentionally increase uncertainty about acceptance of other agents by sending invalid plans (the honest but curious agents), the certainty about acceptance of a plan can only grow, i.e.

$$\left|1 - 2P^*(\pi^\triangleright)\right| - \left|1 - 2P^0(\pi^\triangleright)\right| \geq 0, \text{ thus } \lambda \geq 0.$$

Definition of algorithm's *leaked information* allows us to formally define *strong privacy* of a projected plan-space planning algorithm. sec:strong-priv-pres

Proposition 1. (Strong Privacy). *A planning algorithm is* strong privacy preserving *if it assures* $\lambda = 0$.

Proof. Any information leakage allowing deduction of private information (preconditions or effects) in agent's planning problem during planning affect probability of acceptance or rejection of plan projections by other agents as the preconditions and effects are the only principle preventing of acceptance or rejection of a sequence of actions. Therefore $\lambda = 0$ *holds if and only if no information by Definition 2 leaked.*

Definition 3 (ε-Strong Privacy Preserving Planner). *For given* $\varepsilon > 0$*, an planning algorithm is* ε-strong privacy preserving *if it leaks acceptance or rejection of no more than* ε *local plans, i.e.* $\lambda \leq \varepsilon$.

The high-level idea of our proposed planner (Algorithm 1) is based on a systematic *generate-and-test* principle, similar to our recent principle proposed in [16]. Local plans π are generated in parallel by all agents and their public projections $\pi^\triangleright$ are distributively tested whether there are some projections π^* common to all agents. Since only acceptable local plans π thus public projections $\pi^\triangleright$ are generated and tested, if a public projection common to all agents is found, it is guaranteed to be a global solution [16]. Provided that the distributed

testing is done such that no information leaks, the only other point of possible information leakage is from the fact that a global solution was not found for a particular set of generated local plans. In other words, knowing α_+ refused all possible solutions in a well defined set of plans, tells α_- the plans were refused because of some private preconditions of α_+. Technically, the only parts of the algorithm, where the agents can learn something about each others' plans is therefore at lines 9 and 11, where all agents know that a solution was not found for all plans generated by the iterations of the algorithm so far.

Let us assume the systematicity of the generate-and-test principle is in testing of incrementally longer plans. The length l of the agents' local plans grows with each iteration of the main loop (lines 3, 4 and 13), therefore each distributed intersection (line 9) is done for generated plans of length $\leq l$. After each iteration, which did not end at line 11, the agent α_- knows that the agent α_+ refuses a local plan projection $\pi^\triangleright$ generated by α_-. This increases the certainty about refusal of $\pi^\triangleright$ and therefore increases λ. Note that such situation can be caused only by unfulfilled private preconditions of an action of the agent α_+ which prevent α_+ to generate $\pi^\triangleright$. This principle is known as privately dependent actions, for more detail see [12].

The principle of iterations synchronized by length was used only for the sake of clearer explanation. The argument however holds WLOG also for other iterative schemes. If the length of the generated plans is not synchronized by the iterations, all local plans of length l will be eventually generated by both agents α_+ and α_-. When a solution of length $l+1$ is found, α_- can use the same reasoning as in the previous paragraph to deduce α_+ has some unfulfilled private preconditions in $\pi^\triangleright$.

Generally, the same line of reasoning can be even used for any systematic plan generating algorithm, under the assumption all agents know the other agents' systematic plan generation algorithms. There always exists a point in future when α_- knows that α_+ had generated a plan, which would have be a solution of the problem and the algorithm would have end. And the only reason, why this had happened is that α_+ has some unfulfilled private preconditions.

To fulfill the ε-privacy requirement by the Algorithm 1, the systematically generated plans, which can leak information, are supplemented by a sufficiently large amount of randomly generated plans on line 7. As these plans are longer than the systematic ones, with a probability proportional to the number of such plans generated, they can shortcut finding of a solution sooner than using only the systematic plan generation. The formula $|\textsc{sols}_{>l}(\Pi_i)|(1-\frac{\varepsilon}{|\textsc{sols}_{\leq l}(\Pi_i)|})$ for the number of the shortcut plans k will be explained later as part of the ε-privacy proof.

In summary, all agents sequentially generate solutions to their local problems Π_i at line 5. The systematic local plans are supplemented by longer randomly generated local plans at line 7. Then the agents create public plans by making public projections $\pi^\triangleright$ of their generated solutions. Created public plans $\pi^\triangleright$ are then stored in a set Φ_i. As the plans π are local solutions, they are α_i-extensible. Agents continuously check whether there are some plans in the intersection of

these sets from all other agents. 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 α_i-extensible by all agents and thus form global solutions. If at least one global solution is found in Φ, the algorithm ends at line 11. The algorithm ends for all agents in the same iteration, as the secure intersection is done distributively by all agents for all agents. Therefore the termination condition at line 10 is evaluated by all agents equally.

Algorithm 1. ε-Strong privacy preserving multi-agent planner.

