BLOCKING UNDERHAND ATTACKS BY HIDDEN COALITIONS

Matteo Cristani, Erisa Karafili and Luca Viganò
Dipartimento di Informatica, Università degli Studi di Verona, Verona, Italy

Keywords: Multi-agent systems, Coalitions, Security, Blocking attacks.

Abstract: Similar to what happens between humans in the real world, in open multi-agent systems distributed over the Internet, such as online social networks or wiki technologies, agents often form coalitions by agreeing to act as a whole in order to achieve certain common goals. However, agent coalitions are not always a desirable feature of a system, as malicious or corrupt agents may collaborate in order to subvert or attack the system. In this paper, we consider the problem of hidden coalitions, whose existence and the purposes they aim to achieve are not known to the system, and which carry out underhand attacks, a term that we borrow from military terminology. We give a first approach to hidden coalitions by introducing a deterministic method that blocks the actions of potentially dangerous agents, i.e. possibly belonging to such coalitions. We also give a non-deterministic version of this method that blocks the smallest set of potentially dangerous agents. We calculate the computational cost of our two blocking methods, and prove their soundness and completeness.

1 INTRODUCTION

Context and Motivation. Similar to what happens between humans in the real world, in open multi-agent systems (Davidsson, 2001) distributed over the Internet, such as online social networks or wiki technologies, agents often form coalitions by agreeing to act as a whole in order to achieve certain common goals. For instance, agents may wish to collaborate in order to jointly create and use a group cryptographic key for ensuring the confidentiality and/or integrity of information shared within the group, e.g. (Rafaeli and Hutchison, 2003), or to partake in a mix network or some other anonymous remailer to achieve unobservability of communications, e.g. (Chaum, 1981), or to create secret interest groups within online social networks, e.g. (Sorniotti and Molva, 2010). However, agent coalitions are not always a desirable feature of a system, as malicious or corrupt agents may collaborate in order to subvert or attack the system. For instance, such agents may collaborate to attack the information in transit over different channels in a web service architecture or in a distributed wired and/or wireless computer network, e.g. (Wiehler, 2004), or they might forge and spread false information within the system, e.g. (Hahn et al., 2007).

In order to be able to rigorously formalize and reason about such positive and negative properties of agent coalitions, and thereby allow for the prevention or, at least, the identification of the entailed vulnerabilities, a number of different formal approaches have been recently proposed, such as (Ågotnes et al., 2008; Alur et al., 1998; van der Hoek et al., 2005; Oravec and Fogel, 2006; Pauly, 2001; Troquard et al., 2009; van der Hoek and Wooldridge, 2005).

In this paper, we consider the problem of *hidden coalitions*: a coalition is hidden in a system when its existence and the purposes it aims to achieve are not known to the system. Hidden coalitions carry out *underhand attacks*, a term that we borrow from military terminology. These attacks are particularly subtle since the agents that perform them are not outsiders but rather members of the system whose security properties are posed under threat. Moreover, the mere suspect that a group of individuals act as a whole is typically insufficient to come to a decision about their permanence as members of the system; this, of course, depends also on the nature of the system and the information it contains, since in the presence of highly security-sensitive information, systems may anyway opt for the exclusion of all suspected agents. However, in general, systems, and even more so open ones, will want to adopt a less restrictive policy, excluding only those agents whose malice has indeed been proved. Therefore, the defense against underhand attacks by hidden coalitions is a fundamental but complex matter.

Problems of a similar kind have been studied, for instance, in Game Theory (Aumann and Hart, 1994; Pauly and Parikh, 2003) in relation to the nature of

Cristani M., Karafili E. and Viganò L..
BLOCKING UNDERHAND ATTACKS BY HIDDEN COALITIONS.
DOI: 10.5220/0003189203110320
In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence* (ICAART-2011), pages 311-320
ISBN: 978-989-8425-41-6

collaboration and competition, and from the viewpoint of modeling group formation under the constraints of possible given goals. However, underhand attacks by hidden coalitions pose security problems that cannot be dealt with such traditional means. Nor can they solved by a simple, monotonic, approach based on Coalition Logic(s) such as (Ågotnes et al., 2008; Oravec and Fogel, 2006; Pauly, 2001; van der Hoek and Wooldridge, 2005).

