Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

A Privacy Preserving Algorithm for Multi-Agent Planning and Search

Ronen I. Brafman
Department of Computer Science
Ben-Gurion University of the Negev
Be'er Sheva, Israel
brafman@cs.bgu.ac.il

Abstract

To engage diverse agents in cooperative behavior, it is important, even necessary, to provide algorithms that do not reveal information that is private or proprietary. A number of recent planning algorithms enable agents to plan together for shared goals without disclosing information about their private state and actions. But these algorithms lack clear and formal privacy guarantees: the fact that they do not require agents to *explicitly* reveal private information, does not imply that such information cannot be deduced. The main contribution of this paper is an enhanced version of the distributed forward-search planning framework of Nissim and Brafman that reveals less information than the original algorithm, and the first, to our knowledge, discussion and formal proof of privacy guarantees for distributed planning and search algorithms.

Introduction

As our world becomes better connected and more open ended, and autonomous agents are no longer science fiction, a need arises for enabling groups of agents to cooperate in generating a plan for diverse tasks that none of them can perform alone in a cost-effective manner. Indeed, much like ad-hoc networks, one would expect various contexts to naturally lead to the emergence of ad-hoc teams of agents that can benefit from cooperation. Such teams could range from groups of manufacturers teaming up to build a product that none of them can build on their own, to groups of robots sent by different agencies or countries to help in disaster settings. To perform complex tasks, these agents need to combine their diverse skills effectively. Planning algorithms can help achieve this goal.

Most planning algorithms require full information about the set of actions and state variables in the domain. However, often, various aspects of this information are private to an agent, and it is not eager to share them. For example, the manufacturer is eager to let everyone know that it can supply motherboards, but it will not want to disclose the local process used to construct them, or its inventory levels. Similarly, rescue forces of country A may be eager to help citizens of country B suffering from a tsunami, but without having to provide detailed information about the technology behind their autonomous bobcat to country B, or to country C's humanoid evacuation robots. In both cases, agents have public capabilities they are happy to share, and private processes and information that support these capabilities, which they prefer (or possibly require) to be kept private.

With this motivation in mind, a number of algorithms have recently been devised for distributed privacy-preserving planning [Bonisoli *et al.*, 2014; Torreño *et al.*, 2014; Luis and Borrajo, 2014; Nissim and Brafman, 2014]. In these algorithms, agents supply a public interface only, and through a distributed planning process, come up with a plan that achieves the desired goal without being required to share a complete model of their actions and local state with other agents. But there is a major caveat: it is well known from the literature on secure multi-party computation [Yao, 1982] that the fact that a distributed algorithm does not require an agent to *explicitly* reveal private information does not imply that other agents cannot deduce such private information from other information communicated during the run of the algorithm. Consequently, given that privacy is the raison-d'etre for these algorithms, it is important to strive to improve the level of privacy provided, and to provide formal guarantees of such privacy properties. To date, no such guarantees have been provided.

In this paper we focus on the multi-agent forward search (MAFS) algorithm [Nissim and Brafman, 2014]. Forward search algorithms are the state-of-the-art in centralized classical planning, and their distributed version scales up best among distributed algorithms for classical multi-agent planning. We describe a modified version of MAFS, called SECURE-MAFS in which less information is exposed to other agents, and we provide the first formal treatment of privacy in the context of distributed planning algorithms. SECURE-MAFS is not guaranteed to provide complete privacy always; indeed, we doubt that search based algorithm can provide such guarantees without fundamental new techniques because of the fact that information about intermediate states is being constantly exchanged. Yet, we are able to provide a sufficient condition for privacy, and use it to prove a number of results, placing the discussion of privacy in multi-agent planning on much firmer ground.

The Model

We use a multi-valued variable variant of the MA-STRIPS model [Brafman and Domshlak, 2008]. This framework min-

imally extends the classical STRIPS language to MA planning for cooperative agents. The main benefits of this model are its simplicity, and the fact that it is easily extended to handle the case of *non-cooperative* agents [Nissim and Brafman, 2013]. There is abundant work on multi-agent planning that uses setting much richer than the classical one (e.g., [Durfee, 2001; Bernstein *et al.*, 2005; Jonsson and Rovatsos, 2011; ter Mors *et al.*, 2010] to name a few). We focus on MA-STRIPS because it is so basic – a sort of "classical planning" for multi-agent system. Most planning models that use a factored state space extend it in various ways, and it would be easier to import the ideas and techniques for distribution and privacy preservation from it to such models. Moreover, the techniques we use are not specific to planning, but can be used for other applications that require distributed, privacy preserving search.