1 **Function** `SecureMAPlanner`(Π_i, ε) **is**
2 $\quad \Phi_i \leftarrow \emptyset$;
3 $\quad l \leftarrow 1$;
4 $\quad$ **loop**
5 $\quad\quad \mathcal{S} \leftarrow$ generate all local solutions to Π_i of length l;
$\quad\quad k \leftarrow |\text{SOLS}_{>l}(\Pi_i)|(1 - \frac{\varepsilon}{|\text{SOLS}_{\leq l}(\Pi_i)|})$; $\varepsilon \leftarrow \varepsilon - k$;
6 $\quad\quad \mathcal{S}' \leftarrow$ randomly select k solutions to Π_i of any length $> l$;
7 $\quad\quad \Phi_i \leftarrow \Phi_i \cup \{\pi^{\triangleright} | \pi \in \mathcal{S} \cup \mathcal{S}'\}$;
8 $\quad\quad \Phi \leftarrow$ secure $\left(\bigcap_{\alpha_j \in \mathcal{A}} \Phi_j \right)$;
9 $\quad\quad$ **if** $\Phi \neq \emptyset$ **then**
10 $\quad\quad\quad$ **return** Φ;
11 $\quad\quad$ **end**
12 $\quad\quad l \leftarrow l + 1$;
13 $\quad$ **end**
14 **end**

The description of the planning algorithm is followed by proofs of its soundness (a result of the algorithm is always a correct solution), completeness (if a planning problem has a solution, it is returned by the algorithm) and assurance on information leakage no more than required ε.

Theorem 1 (Soundness and Completeness). *Algorithm* `SecureMAPlanner` *is sound and complete, under the assumption that the systematic plan generation procedure (line 5) is complete.*

Proof. ***(Soundness)*** *Every public plan returned by the algorithm is α_i-extensible by every agent, and thus it can be extended by all agents to a valid global solution (Lemma 1 in [16]).*

(Completeness) *Since there is only finite number of different plans of length at most l, all plans are eventually (in finite time) added to the plan set Φ_i under the assumption that the underlying plan generation procedure of the local solutions (line 5) is complete. The longest possible solution is finite by Eq. (1), thus* `SecureMAPlanner`() *with systematic local planner ends in finite time when $\mathcal{M}$ has a solution.*

Theorem 2 (ε-Strong Privacy). *Algorithm* `SecureMAPlanner()` *is* ε*-strong privacy preserving when ideal PSI is used for the secure plan projection intersection (line 9).*

Proof. The only points in the algorithm, where the agents communicate is in the distributed intersection of the public projections (line 9) and implicitly in the synchronized termination (lines 10–12).

To ensure privacy of the first point, 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 [6,11]. No private information leaks within ideal PSI, therefore no private information leaks during the distributed intersection.

There can be, however, private information leakage, when the algorithm continues several iterations, i.e. the algorithm does not terminate (the second point). When the agent α_- *finds out that some set of plans is unacceptable by the agent* α_+ *(which is the only reason, why the algorithm has to continue with another iteration), private information leaks simply by growth of certainty by Definition 2.*

As α_+ *does not know how many plans* α_- *has generated thus how many plans were refused, if we want to upper-bound the possible leaked information,* α_+ *has to consider that all possible plans of length* l *were generated by* α_- *and refused by* α_+*, i.e.* $\lambda_{l+1} - \lambda_l \leq |\text{SOLS}_l(\Pi_i)|$*. Such situation reflects the maximal possible growth in the certainty about acceptance of possible plans. In sum over all plans lengths we get* $\lambda_l \leq \sum_{1 \leq l' \leq l} |\text{SOLS}_{l'}(\Pi_i)| = |\text{SOLS}_{\leq l}(\Pi_i)|$.

To limit the leakage by shortcutting the solution prematurely by the randomly selected plans, it has to happen that all agents generate by chance a global solution (line 7) sooner than systematically in its iteration by the length l*. As the best we can get is an upper-bound (the number of real solution is not known in beforehand) on the needed number of randomly generated plans* k*, we can assume that there is only one solution plan. A chance to randomly choose one particular plan by a selection of* k *random plans from all solutions* $|\text{SOLS}_{>l}| = n$ *longer than the current iteration length* l *is*

$$\frac{\binom{n}{k} - \binom{n-1}{k}}{\binom{n}{k}} = \frac{\binom{n-1}{k-1}}{\binom{n}{k}} = \frac{k}{n} = \frac{k}{|\text{SOLS}_{>l}(\Pi_i)|}. \tag{3}$$

The change of not selecting the solution plan is simply $1 - \frac{k}{|\text{SOLS}_{>l}(\Pi_i)|}$*, which with the upper-bound on the certainty about acceptance of possible plans gives us a parameterized tighter upper-bound on the leaked information* $\lambda_l \leq |\text{SOLS}_{\leq l}(\Pi_i)|(1 - \frac{k}{|\text{SOLS}_{>l}(\Pi_i)|}) \leq \text{SOLS}_{\leq l}(\Pi_i)$*. Note that each agent has the same chance to select the one common solution, therefore the chance is not decreased with increasing number of agents.*

By Definition 3, $\lambda \leq \varepsilon$ *has to hold for* ε*-privacy preserving planner, that means for the last iteration with a systematically found global plan* $\lambda_{|\pi^*|} \leq \varepsilon$*. As the length of the systematic solution plan* $|\pi^*|$ *is not know in beforehand, we have*

to heuristically estimate it. As $l \leq |\pi^*|$ *holds for all iterations of the algorithm, we can modify the formula and derive the upper-bound on the number of shortcut plans to fulfill* ε*:*

$$\lambda_l \leq \varepsilon, \tag{4}$$

$$|\text{SOLS}_{\leq l}(\Pi_i)|(1 - \frac{k}{|\text{SOLS}_{>l}(\Pi_i)|}) \leq \varepsilon, \tag{5}$$

$$|\text{SOLS}_{>l}(\Pi_i)|(1 - \frac{\varepsilon}{|\text{SOLS}_{\leq l}(\Pi_i)|}) \leq k. \tag{6}$$