To illustrate all this further, consider the following concrete example from an online social network such as Facebook, where abuse, misuse or compromise of an account can be reported to the system administration. In particular, a group of agents (in this case, Facebook users) can report a fake profile:

> You can report a profile that violates Facebook's Statement of Rights and Responsibilities by clicking the "Report/Block this Person" link in the bottom left column of the profile, selecting "Fake profile" as the reason, and adding the appropriate information. [...] *(Excerpt from* http://www.facebook.com/help/?search=fake*)*

The administrator of the system gives an ultimatum to the agent that uses the reported profile and then may, eventually, close it. An underhand coalition can exploit this report mechanism to attack an agent who possesses a "lawful" original profile: at first they create a fake profile with personal information and photos of the agent under attack, and then they become friends of her. After that, they report the original profile so that the administrator closes it. The report is a lawful action, and by creating the new profile and having a big enough number of agents who report the same profile no suspicion about the hidden coalition is raised, so that the attack succeeds.

Contributions. A formalism to define and reason about such hidden coalitions is thus needed. Indeed, Coalition Logic allows one to define coalitions that are explicit (i.e. not hidden) and is characterized by monotonic permissions to act in groups and individually. What is missing, however, is the notion of hidden coalition and a method to block the underhand attacks such coalitions carry out. The idea underlying our approach is to circumscribe the problem in algebraic terms, by defining a system that can be represented by a coalition logic, and then activate a non-monotonic control on the system itself to block the underhand attacks that hidden coalitions are attempting to carry out.

More specifically, we consider multi-agent systems whose security properties depend on the values of sets of logical formulas of propositional logic, which we call the *critical* (or *security*) *formulas* of the systems: for concreteness, we say that a system is *secure* if all the critical formulas are false, and is thus insecure if one or more critical formula is true. (Of course, we could also invert the definition and consider a system secure when all critical formulas are true.) The system agents control the critical formulas in that they control the propositional variables that formulas are built from: we assume that every variable of the system is controlled by an agent, where the variables controlled by an agent are controlled just by that agent without interference by any other agent. The actions performed by each agent consist thus in changing some of the truth values of the variables assigned to that agent, which means that the values of the critical formulas can change due to actions performed by the agents, including in particular malicious insider agents who form hidden coalitions to attack the system by making critical formulas become true. Returning to the Facebook example, this is exactly what happens when agents report the original profile as fake by setting the flag (clicking on the link).[1]

At each instant of time, agents ask the system to carry out the actions they wish to perform, i.e. changing the truth value of the variables they control, and the system has to decide whether to allow such actions, but without knowing of the existence of possible hidden coalitions and thus at the risk of the system becoming insecure. To block such attacks, we formalize here a *deterministic blocking method*, implemented by a greedy algorithm, which blocks the actions of potentially dangerous agents. We prove that this method is sound and complete, in that it does not allow a system to go in an insecure state when it starts from a secure state and it ensures that every secure state can be reached from any secure state. However, this algorithm is not optimal as it does not block the smallest set of potentially dangerous agents. We thus introduce also a *non-deterministic blocking method*, which we obtain by extending the deterministic method with an oracle to determine the minimum set of agents to block so to ensure the security of the system. We show that the soundness and completeness result extends to this non-deterministic method.

We also calculate the computational cost of our two blocking methods. This computational analysis is completed by determining upper bound results for the problem of finding a set of agents to be blocked so to prevent system transitions into insecure states,

[1] In this paper, we do not consider how the administrator decides to close the profile, nor do we consider in detail the non-monotonic aspects of how agents enter/exit/are banned from a system or enter/exit a hidden coalition, or how members of a hidden coalition synchronize/organize their actions. All this will be subject of future work.

and the problem of finding an optimal set of agents satisfying the above condition.

Organization. In §2, we introduce our approach to the problem of blocking underhand attacks by hidden coalitions. §3 and §4 respectively introduce our deterministic and non-deterministic blocking methods, giving concrete examples for their application. In §5, we study the computational aspects of these two methods, calculating in particular their computational cost, and show that they are both sound and complete. Finally, in §6, we summarize our main results, discuss related work and sketch future work. Proofs of the formal results are given in (Cristani et al., 2010).

2 AN APPROACH TO THE PROBLEM OF BLOCKING UNDERHAND ATTACKS

We introduce our approach to the problem of blocking underhand attacks. We also recall some basic notions and, in particular, the relevant notions of the Coalition Logic of Propositional Control CL-PC (Oravec and Fogel, 2006; van der Hoek and Wooldridge, 2005).