Definition 1. *A MA-STRIPS planning task for a set of agents* $\Phi = \{\varphi_i\}_{i=1}^k$ *is given by a 4-tuple* $\Pi = (V, \{A\}_{i=1}^k, I, G)$ *with the following components:*

- V *is a finite set of finite-domain variables,*
- *For* $1 \leq i \leq k$, A_i *is the set of actions that* φ_i *can perform. Each action* $a = \langle pre,eff,c \rangle \in A = \cup A_i$ *is given by its preconditions, effects, and cost, where preconditions and effects are partial assignments to* V.

A state s is a complete assignment to all variables in V. An action a is *applicable* in state s if a's precondition is consistent with s. The result of applying a in s, denoted $a(s)$ is defined for states s in which a is applicable. $a(s)$ is identical to the effect of a, where defined, and is identical to s, elsewhere. A *solution* to a planning task is a plan $\pi = (a_1, \ldots, a_k)$ such that $G \subseteq a_k(\ldots(a_1(s)\ldots)$. That is, it is a sequence of actions that transforms the initial state into a state satisfying the goal conditions. A solution is *cost-optimal* if it has minimal total cost over all possible solutions.

As an example, consider the well known Logistics classical planning domain, in which packages should be moved from their initial to their target locations. The agents are vehicles: trucks and airplanes that can transport these packages. The packages can be loaded onto and unloaded off the vehicles, and each vehicle can move between a certain subset of locations. Variables denoting possible locations are associated with each package and vehicle. Possible actions are *drive, fly*, *load*, and *unload*, each with its suitable parameters (e.g., *drive(truck, origin, destination)* and *load(package, truck, location)*). The actions associated with each particular agent are ones of loading and unloading from the corresponding vehicle, and moving it around its possible locations. MA-STRIPS specifies this association of actions to agents explicitly as part of its problem description.

Privacy

Privacy guarantees in MA planning come in the form of *private variables, values,* and *private actions*. If a value is private to an agent, then (ideally) only it knows about its existence. If an action is private to an agent, (ideally) only it knows about its existence, its form, and its cost. In this paper we shall use the deduced notion of private variables and private actions described below, following [Brafman and Domshlak, 2008]. Alternatively, one can specify privacy requirements as part of the input. In that case, if privacy is defined over variable values, our results, with the exception of completeness (which can be affected by the choice of private variables) carry through.

A *private* variable-value of agent φ is a value required and affected *only* by the actions of φ. A *private* variable is one, all of whose values are private. We refer to all other variables and variable-values as *public*. The *private state* of agent φ_i in state s, denoted by $s_{pr\text{-}i}$, consists of the private variable-value pairs that hold in s. An action is *private* if all its preconditions and effects are private. All other actions are classified as *public*. That is, φ's private actions affect and are affected only by the actions of φ, while its public actions may require or affect the actions of other agents. Note, however, that the public actions of an agent may have private preconditions and effects. Thus, the *public projection* of an action a is an action that contains only the public preconditions and effects of a. For ease of presentation, we assume that all actions that achieve a goal condition are *public*. Our methods are easily modified to remove this assumption.

In Logistics, all vehicle locations are private (affect and are affected only by their respective agent's *move* actions). Thus, the location of a vehicle is a private variable. *move* actions are also private, as they depend on and affect only the private location values. Package location values can be either private or public, depending on whether they are required/affected by more than one agent. Thus, *at(package)=location* is private if *location* is accessible only to one agent and public otherwise. Since, typically, some package locations are public (e.g., the airport), package location is not a private variable, although some of its values are private. Therefore, *load* (respectively *unload*) actions that require (respectively affect) public package location propositions are public.

We can now talk about the privacy guarantees of an algorithm. We say that an algorithm is *weakly private* if no agent communicates a private value of a variable (in the initial state, the goal, or any other intermediate state) to an agent to whom this value is not private during a run of the algorithm, and if the only description of its own actions it needs to communicate to another agent φ is their public projection.

Private values of one agent do not appear in preconditions or effects of other agents' actions. Thus, they do not influence their applicability. Hence, weak privacy is easily achieved by encrypting the private values of a variable. However, agents may deduce the existence and properties of private variables, values, and action preconditions, from information communicated during the run of the algorithm. For example, *at(package)=truck* is a value private to the truck because it appears only in the preconditions and effects of its load and unload actions. However, suppose a package is in location A at one state, but is no longer there at the following state. Other agents can deduce that the variable *at(package)* has some private value. Or, consider the location of the truck. This is a private variable, and thus, other agents should not be able to distinguish between states that differ only in its value. However, when the truck is in location a, it has children in which packages appear or disappear from a, whereas when it is in location b, packages will appear or disappear from loca-

tion b. From this, other agents can deduce the existence of a private variable that distinguishes between these two states. Moreover, they can also infer the number of its possible values. While it is not necessarily the case that agents will be able to make these deductions, the lack of formal guarantees to the contrary is worrisome.