As such k *is only an estimate assuming each iteration is the last one, we have to decrease the allowed leakage in each iteration by* $\varepsilon \leftarrow \varepsilon - k$ *at line 6.*

The parameter ε, and consequentially also k, acts as a trade-off parameter between security and efficiency. If the agent "randomly" selects all its plans $\text{SOLS}_{>l}(\Pi_i)$, then no information about refused plans can leak as it is assured that planning finds the (at least one existing) solution in the first iteration. Thus it would imply the strong privacy, i.e. for $k = |\text{SOLS}_{>l}(\Pi_i)|$ we get $\varepsilon \geq 0 \geq \lambda$ from Eq. 5 and Definition 3. Conversely, if we plan only systematically $k = 0$, the leakage upper-bounded $\varepsilon \geq |\text{SOLS}_{\leq l}(\Pi_i)| \geq \lambda$.

In the previous cases, $\text{SOLS}_{>l}$ can be replaced by $\text{SEQS}_{>l}$, as there cannot be less sequences than solutions and we are dealing with upper bounds. However, we kept the tighter $\text{SOLS}_{>l}$ in the proof and discussion. The possible issue with $|\text{SOLS}_{>l}|$ is that it can be hard to evaluate them efficiently, which is not problem with $\text{SEQS}_{>l}$. The drawback of $\text{SEQS}_{>l}$ is their exponentially larger amount, therefore exponential "looseness" of the bounds and a need for possibly exponentially larger k.

The leakage bounds are illustrated in Fig. 1 for an example planning problem. The problem has 2, 4, 8, 16 and 32 solutions (in the set $\text{SOLS}_{\leq l}(\Pi_i)$) for plan lengths l 1, 2, 3, 4 and 5 respectively. This gives us 30, 28, 24, 16 and 0 solutions in the set $\text{SOLS}_{>l}(\Pi_i)$, again for lengths $l = 1 <$, 2, 3, 4 and 5. The lines depict the upper-bound of the leakage λ for different numbers of shortcut plans k. For example, in the first iteration, only the two possibly refused plans can leak information, therefore even when $k = 0$, i.e. without any shortcut plans, $\lambda_1 = 2$. Conversely, to assure the planning process ends in the first iteration and does not leak any information, k has to equal to the rest of plans for $> l$, which is 30, where maximal leakage is ensured to be 0. Based on the changing numbers of already generated and still to be generated plans the ratio changes with the following iterations.

The points, where the upper-bounds equals to 0, represent numbers of shortcut plans needed to provide strong privacy (Proposition 1). The Fig. 2 depicts the numbers k of shortcut plans for the particular iterations of the example planning problem. Although the example shows only a small and synthetic planning problem, the principles and trends of the privacy bounds are general.

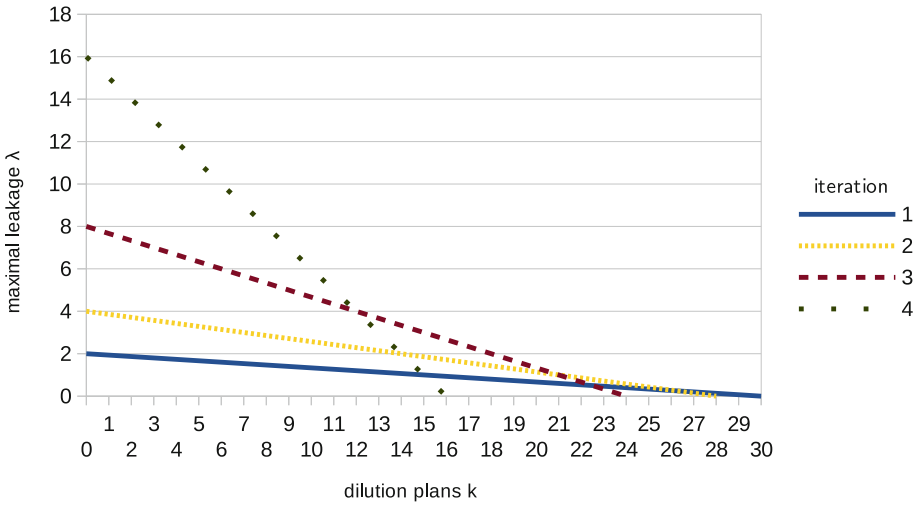

Fig. 1. Upper-bounds of required shortcut plans to assure leakage of the planning algorithm for iterations of the presented example planning problem.

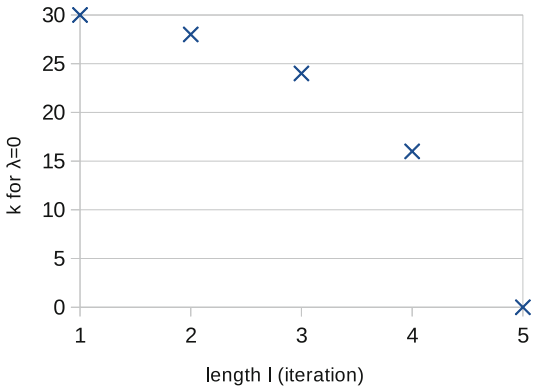

Fig. 2. Amounts of shortcut plans k assuring leakage of no private information, therefore privacy preserving run of the planning algorithm in the presented example planning problem.