2.1 Syntax

We consider multi-agent systems $\mathcal{S}$ that are described by a set of *critical* (or *security*) *formulas* Φ and by a temporal sequence $\sigma : T \rightarrow \Theta(\Phi)$, with T the temporal axis and $\Theta(\Phi)$ the propositional assignment in the set of formulas. In this work, we focus only on the formulas in Φ, which represent the security-critical characteristics of a system (which depend on the application and which we thus do not describe further here, as our approach is independent of the particular application). We say that a system is *secure* if all the critical formulas are false, and it becomes insecure if one or more $\phi \in \Phi$ becomes true. Hence, the *state* of a system is defined by the value of the propositional variables that occur in the critical formulas of Φ.

The *agents* of a system $\mathcal{S}$ control the set Φ and hence the state of $\mathcal{S}$. We require that there is no formula in our systems that cannot change its truth value. Moreover, the distribution of the variables to the agents should be such that one formula cannot be controlled by one single agent, but rather different agents control one formula, and every formula is controlled by some agents. In particular, for a set *Ag* of system agents:

- every variable of the system is controlled by an agent $a \in Ag$, and
- the variables controlled by an agent are controlled just by that agent without interference by any other agent.

The actions performed by each agent $a \in Ag$ are thus the changing of the truth values of the variables assigned to a. The agents we consider are *intelligent agents* in the sense of (Wooldridge and Jennings, 1995; Wooldridge and Jennings, 1998): they are autonomous, have social ability, reactivity and pro-activeness, and have mental attitudes, i.e. states that classify the relation of the agents and the cognitive environment. In our approach, we consider intelligent agents but do not make specific assumptions about their mental attitudes, except for their collaborative attitudes that constitute a threat to (the security of) the system.[2]

In Game Theory (van der Hoek et al., 2005), strategies are often associated with a preference relation for each agent that indicates which output the agent is going to select in presence of alternatives. In our approach, agents change the value of "their" variables according to their strategies and create coalitions with other agents so to be more expressive: by collaborating, agents can change the values of different variables and thus, ultimately, of the critical formulas that comprise such variables. The novelty in this work is that we don't deal just with coalitions that are known by the system but also with hidden coalitions, whose existence and purposes are unknown by the system.

Let us now formalize the language of our approach. Following CL-PC, given a set *Ag* of agents, a set *Vars* of propositional variables, the usual operators $\neg$ and $\vee$ of classic propositional logic, and the *cooperation mode* $\Diamond$, we consider formulas built using the following grammar:

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi \vee \phi \mid \Diamond_C \phi$$

where $p \in Vars$, $C \subseteq Ag$, and $\Diamond_C\phi$ is a *cooperation formula*. Slightly abusing notation, we denote with $Vars(\phi)$ the set of propositional variables that occur in ϕ and with $Ag(\phi)$ the agents that control the variables in $Vars(\phi)$. $\Diamond_C\phi$ expresses that the coalition C has the contingent ability to achieve ϕ; this means that the members of C control some variables of ϕ and have choices for ϕ such that if they make these choices and nothing else changes, then ϕ will be true.

[2] An extension of the work presented here with a detailed formalization of the mental and collaborative attitudes of the agents will be subject of future work.

2.2 Semantics

A model is a tuple $\mathcal{M} = \langle Ag, Vars, Vars|_1, ..., Vars|_n, \theta \rangle$, where: $Ag = \{1, ..., n\}$ is a finite, non-empty set of agents; $Vars = \{p, q, ...\}$ is a finite, non-empty set of propositional variables; $Vars|_1, ..., Vars|_n$ is a partition of *Vars* among the members of *Ag*, with the intended interpretation that $Vars|_i$ is the subset of *Vars* representing those variables under the control of agent $i \in Ag$; $\theta : Vars \to \{\top, \bot\}$ is a propositional valuation function that determines the truth value of each propositional variable.