In the area of *secure multi-party computation* [Yao, 1982], a subfield of Cryptography, stronger privacy guarantees are sought from multi-party algorithms. The goal of methods for secure multi-party computation is to enable multiple agents to compute a function over their inputs, while keeping these inputs private. More specifically, agents $\varphi_1, \ldots, \varphi_n$, having *private* data $x_1, \ldots, x_n$, would like to jointly compute some function $f(x_1, \ldots, x_n)$, without revealing any information about their private information, other than what can be reasonably deduced from the value of $f(x_1, \ldots, x_n)$. Thus, we will say that a multi-agent planning algorithm is *strongly private* if no agent can deduce information beyond the information that can be deduced from its own actions' description, the public projection of other agents' actions, and the public projection of the solution plan, about the existence of a value or a variable that is private to another agent, or about the model of an action. More specifically, we will say that a variable or, a specific value of a variable is *strongly private* if the other agents cannot deduce its existence from the information available to them.

Strong privacy is not simply a property of an algorithm. It requires that we consider issues such as: the nature of the other agents, their computational power, and the nature of the communication channel. For example, the other agents may be curious but honest, that is, they follow the algorithm faithfully, but will try to learn as much as they can from the other agents. Or, they may be malicious, i.e., they will deviate from the algorithm if this allows them to gain private information. How much information agents can learn will depend on the information they are exposed to within the algorithm, whether they share information with other agents, what their computational power is, and even the nature of the communication channel – e.g., whether it is synchronous or asynchronous, and whether it is lossy.

Existing algorithms for classical multi-agent planning guarantee only weak privacy, which, as observed above, is a somewhat superficial notion. The first discussion of potentially stronger privacy guarantees appeared in [Nissim and Brafman, 2014] in the context of the MAFS algorithm. But that discussion came short of providing formal results and tools. Indeed, in MAFS agents can infer information about an agent's private state from the public part of its descendent states, as illustrated above for the truck's location variable. The goal of this paper is to provide an enhanced version of MAFS, called SECURE-MAFS, that reveals less information and comes with clearer privacy guarantees and tools.

The MAFS Algorithm

The multi-agent forward search algorithm (MAFS) [Nissim and Brafman, 2014] performs a distributed version of forward-search planning. In fact, it is a schema for distributing search algorithms while preserving some privacy. The distribution is along the "natural" lines. An agent φ_i will expand a state using its own actions only. If the public projection of an action of φ_j is applicable in this new state, φ_i will send the state to φ_j who will expand it. This idea can be applied to any search algorithm, although a bit of care needs to be taken to identify solutions, especially if an optimal one is sought (see [Nissim and Brafman, 2014] for the details).

Algorithms 1-3 depict the MAFS algorithm for agent φ_i. In MAFS, each agent maintains a separate search space with its own *open* and *closed* lists. It expands the state with the minimal f value in its open list, which is initialized with the agent's projected view of the initial state. When an agent expands state s, *it uses its own operators only*. This means that two agents (that have different operators) expanding the same state, will generate *different* successor states.

Since no agent expands all relevant search nodes, messages must be sent between agents, informing one agent of open search nodes relevant to it expanded by another agent. Agent φ_i characterizes state s as relevant to agent φ_j if φ_j has a public operator whose public preconditions (the preconditions φ_i is aware of) hold in s, and the creating action of s is public. In that case, Agent φ_i will send s to Agent φ_j.

Algorithm 1 MAFS for agent φ_i

```
1: Insert I into open list
2: while TRUE do
3:    for all messages m in message queue do
4:       process-message(m)
5:    s ← extract-min(open list)
6:    expand(s)
```

Algorithm 2 process-message($m = \langle s, g_{\varphi_j}(s), h_{\varphi_j}(s) \rangle$)

```
1: if s is not in open or closed list or g_{φi}(s) > g_{φj}(s) then
2:    add s to open list and calculate h_{φi}(s)
3:    g_{φi}(s) ← g_{φj}(s)
4:    h_{φi}(s) ← max(h_{φi}(s), h_{φj}(s))
```