To increase efficiency of the intersections, the principle proposed in PSM planner [16] can be used. Each agent stores generated plans in a form of planning state machines, special version of finite state machines. An algorithm, which can be used for secure intersection of planning state machines, was presented in [5]. In the case of different representation of public plans, more general approach of generic secure computation can be applied [1,20,21].

4 Logistics Example

Let us consider a simple logistics scenario to demonstrate how private information 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

$$\texttt{fly}(\mathit{loc}_1, \mathit{loc}_2) \text{ and } \texttt{drive}(\mathit{loc}_1, \mathit{loc}_2)$$

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

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

The aforementioned actions are defined using binary variables $\texttt{at}(\mathit{veh}, \mathit{loc}) \in \{\mathsf{true}, \mathsf{false}\}$ to describe possible vehicle locations and binary variables $\texttt{in}(\texttt{crown}, \mathit{loc})$ and $\texttt{in}(\texttt{crown}, \mathit{veh})$ to describe positions of `crown`. A variable is assigned true value if the fact it is describing holds. E.g. $\texttt{in}(\texttt{crown}, \texttt{plane}) = \mathsf{true}$ represents the fact that `crown` is in `plane`. We omit action names in examples when no confusion can arise. For example, we have the following actions:

$$\begin{aligned}
\texttt{fly}(\mathit{loc}_1, \mathit{loc}_2) = \langle \mathsf{pre}(.) &= \{\texttt{at}(\texttt{plane}, \mathit{loc}_1) = \mathsf{true}\},\\
\mathsf{eff}(.) &= \{\texttt{at}(\texttt{plane}, \mathit{loc}_2) \leftarrow \mathsf{true},\\
&\qquad \texttt{at}(\texttt{plane}, \mathit{loc}_1) \leftarrow \mathsf{false}\}\rangle,
\end{aligned}$$

$$\begin{aligned}
\texttt{load}(\mathit{veh}, \mathit{loc}) = \ \ \langle \mathsf{pre}(.) &= \{\texttt{at}(\mathit{veh}, \mathit{loc}) = \mathsf{true},\ \texttt{in}(\texttt{crown}, \mathit{loc}) = \mathsf{true}\},\\
\mathsf{eff}(.) &= \{\texttt{in}(\texttt{crown}, \mathit{veh}) \leftarrow \mathsf{true},\\
&\qquad \texttt{in}(\texttt{crown}, \mathit{loc}) \leftarrow \mathsf{false}\}\rangle,\\
\texttt{unload}(\mathit{veh}, \mathit{loc}) = \langle \mathsf{pre}(.) &= \{\texttt{at}(\mathit{veh}, \mathit{loc}) = \mathsf{true},\ \texttt{in}(\texttt{crown}, \mathit{veh}) = \mathsf{true}\},\\
\mathsf{eff}(.) &= \{\texttt{in}(\texttt{crown}, \mathit{loc}) \leftarrow \mathsf{true},\\
&\qquad \texttt{in}(\texttt{crown}, \mathit{veh}) \leftarrow \mathsf{false}\}\rangle.
\end{aligned}$$

The initial state and the goal are given as follows:

$$\begin{aligned} s_I &= \{\mathtt{at}(\mathtt{plane},\mathtt{prague}) = \mathsf{true},\ \mathtt{at}(\mathtt{truck},\mathtt{brno}) = \mathsf{true},\\ &\quad\ \mathtt{in}(\mathtt{crown},\mathtt{prague}) = \mathsf{true}\}\\ s_\star &= \{\mathtt{in}(\mathtt{crown},\mathtt{ostrava}) = \mathsf{true}\} \end{aligned}$$

All other variables not present in the initial state s_I are false.

In our example, the only variable shared by the two agents is $\mathtt{in}(\mathtt{crown},\mathtt{brno})$ and as required by $\mathsf{vars}(s_\star) \subseteq \mathcal{V}^{\mathsf{pub}}$ (see Sect. 2), the goal $\mathtt{in}(\mathtt{crown},\mathtt{ostrava})$. We have the following variable classification:

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

The actions and their projections important for the following discussion are:

$$\begin{aligned} \mathtt{load}(\mathtt{truck},\mathtt{brno}) &= \langle \mathsf{pre}(.) = \{\mathtt{in}(\mathtt{crown},\mathtt{brno}) = \mathsf{true},\\ &\qquad\qquad \mathtt{in}(\mathtt{truck},\mathtt{brno}) = \mathsf{true}\}\rangle,\\ &\qquad\quad \mathsf{eff}(.) = \{\mathtt{in}(\mathtt{crown},\mathtt{brno}) \leftarrow \mathsf{false},\\ &\qquad\qquad \mathtt{in}(\mathtt{crown},\mathtt{truck}) \leftarrow \mathsf{true}\},\\ \mathtt{load}(\mathtt{truck},\mathtt{brno})^{\triangleright} &= \langle \mathsf{pre}(.) = \{\mathtt{in}(\mathtt{crown},\mathtt{brno}) = \mathsf{true}\},\\ &\qquad\quad \mathsf{eff}(.) = \{\mathtt{in}(\mathtt{crown},\mathtt{brno}) \leftarrow \mathsf{false}\}\rangle, \end{aligned}$$