Algorithm 3 expand(s)

```
1: move s to closed list
2: if s is a goal state then
3:    broadcast s to all agents
4:    if s has been broadcasted by all agents then
5:       return s as the solution
6: if the last action leading to s was public then
7:    for all agents φj ∈ Φ with a public action whose public preconditions are satisfied in s do
8:       send s to φj
9: apply φi's successor operators to s
10: for all successors s' do
11:    update g_{φi}(s') and calculate h_{φi}(s')
12:    if s' is not in closed list or f_{φi}(s') is now smaller than it was when s' was moved to closed list then
13:       move s' to open list
```

Messages sent between agents contain the full state s, the cost of the best plan from the initial state to s found so far, and the sending agent's heuristic estimate of s. The values of private variables in s are encrypted so that only the relevant agent can decipher them. By definition, if q is private to an

agent, other agents do not have operators that affect its value, and so they do not need to know its value. They can simply copy the encrypted value to the next state.

When φ receives a state via a message, it checks whether this state exists in its open or closed lists. If it does not appear in these lists, it is inserted into the open list. If a copy with higher g value exists, its g value is updated, and if it is in the closed list, it is reopened. Otherwise, it is discarded. Once an agent expands a *solution* state s, it sends s to all agents and awaits their confirmation, which is sent whenever they expand, and then broadcast state s (Line 3 of Algorithm 3). For more details, see [Nissim and Brafman, 2014].

Secure MAFS

MAFS does not require agents to know private information of other agents, nor is such information provided to them. Although agents see the private state of other agents, it is encrypted, and they cannot and need not manipulate it. Thus, it is weakly private. However, agents are exposed to more information than the projected public actions of other agents and the solution plan because they receive many intermediate states during the search process. From this information, it may be possible, in principle, to deduce private information. For example, from the complete search tree, even with encrypted private states, an agent may be able to deduce that the state of an agent is identical in two different search states based on the public parts of the sub-tree rooted at them. Thus, it could deduce the number of private variables of other agents. And maybe, potentially, construct a model of the transitions possible between them.

In reality, agents are not exposed to the entire search space, as only a small portion of it is explored typically, some states are not sent to them, and it may not be easy to deduce that one state is a parent of another state. However, this observation is neither a proof nor a guarantee that MAFS maintains privacy. We now develop SECURE-MAFS, a variant of MAFS for which we can offer better guarantees.

The basic idea is as follows: the effect of φ_i's action on the components of the state that are not private to φ_j is the same, regardless of φ_j's private state. Thus, if φ_j knows the effect of φ_i's action on a state s_1, it knows its effect on a similar state s_2 that differs only in its private state. In fact, the same holds true for a sequence of actions executed by agents other than φ_j. Hence, based on the work done on s_1 by other agents, φ_j can simulate the effect of the work that will be done on s_2 without actually sending it. The resulting algorithm provides better privacy, and can also lead to fewer messages.

Given a state s, recall that $s_{pr\text{-}i}$ is the private state of φ_i in s. We write s_{-i} to denote the part of state s that is not private to agent φ_i. Thus, s can be partitioned to $s_{pr\text{-}i}$ and s_{-i}. We write s_i to denote the part of the state that consists of variables private to i and public variables. That is, the information about s that is readily accessible to φ_i. We say that s' is an i-child of state s (and s is an i-parent of s') if s' is obtained from s by the application of a sequence $a_1, \ldots, a_l$ of actions of agents other than φ_i, and the last action in this sequence, a_l, is public. Intuitively, this means that s' is a state that would be sent by some agent to φ_i, and φ_i was not involved in the generation of any state between s and s'.

Instead of a regular closed list, each agent φ_i maintains a set of three-tuples. The first part is the *key* which contains a non-private part of a state that was sent (i.e., something of the form s_{-i}). The second part is a list of private parts of states consistent with the key, that were sent or were supposed to be sent. The third part is the non-private part of their i-children. That is, if state s was sent, there must be a three-tuple with key s_{-i} and second part containing $s_{pr\text{-}i}$. The second part may contain additional private states, $s'_{pr\text{-}i}$, for every state s' sent by the agent that satisfies: $s'_{-i} = s_{-i}$. The third part may be initially empty, but will grow to contain s''_{-i} for every state s'' such that s'' is an i-child of s or of any s' whose private part appears in the second list. That is, the third part is a list of i-children of states whose non-private part is identical to s that were sent or would have been sent in MAFS to φ_i.

The modified *expand*, called *secure-expand*, differs in two elements. Instead of sending the state only to agents that have an applicable action (line 7 in *expand*), we send it to all agents. And we replace "send" (line 8 in *expand*), with "virtual-send." The virtual-send (algorithm 6) ensures that no two states with the same non-private component will be sent by an agent. Suppose that φ_i wants to send state s to other relevant agents. First, it checks whether in the past it sent a state s' such that $s_{-i} = s'_{-i}$. If not, it sends s, as before, but adds a new three-tuple to the list of sent items of the form $\langle s_{-i}, \{s_{pr\text{-}i}\}, \{\}\rangle$. If, however, a state s' such that $s_{-i} = s'_{-i}$ was sent earlier, then it simply adds $s_{pr\text{-}i}$ to the second part of the three-tuple whose key is s'_{-i}, and it *does not* send s. Instead, it "virtually" sends it, emulating the answer that it would get. That is, for every i-child s'' of s' in this three-tuple, it sends itself a message $\hat{s}$ such that $\hat{s}_{-i} = s''_{-i}$ and $\hat{s}_{pr\text{-}i} = s'_{-i}$. That is, it knows that if in the past it sent a message s' and received an i-child s'' then if it sends s now, it will receive something that is identical to s'', except that its private part will be the same as in s.

Algorithm 4 secure-process-message($m = \langle s, g_{\varphi_j}(s), h_{\varphi_j}(s)\rangle$)

```
1: Let S = {s}
2: if s_{-i} has an i-parent then
3:     Let s'_{-i} be the i-parent of s_{-i}
4:     for all element s''_{pr-i} in the 2nd component of the tuple
       with key s'_{-i} do
5:         Replace s_{pr-i} with s''_{pr-i} in s
6:         S = S ∪ {s}
7: else
8:     Replace s_{pr-i} with I_{pr-i}
9:     S = S ∪ {s}
10: for all s ∈ S do
11:     if s is not in open or closed list or g_{φi}(s) > g_{φj}(s)
        then
12:         add s to open list and calculate h_{φi}(s)
13:         g_{φi}(s) ← g_{φj}(s)
14:         h_{φi}(s) ← max(h_{φi}(s), h_{φj}(s))
```

Secure-process-message modifies *process-message* to be consistent with the above ideas. First, when a state s'' is re-

Algorithm 5 secure-expand(s)

```
1: move s to closed list
2: if s is a goal state then
3:    broadcast s to all agents
4:    if s has been broadcasted by all agents then
5:       return s as the solution
6: if the last action leading to s was public then
7:    virtual-send s to all agents
8: apply φ_i's successor operators to s
9: for all successors s' do
10:   update g_{φ_i}(s') and calculate h_{φ_i}(s')
11:   if s' is not in closed list or f_{φ_i}(s') is now smaller than
      it was when s' was moved to closed list then
12:      move s' to open list
```

ceived by φ_i, we identify the state s such that s'' is an i-child of s. (Technically, this is done by maintaining a state id in the private part of states sent). We then record this fact in the three-tuple that corresponds to s_{-i}. The fact that s'' is an i-child of s implies that for every s' such that $s'_{-i} = s_{-i}$ there exists an i-child $\hat{s}$ such that $\hat{s}_{-i} = s''_{-i}$ and $\hat{s}_{pr\text{-}i} = s'_{pr\text{-}i}$. Thus, we treat $\hat{s}$ as if it, too, was just received by φ_i and add it to φ_i's open list. The modified code for the two procedures appears in Algorithms 4 & 5.

Algorithm 6 virtual-send(s,φ_j)

```
1: if a three-tuple with key s_{-i} exists then
2:    for all i-children s'_{-i} in this three-tuple do
3:       Let s̄ = s'_{-i} · s_{pr-i}. That is, augment s'_{-i} with the
         local state of φ_i in s.
4:       Add s̄ to your open list
5: else
6:    Replace s_{pr-i} with a unique id for tracking i-children
7:    Send s to φ_j
8:    Create new three-tuple: {s_{-i}, {s_{pr-i}}, {}}.
```

The net effect of these changes is that no two states with the same non-private part are sent, yet the entire reachable search-space can be explored. While this may have a computational advantage of fewer messages sent, we focus on the impact on privacy when agents always see a single private state associated with every non-private part of the global state.

Soundness and Completeness

Lemma 1. *If s is inserted into the open list of an agent φ_i then s is reachable.*

Proof. First, we note that MAFS and SECURE-MAFS generate only action sequences in which a private action of an agent is followed by another action of that agent. It was shown in [Nissim and Brafman, 2014] that for any goal involving public variables only, it is sufficient to consider such sequences only.

The proof is by induction on the step in which the state s was inserted into the open list. We assume an interleaving semantics, as true concurrency is not required by the algorithm. The base case is I, which is the first state inserted into any open list, which is reachable. For the inductive step, suppose s was inserted into φ_i's open list. There are three cases:

1. s was inserted by an expansion of some state s' via action a (line 12 of *secure-expand*). Since s' was inserted into the open list of φ_i earlier, it is reachable, and so is s.