$$\begin{aligned} \mathtt{unload}(\mathtt{truck},\mathtt{ostrava}) &= \langle \mathsf{pre}(.) = \{\mathtt{in}(\mathtt{truck},\mathtt{ostrava}) = \mathsf{true},\\ &\qquad\qquad \mathtt{in}(\mathtt{crown},\mathtt{truck}) = \mathsf{true}\}\rangle,\\ &\qquad\quad \mathsf{eff}(.) = \{\mathtt{in}(\mathtt{crown},\mathtt{ostrava}) \leftarrow \mathsf{true},\\ &\qquad\qquad \mathtt{in}(\mathtt{crown},\mathtt{truck}) \leftarrow \mathsf{false}\},\\ \mathtt{unload}(\mathtt{truck},\mathtt{ostrava})^{\triangleright} &= \langle \mathsf{pre}(.) = \emptyset,\\ &\qquad\quad \mathsf{eff}(.) = \{\mathtt{in}(\mathtt{crown},\mathtt{ostrava}) \leftarrow \mathsf{true}\ \}\rangle \end{aligned}$$

All the actions arranging vehicle movements are private. Only the actions providing package treatment at public locations (`brno`, `ostrava`) are public:

$$\begin{aligned} O^{\mathsf{pub}}_{\mathit{Truck}} &= \{\ \mathtt{load}(\mathtt{truck},\mathtt{brno}), \mathtt{unload}(\mathtt{truck},\mathtt{brno}),\\ &\quad\ \mathtt{load}(\mathtt{truck},\mathtt{ostrava}), \mathtt{unload}(\mathtt{truck},\mathtt{ostrava})\ \},\\ O^{\mathsf{pub}}_{\mathit{Plane}} &= \{\mathtt{load}(\mathtt{plane},\mathtt{brno}), \mathtt{unload}(\mathtt{plane},\mathtt{brno})\}. \end{aligned}$$

The agent ***Plane*** generates possible plans using the systematic plan generation algorithm (e.g. Best-First Search) and thus it sequentially generates following public plans:

$$
\begin{aligned}
\pi_1^{Plane} &= \langle\ \mathtt{unload(truck, ostrava)}\ \rangle, l = 1\\
\pi_2^{Plane} &= \langle\ \mathtt{unload(plane, brno)},\\
&\qquad \mathtt{unload(truck, ostrava)}\ \rangle, l = 2,\\
\pi_3^{Plane} &= \langle\ \mathtt{unload(truck, brno)},\\
&\qquad \mathtt{unload(truck, ostrava)}\ \rangle, l = 2\\
\pi_4^{Plane} &= \langle\ \mathtt{unload(truck, ostrava)},\\
&\qquad \mathtt{unload(truck, ostrava)}\ \rangle, l = 2\\
&\dots\\
\pi_n^{Plane} &= \langle\ \mathtt{unload(plane, brno)},\ \mathtt{load(truck, brno)},\\
&\qquad \mathtt{unload(truck, ostrava)}\ \rangle, l = 3.
\end{aligned}
$$

Note that any locally valid sequence of action containing action

$$\mathtt{unload(truck, ostrava)}$$

seems to be a valid solution to the *Plane* agent. In this example, π_n^{Plane} is the first plan extensible to a global solution by both *Plane* and *Truck* generated by the systematic planning process.

Similarly, agent *Truck* sequentially generates following public plans:

$$
\begin{aligned}
\pi_1^{Truck} &= \langle\ \mathtt{unload(plane, brno)},\ \mathtt{load(truck, brno)},\\
&\qquad \mathtt{unload(truck, ostrava)}\ \rangle, l = 3,\\
\pi_2^{Truck} &= \langle\ \mathtt{unload(plane, brno)},\ \mathtt{unload(plane, brno)},\\
&\qquad \mathtt{load(truck, brno)},\ \mathtt{unload(truck, ostrava)}\ \rangle, l = 4,\\
&\dots
\end{aligned}
$$

We can see that *Truck* generates an extensible plan as the first one and *Plane* generated equivalent solution as the n-th plan. Thus, once both agents agree on a solution, agent *Plane* can try to deduce something about *Truck* private information. Since all plans $\pi_1^{\mathtt{plane}}, \dots, \pi_4^{\mathtt{plane}}$ are strictly shorter than the accepted solution $\pi_n^{\mathtt{plane}}$ and they were not generated by *Truck*, it implies that these plans are not acceptable by *Truck*, i. e. for example $P^*(\pi_1^{Plane}) = 0$. More specifically, *Plane* can deduce following about *Truck*'s private information:

- The action $\mathtt{unload(truck, ostrava)}$ has to contain some private precondition, otherwise π_1^{Plane} would be generated also by *Truck* before π_1^{Truck}, because it is shorter.
- Private preconditions of $\mathtt{unload(truck, ostrava)}$ certainly depend on private fact (possibly indirectly) generated by $\mathtt{load(truck, brno)}$, otherwise π_2^{Plane} would be generated before π_1^{Truck}.

In this example, we have shown how systematic generation of plans can cause private information leakage. Let us now consider a case when both agents add one shortcut plan after each systematically generated one, i. e. $k = 1$. For the simplicity, we will consider previous sequence of plans, where π_n^{Plane} is selected as the shortcut plan in the first iteration for $l = 1$ by both agents. In such case, the amount of leaked information is smaller by Eq. (5). If there is only one solution π_n^{Plane}, the leakage will be 0 and the agents would not be able to deduce

any private information about the other agents. If the shortcut plan π_n^{Plane} is added in next iteration for length $l = 3$, *Plane* can deduce that $P^*(\pi_1^{Plane}) = 0$, 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 the shortcut plan, the solution can be found sooner.

Obviously $k = 1$ decreases the leaked information only minimally. To decrease the private information leakage significantly, k has to grow by Eq. (5) towards $|\text{SOLS}_{>l}(\Pi_{Plane})|$ as we showed in proof of Theorem 2.

5 Code Lock Example and General Privacy Leakage Upper-Bound

The other example is designed such that it shows the successive leaking of information by learning about refused sequences of the actions. There is a combination code lock and two agents. The agent α_- is unlocking the lock with help of the other agent α_+, which knows the combination. The lock requires a correct sequence of n pressed buttons reachable by α_+ (each button can be pressed only once), finished with pressing two `unlock` buttons, each reachable only by one of the agents. Note that strictly speaking if only one combination is correct no private information would leak as Definition 2 excludes the solution plan π^*. We could modify the example such that there are more correct solution and α_- is trying to deduce all of them, however to simplify the latter discussion, we will stick to one solution, which we assume to be secret.

This assumption is not unrealistic, as in reality the press actions would be private, however by the requirement of privacy preserving MA-MPT planning, all actions incl. `press` are public.

The binary variables of the problem are

$$\begin{aligned}\mathcal{V}^{\text{pub}} &= \{\texttt{unlocked}_+, \texttt{unlocked}_-\},\\ \mathcal{V}^{\text{priv}}_{\alpha_+} &= \{\texttt{pressed}_0, \texttt{pressed}_1, \ldots, \texttt{pressed}_n\},\\ \mathcal{V}^{\text{priv}}_{\alpha_-} &= \emptyset.\end{aligned}$$

The two public `unlocked` variables describe whether the two agents successfully unlocked their side of the lock. For simplicity, we assume only α_+ need to enter the correct sequence, which allows to unlock its side. The agent α_- attempts to deduce the constraints among the presses during the process. Provided that the α_- agent has similar combination as α_+ the example would work symmetrically for both agents, or even for more agents unlocking the lock in coordination. The successfully entered steps of the combination are represented by the private `pressed` variables.

In the initial state, all variables are set to false with the exception of $\texttt{pressed}_0$ allowing to press the first correct button. The goal of the problem is to unlock both sides of the lock:

$$\begin{aligned}s_I &= \{\texttt{pressed}_0 = \mathsf{true}\},\\ s_\star &= \{\texttt{unlocked}_+ = \mathsf{true}, \texttt{unlocked}_- = \mathsf{true}\}.\end{aligned}$$

The actions of the problem are the two $\texttt{unlock}$ing the lock and actions representing pressing the buttons. All actions are public (following the assumption on no private actions), however actions of α_+ have private preconditions and effects constraining only the correct unlock sequence. The action sets consist of:

$$O_{\alpha_+}^{\mathsf{pub}} = \{\texttt{unlock}_+, \texttt{press}_1, \texttt{press}_2, \ldots, \texttt{press}_n\},$$
$$O_{\alpha_-}^{\mathsf{pub}} = \{\texttt{unlock}_-\}.$$

The action $\texttt{unlock}_-$ (and its projection) has no preconditions and the sole effect $\texttt{unlocked}_- \leftarrow \mathsf{true}$, which is required by the goal:

$$\texttt{unlock}_- = \texttt{unlock}_-^{\triangleright} = \langle \mathsf{pre}(.) = \emptyset, \mathsf{eff}(.) = \{\texttt{unlocked}_+ \leftarrow \mathsf{true}\}\rangle.$$

Let the unlocking sequence of button indices be described by a mapping $u(i) \mapsto i'$ which for each step i returns next button index i' to be pressed. Then each button pressing actions is defined as:

$$\begin{aligned} \texttt{press}_i = \langle \mathsf{pre}(.) &= \{\texttt{pressed}_{u(i)-1} = \mathsf{true}\}, \\ \mathsf{eff}(.) &= \{\texttt{pressed}_{u(i)} \leftarrow \mathsf{true}\}\rangle. \\ \texttt{unlock}_+ = \langle \mathsf{pre}(.) &= \{\texttt{pressed}_n = \mathsf{true}\}, \\ \mathsf{eff}(.) &= \{\texttt{unlocked}_+ \leftarrow \mathsf{true}\}\rangle. \end{aligned}$$

For an example, a $n = 3$ step unlocking sequence $2, 3, 1$ will induce following actions for the agent α_+:

$$\begin{aligned} \texttt{press}_2 = \langle \mathsf{pre}(.) &= \{\texttt{pressed}_0 = \mathsf{true}\}, \\ \mathsf{eff}(.) &= \{\texttt{pressed}_1 \leftarrow \mathsf{true}\}\rangle, \\ \texttt{press}_3 = \langle \mathsf{pre}(.) &= \{\texttt{pressed}_1 = \mathsf{true}\}, \\ \mathsf{eff}(.) &= \{\texttt{pressed}_2 \leftarrow \mathsf{true}\}\rangle, \\ \texttt{press}_1 = \langle \mathsf{pre}(.) &= \{\texttt{pressed}_2 = \mathsf{true}\}, \\ \mathsf{eff}(.) &= \{\texttt{pressed}_3 \leftarrow \mathsf{true}\}\rangle. \\ \texttt{unlock}_+ = \langle \mathsf{pre}(.) &= \{\texttt{pressed}_3 = \mathsf{true}\}, \\ \mathsf{eff}(.) &= \{\texttt{unlocked}_+ \leftarrow \mathsf{true}\}\rangle. \end{aligned}$$