2. s was inserted in line 12 of *secure-process-message*. This means that prior to the insertion of s, a message containing some state s' was received by φ_i from φ_j. If s' was sent, it was taken out of the open list of φ_j earlier, and hence it is reachable. We know that s is obtained from s' as follows: we find the i-parent of s', s''. s'' was sent earlier by i, and hence it is reachable. There is some state s''' that appeared in the open list of φ_i prior to the insertion of s, such that $s'''_{-i} = s''_{-i}$ and s is defined so that $s_{-i} = s'_{-i}$ and $s_{pr\text{-}i} = s'''_{pr\text{-}i}$. We claim that s is reachable.

To see this, notice that all states between s'' and s' must have been obtained by expansion. Otherwise, some component of s'' and s', other than $s''_{pr\text{-}i}$ must be different, in which case $s'''_{-i} \neq s''_{-i}$. Because $s'''_{-i} = s''_{-i}$, the same sequence of actions is applicable at s''', and must lead to a state whose non-private component must be the same as s'_{-i}. Its private component must be the same as $s'''_{pr\text{-}i}$ because that private state is not affected by these actions and none of these actions are in A_i (by definition of i-parent).

3. s was inserted in line 4 of *virtual send.* This is similar to the above case. s' is taken out of φ_i's open list (and thus, reachable). We see that some other state s'', such that $s''_{-i} = s'_{-i}$ has already been sent by φ_i, and is thus reachable. It has an i-child s'''. Following the same line of reasoning, s, which is obtained by applying to s' the same actions that led from s'' to s''' is also reachable.

□

To simplify the remaining proofs we assume, without loss of generality, that there are public actions only. This can be achieved by replacing every set A_i with all legal sequences of private actions followed by a single public action, all from A_i. There is no loss of generality because no information is exchanged following private actions, and a pubic goal is reachable iff it is reachable via a sequence of actions in which a private action by an agent is always followed by another action (private or public) by the same agent.

Lemma 2. *If s is reachable via some sequence of actions then s will be generated by some agent.*

Proof. The proof is by induction on the length of the action sequence $a_1, \ldots, a_k$ leading to s.

For $k = 0$, $s = I$ which is inserted initially to all open lists. Next, consider a sequence of length $k+1$ reaching s. Assume $a_{k+1} \in A_i$. Denote the state reached following the first k actions by s'. By the inductive hypothesis, s' was generated by some agent φ_j. If $i = j$ then that agent will also generate s. Otherwise, we know that φ_j must virtual-send s' to all agents eventually (as it inserts it into its open list and, unless a goal is found earlier, will eventually expand it and send it) and in particular, to φ_i. If it actually sends s', then we know φ_i will insert it into its open list and eventually expand it, generating s. If it does not send it to φ_i, we know that φ_j sent some other state s'' to φ_i, where $s'_{-j} = s''_{-j}$. a_{k+1} is applicable at

s'' because of this. Thus, at some point φ_i will apply it and send $a_{k+1}(s'')$ to φ_j. The latter would recognize that it is an i-child of s'', and would generate a state s''' such that $s'''_{-j} = (a_{k+1}(s''))_{-j}$ and $s'''_{pr-j} = (a_{k+1}(s''))_{pr-j}$. However, since $a_{k+1} \notin A_j$, $s''' = a_{k+1}(s') = s$, as required. □

We note that the soundness and completeness of SECURE-MAFS also implies that the secure variant of MAD-A* remains optimal. SECURE-MAFS can also lead to the transmission of fewer messages compared to MAFS. In MAFS' *expand,* a state is sent to all agents that have an applicable public action at that state. From the completeness proof above, it can be seen that SECURE-MAFS is still complete if the state is sent only to agents that have an applicable action in that state *and* the agent that sent the parent state. Thus, SECURE-MAFS will send at most one more message per state than MAFS, whereas if there are multiple paths that differ only on the private effects (e.g., driving in different routes, or using different vehicles, or tools) its caching mechanism will refrain from sending all but one state to *all* agents with applicable actions.

Privacy Guarantees

A search-based approach in which intermediate search nodes are shared among agents is unlikely to be strongly private always, yet we now show that SECURE-MAFS maintains strong privacy of some key aspects of the problem.

The forward search tree G^{Π} associated with a planning task Π is a well known concept. It contains the initial state, and for every state in it, it contains, as its children, all states obtained by applying all applicable actions to it. Leaf nodes are goal states that satisfy the optimization criteria (e.g., in satisficing planning, all goal states; in optimal planning, goal states with optimal f value) and dead-ends. In principle, the states in the tree can be generated in any topological order, but the use of a particular search algorithm restricts the order further. We define the *$-i$-projected tree*, G^{Π}_{-i}, to be the same tree, except that each state s is replaced by s_{-i}.