The projections of all $\texttt{press}$ actions have no preconditions and effects (they are effectively noops with different action names from perspective of α_-), as the $\texttt{pressed}$ variables are private. The action $\texttt{unlock}$ has only its goal effect:

$$\begin{aligned} \texttt{press}_1^{\triangleright}, \ldots, \texttt{press}_n^{\triangleright} &= \langle \mathsf{pre}(.) = \emptyset, \mathsf{eff}(.) = \emptyset\rangle = \mathsf{noop}, \\ \texttt{unlock}_+^{\triangleright} &= \langle \mathsf{pre}(.) = \emptyset, \mathsf{eff}(.) = \{\texttt{unlocked}_+ \leftarrow \mathsf{true}\}\rangle. \end{aligned}$$

From perspective of α_-, any sequence of α_+ ending with $\texttt{unlock}_+$ is legitimate. On the contrary, only the correct sequence of $\texttt{press}$ actions and $\texttt{unlock}_+$ is valid from perspective of α_+.

In each iteration l, if a solution is not found, α_- eliminates all possible combinations of the code of length l. These eliminations represent the private variables $\texttt{pressed}$ in form of preconditions and effects of actions of α_+. Because of the

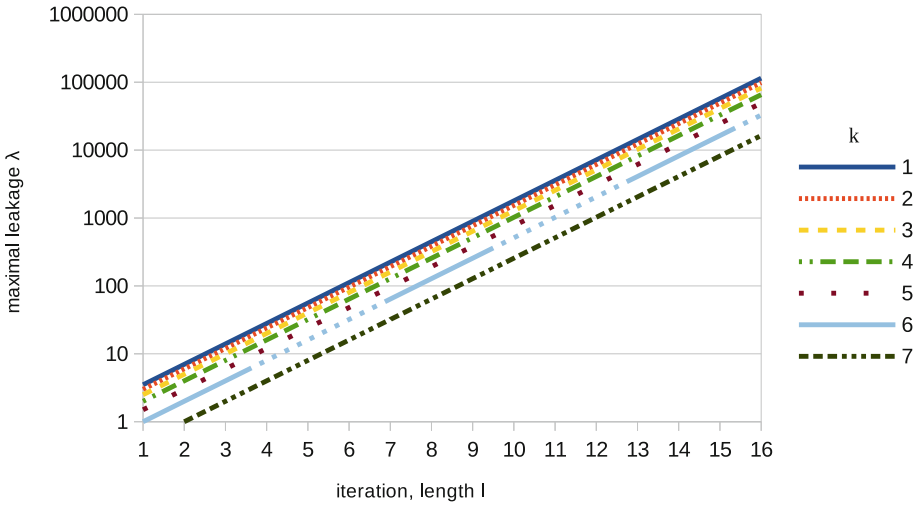

Fig. 3. Hypothetical upper-bounds on information leakage in the Code Lock example problem for $n = 2$ without decreasing ε during the iterations by $\varepsilon \leftarrow \varepsilon - k$. For $k = 8$ the leakage $\lambda = 0$. For $l \geq 3$, the leakage assumes possible longer plans of the problem (e.g., by more then two agents) than those limited by $l \leq n$.

problem formulation, the number of projected local plans of α_+ from perspective of α_- are:

$$n^1 + n^2 + \cdots + n^l + n^{l+1} + \cdots + n^{n-1} + n^n,$$

where n is the number of `press` actions and l is the intermediate length of the solution. We can bound sizes of $\text{SOLS}_{\leq l}$ and $\text{SOLS}_{>l}$ using this sequence, $l \leq n$ by the count of meaningful plans by Eq. 1 and an assumption on repetitions of the `press` actions are allowed as α_- cannot know otherwise:

$$|\text{SOLS}_{\leq l}| \leq \sum_{i=1}^{l} n^i \leq n^{l+1}, \tag{7}$$

$$|\text{SOLS}_{>l}| \leq \sum_{i=1}^{n-l} n^{l+i} \leq n^{n+1}. \tag{8}$$

Without the shortcut plans $k = 0$, we get that $n^{l+1} \leq \varepsilon$ using Eqs. (5) and (7). This shows the possible maximal leakage of information is exponentially bound by the length of the plan which is bound by the number of actions in Π_+. It is not surprising though that if we want a strong privacy preserving variant $\varepsilon = 0$, we need to generate exponential number of plans n^{n+1} in the number of actions of the problem for the first iteration by Eqs. (6) and (8).

The exponential growth of information leakage in l in this example is illustrated in Fig. 3. The other exponential dependency is on the number of (`press`) actions, i.e. on n.

As the Code Lock problem is designed such that there are no public dependencies among the actions (with exception of the required goal conditions), it represents a planning problem with the maximal amount of private information and only one solution as assumed in the proof of Theorem 2. Therefore the resulting exponential bounds on leakage in l and in n hold not only for this particular planning problem, but as a general upper bound on the privacy leakage in MA-Strips planning with n actions and k shortcut plans in l-th iteration of the `SecureMAPlanner()` algorithm (by Eqs. (5), (7) and (8)):

$$\lambda_l \leq n^{l+1}(1 - \frac{k}{n^{n+1}}). \tag{9}$$

6 Conclusions

In this article, we have provided a refined variant of the multi-agent planning principle preserving privacy from our previous version of the paper [17]. The principle of the algorithm is an application of the private set intersection (PSI) algorithm to privacy preserving multi-agent 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. As we showed in [17] and in the refined proof of Theorem 2 here, the intersection process can be secure in one iteration by PSI, but some private information 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 actions which obviate usage (extensibility) of the plan to a global solution. In the previous version of the algorithm, we have proposed to dilute the plans by an sufficient amount of randomized plans, however the number in general needed to be infinite [17]. In this variant of the algorithm, we have shown that the privacy leakage can be arbitrarily shortcut by randomly selected local plans and fully prevented by using all solutions (exponential in the number of actions) already in the first iteration. Although the number of such shortcut plans achieving strong privacy is exponential in general, in contrast to the dilution approach, the number is finite. The results are also in agreement with our recent results in [14]. As in the previous version of the paper, we have illustrated the principle on the logistics example, however with new results using the improved version of the algorithm. Newly, we have demonstrated the principle on a synthetic planning problem, which shows also the novel privacy leakage upper bound (Eq. 9) which holds in general.

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

References

1. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, STOC 1988, pp. 1–10. ACM, New York (1988)
2. Brafman, R.I.: A privacy preserving algorithm for multi-agent planning and search. In: Yang, Q., Wooldridge, M. (eds.) Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, 25–31 July 2015, pp. 1530–1536. AAAI Press (2015)
3. Brafman, R.I., Domshlak, C.: From one to many: planning for loosely coupled multi-agent systems. In: Proceedings of the ICAPS 2008, pp. 28–35 (2008)
4. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theor. **22**(6), 644–654 (1976)
5. Guanciale, R., Gurov, D., Laud, P.: Private intersection of regular languages. In: Miri, A., Hengartner, U., Huang, N., Jøsang, A., García-Alfaro, J. (eds.) 2014 Twelfth Annual International Conference on Privacy, Security and Trust, Toronto, ON, Canada, 23–24 July 2014, pp. 112–120. IEEE (2014). https://doi.org/10.1109/PST.2014.6890930
6. Jarecki, S., Liu, X.: Fast secure computation of set intersection. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 418–435. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15317-4_26
7. Maliah, S., Shani, G., Brafman, R.: Online macro generation for privacy preserving planning. In: Proceedings of the 26th International Conference on Automated Planning and Scheduling, ICAPS 2016 (2016)
8. Maliah, S., Shani, G., Stern, R.: Collaborative privacy preserving multi-agent planning. In: Proceedings of the AAMAS 2016, pp. 1–38 (2016)
9. Nissim, R., Brafman, R.I.: Multi-agent A* for parallel and distributed systems. In: Proceedings of AAMAS 2012, pp. 1265–1266 (2012)
10. Nissim, R., Brafman, R.I.: Distributed heuristic forward search for multi-agent planning. JAIR **51**, 293–332 (2014)
11. Pinkas, B., Schneider, T., Segev, G., Zohner, M.: Phasing: Private set intersection using permutation-based hashing. In: 24th USENIX Security Symposium (USENIX Security 15), pp. 515–530. USENIX Association, Washington, D.C. (2015)
12. Štolba, M., Tožička, J., Komenda, A.: Secure multi-agent planning. In: Proceedings of the International Workshop on PrAISe (2016)
13. Stolba, M., Tozicka, J., Komenda, A.: Secure multi-agent planning algorithms. ECAI **2016**, 1714–1715 (2016)
14. Stolba, M., Tozicka, J., Komenda, A.: The limits of strong privacy preserving multi-agent planning. In: Proceedings of the 27th International Conference on Automated Planning and Scheduling. ICAPS 2017 (2017). To appear
15. Torreño, A., Onaindia, E., Sapena, O.: FMAP: distributed cooperative multi-agent planning. AI **41**(2), 606–626 (2014)
16. Tožička, J., Jakubův, J., Komenda, A., Pěchouček, M.: Privacy-concerned multiagent planning. KAIS, pp. 1–38 (2015)
17. Tozicka, J., Komenda, A., Stolba, M.: ε-strong privacy preserving multiagent planner by computational tractability. In: van den Herik, H.J., Rocha, A.P., Filipe, J. (eds.) Proceedings of the 9th International Conference on Agents and Artificial Intelligence. ICAART 2017, vol. 1, Porto, Portugal, 24–26 February 2017, pp. 51–57. SciTePress (2017). https://doi.org/10.5220/0006176400510057

18. Van Der Krogt, R.: Quantifying privacy in multiagent planning. Multiagent Grid Syst. **5**(4), 451–469 (2009)
19. Štolba, M., Komenda, A.: Relaxation heuristics for multiagent planning. In: 24th International Conference on Automated Planning and Scheduling (ICAPS), pp. 298–306 (2014)
20. Yao, A.C.: Protocols for secure computations. In: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, pp. 160–164. SFCS 1982. IEEE Computer Society, Washington, DC (1982)
21. Yao, A.C.C.: How to generate and exchange secrets. In: Proceedings of the 27th Annual Symposium on Foundations of Computer Science, SFCS 1986, pp. 162–167. IEEE Computer Society, Washington, DC (1986)