At present we do not have techniques that can take into account differences in cost and heuristic value and their impact on the order of generation of different sequences of actions by a particular search algorithm (but see the discussion). Thus, as before, we focus on the simpler case where all actions are public, have unit cost, and the heuristic value for all states does not depend on the private variables. While there is no loss of generality in assuming all actions are public, assuming that actions have identical cost and the heuristic value is not affected by the private part of the state is a restriction, especially if we assume that private actions are compiled away (in which case their cost is ignored).

Theorem 1. *Let Π and Π' be two planning problems. If $G^{\Pi}_{-i} = G^{\Pi'}_{-i}$ and A_i contains unit cost public actions only, and the heuristic value of a node depends only on its non-private component, then agents other than φ_i cannot distinguish between an execution of* SECURE-MAFS *on Π and Π'*

Proof. $G^{\Pi}_{-i} = G^{\Pi'}_{-i}$ implies that the search-tree is identical, except for, possibly, the various private parts (and multiplicity) of states with identical non-private part. Given our assumption on g and h, the expansion order of this tree depends only on the non-private component of the states, which is identical in both cases. Messages are sent based on the non-private part only, so in both cases, φ_i will send identical messages. Different messages may be virtually sent, but this will not impact what other agents see. Since other agents see the same messages, they will send the same messages to φ_i. □

Theorem 1 provides a tool for proving strong privacy. As an example, consider the case of independent private variables. Formally, v is an *independent private* variable of φ_i if v is a private variable of φ_i and if v appears in a precondition of a then all the variables that appear in a's description are private, and all its effects are on independent private variables. For example, suppose that the truck doesn't only deliver goods, but the driver sells drugs on the side, having actions of buying and selling them at various places. The action of selling/buying a drug at a location has no effect on public variables, nor on private variables that can impact public variables.

Given a set of variables, the actions in the domain induced by the removal of these variables contain the original actions, except for those in which the removed variables appear as preconditions and/or effects. In our example, if we remove the variable *has-drugs(agent)*, the actions of selling and buying drugs are also removed.

Lemma 3. *Let Π be a planning problem and let R be a set of independent private variables of φ. Agents other than φ_i cannot distinguish between an execution of* SECURE-MAFS *on Π and Π_{-R}, the problem obtained by removing R, assuming R does not influence h.*

Proof. The addition or removal of private independent variables does not influence G^{Π}. It simply leads to additional public actions (when we compile away private actions) that differ only in their effect on the variables in R. Since the value of variables in R does not influence our ability to apply any of the other actions, $G^{\Pi}_{-i} = G^{\Pi_{-R}}_{-i}$. □

We illustrate the utility of the above techniques by showing that SECURE-MAFS is strongly private for logistics. We say that a location is public if more than one agent can reach it.

Lemma 4. *Under the above assumptions,* SECURE-MAFS *is strongly private for logistics.*

Proof. The public information in logistics is whether or not a package is in a public location. Thus, consider two planning problems Π and Π' that have the same set of packages, the same set of public locations, and, initially, identical locations for packages that are in locations that are not private to φ_i. We claim that $G^{\Pi}_{-i} = G^{\Pi'}_{-i}$. To see this, note that two states that differ only in the location of packages that are in places private to φ_i have the same sub-tree, projected to $-i$ under the assumption that every private location is reachable from every other private location. Thus, from Theorem 1 it follows that agents cannot distinguish between these domains. □

Suppose, further, that logistics is augmented with public actions that are executable in a private location. For example, suppose the amount of fuel of an agent is public, and agents can fuel in private locations. The above result remains true. On the other hand, consider a, somewhat contrived, logistics

variant in which certain private locations are reachable from some other private locations only by passing through some public locations, and that trucks must unload their cargo when passing in a public location. In that case, $G^{\Pi}_{-i} \neq G^{\Pi'}_{-i}$, and we would not be able to apply Theorem 1. Thus, we see that our strong privacy proofs are sensitive to the specifics of the domains. Finally, note that in all proofs above, we assume that the agents are honest but curious, and these proofs are correct even if all agents collude, combining their information.

We now briefly, and informally, consider two other domains. The *satellites* domain is strongly private, both for MAFS and SECURE-MAFS because no satellite supplies or destroys preconditions of actions of another satellite, and the only public actions are actions that achieve a sub-goal. The only information one agent can learn about another agent when this property holds is whether or not it is able to achieve a sub-goal. Private sub-goals can be modelled by adding a proposition such as *private-sub-goals-achieved* and an action that achieves it with the set of private sub-goals as its (private) preconditions, and the above remains true. The *rovers* domain is more similar to *logistics*. Agents can block or free a shared resource, the channel, required for communication, and some of the sub-goals are achievable by multiple agents. Because of the shared channel, all communication actions are public. Private information includes the location of the agent, the state of its internal instruments, the existence of such instruments, and what information the agent has acquired. We claim that this information is strongly private in SECURE-MAFS in the following sense. If an agent can move between locations freely, without requiring intermediate public actions (which implies that it can collect information in whatever order it wishes), the projected search trees of two planning problems in which the agents have identical capabilities, that is, they can acquire the same information (pictures, soil samples, etc.), are identical. Thus, external agents cannot differentiate between them.

Discussion and Summary

We presented SECURE-MAFS, a new variant of the state-of-the-art MAFS algorithm with better privacy guarantees and potentially fewer messages. Beyond SECURE-MAFS, our central contribution is the first formal discussion of strong privacy in planning, a sufficient condition for strong privacy in SECURE-MAFS, and an illustration of its use. Consequently, we believe this work can play an important role in placing the discussion and analysis of privacy preserving planning algorithms on much firmer ground.

There is much left for future work. We feel that the notion of strong privacy requires additional development to capture some of our intuitions, and new proof techniques. Specifically, our current central technique focuses on showing the equivalence of two problem domains, but can we give a constructive technique for showing that a certain variable is strongly private? Another issue is the restriction to unit cost actions and heuristics that are not sensitive to private variables. A simple variant of SECURE-MAFS with a form of ϵ-exploration might overcome this problem: with probability ϵ the choice of next node to expand from the open list is random, and with probability $1 - \epsilon$ we expand the first node. In that case, the expansion order is less sensitive to the heuristic value, and it is more difficult to differentiate between search trees for different problems. We note that in this respect, MAFS has a certain advantage: it appears that an agent cannot tell the relation between a sequence of nodes generated by another agents, i.e., whether the nodes are siblings or are ancestors of one another. This property could be useful for proving privacy guarantees for MAFS.

Acknowledgements: We thank the anonymous IJCAI'15 and DMAP'15 WS reviewers for their useful comments, and the support of ISF through Grant 933/13, the Lynn and William Frankel Center for Computer Science, and the Helmsley Charitable Trust through the Agricultural, Biological and Cognitive Robotics Center of Ben-Gurion University.

References

[Bernstein *et al.*, 2005] Daniel S. Bernstein, Eric A. Hansen, and Shlomo Zilberstein. Bounded policy iteration for decentralized pomdps. In *IJCAI*, pages 1287–1292, 2005.

[Bonisoli *et al.*, 2014] Andrea Bonisoli, Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. A privacy-preserving model for the multi-agent propositional planning problem. In *ICAPS'14 Workshop on Distributed and Multi-Agent Planning*, 2014.

[Brafman and Domshlak, 2008] Ronen I. Brafman and Carmel Domshlak. From one to many: Planning for loosely coupled multi-agent systems. In *ICAPS*, pages 28–35, 2008.

[Durfee, 2001] Edmund H. Durfee. Distributed problem solving and planning. In *EASSS*, pages 118–149, 2001.

[Jonsson and Rovatsos, 2011] Anders Jonsson and Michael Rovatsos. Scaling up multiagent planning: A best-response approach. In *ICAPS*, 2011.

[Luis and Borrajo, 2014] Nerea Luis and Daniel Borrajo. Plan merging by reuse for multi-agent planning. In *ICAPS'14 Workshop on Distributed and Multi-Agent Planning*, 2014.

[Nissim and Brafman, 2013] Raz Nissim and Ronen I. Brafman. Cost-optimal planning by self-interested agents. In *AAAI*, 2013.

[Nissim and Brafman, 2014] Raz Nissim and Ronen I. Brafman. Distributed heuristic forward search for multi-agent planning. *Journal of AI Research*, 51:292–332, 2014.

[ter Mors *et al.*, 2010] Adriaan ter Mors, Chetan Yadati, Cees Witteveen, and Yingqian Zhang. Coordination by design and the price of autonomy. *Autonomous Agents and Multi-Agent Systems*, 20(3):308–341, 2010.

[Torreño *et al.*, 2014] Alejandro Torreño, Eva Onaindia, and Oscar Sapena. Fmap: Distributed cooperative multi-agent planning. *Applied Intelligence*, 41(2):606–626, 2014.

[Yao, 1982] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.