Since $Vars|_1, ..., Vars|_n$ is a partition of *Vars*, we have: $Vars = Vars|_1 \cup ... \cup Vars|_n$ i.e. every variable is controlled by some agent; and $Vars|_i \cap Vars|_j = \emptyset$ for $i \neq j \in Ag$, i.e. no variable is controlled by more than one agent. We denote with $Vars|_C$ the variables controlled by the agents that are part of the coalition $C \subseteq Ag$. Given a model $\mathcal{M} = \langle Ag, Vars, Vars|_1, ..., Vars|_n, \theta \rangle$ and a coalition *C*, a *C*-valuation function is $\theta_C : Vars|_C \to \{\top, \bot\}$. Valuations θ extend from variables to formulas in the usual way and for a model $\mathcal{M} = \langle Ag, Vars, Vars|_1, ..., Vars|_n, \theta \rangle$ we write $\mathcal{M} \models \phi$ if $\theta(\phi) = \top$. We write $\models \phi$ if $\mathcal{M} \models \phi$ for all $\mathcal{M}$.

2.3 Secure and Insecure Systems

All the semantic notions introduced above actually depend on the current time, and we will thus decorate them with a superscript ^{S_t} denoting the system state at time *t*, e.g. θ^{S_t} and $\models^{S_t}$. Time is discrete and natural, and is defined with a non empty set of time points *T* and a transitive and irreflexive relation $\prec$ such that $t \prec u$ means that *t* comes before *u* for $t, u \in T$. In our case, since $t, t+1 \in T$ it follows naturally that $t \prec t+1$.

The passing of time is regulated by a general *clock*, which ensures that the system can execute a definite number of actions in an instant of time: at every clock of time, the system changes its state, which is thus defined by the actions that the system executes. Even if there are no actions to execute, the system changes its state from S_t to S_{t+1}, which in this case are equal.

We assume that each system $\mathcal{S}$ starts, at time t_0, from a secure state S_0, i.e. a state in which all the critical formulas of Φ are false, so that none of the features of the system is violated. In general:

$\mathcal{S}$ is *secure at a state* S_t iff $\not\models^{S_t} \phi$ for all $\phi \in \Phi$

and $\mathcal{S}$ is *secure* iff it is secure at all S_t.

At time *t*, the system is in state S_t and goes to state S_{t+1} and executes all the actions of the agents that want to change the value of their variables. Denoting with Γ_{t+1} the set of actions that the agents want to execute at the time instant *t*, we can write

$$S_t \overset{\Gamma_{t+1}}{\longmapsto} S_{t+1}.$$

and the aim of our approach is to guarantee that each reachable state S_{t+1} is secure, where the differences between S_t and S_{t+1} are in their respective Θ.

Since a coalition can change the value of the variables it controls, it can attempt to change the value of a critical formula to true; formally, for a coalition *C* and a formula ϕ if $\Diamond_C \phi$ is true then it means that *C* can make ϕ true and thus the system insecure, which we can write by negating the above definition or alternatively, and basically equivalently, as:

$\mathcal{S}$ is *insecure* at a state S_t iff $\models^{S_t} \Diamond_C \phi$ for some $C \subseteq Ag$ and some $\phi \in \Phi$

To help the control of the system (but without loss of generality), we can create a *filter* for the actions that imposes a limit on the number of the actions that can be executed in an instant of time. This can decrease the performance of the system, so we need a trade-off between control and performance.

3 A DETERMINISTIC BLOCKING METHOD

Our aim is to introduce a method that guarantees the security of the system, which amounts to blocking the actions of hidden coalitions. Indeed, in the case of "normal" coalitions, the property $\Diamond_C \phi$ allows us to list the actions of the agents in *C*, while if the coalition is hidden then we cannot block any action as we cannot directly identify the participants of a coalition we do not even know to exist. Since the actions of participants of hidden coalitions are not predictable, we cannot oppose these coalitions using $\Diamond$, so we introduce a method that disregards the existence of this property.

Our (main) method for the protection of the system is a *blocking method* based on the *greedy* Algorithm 1: the agents make a request to the system for the actions Γ_{t+1} they wish to execute at time *t*, and the system then simulates (via a method *Simulate* we assume to exist) the actions in order to control whether the system after the execution of the actions is still secure or not. The simulation says if the system can proceed with the execution of the actions or not, in which case it is given a list of the formulas Φ' that became true along with the set of agents $\mathcal{A}'$ that made them become true.

If the simulation says that the system can go in an insecure state, the blocking method constructs a

Algorithm 1: A GREEDY, DETERMINISTIC BLOCKING METHOD.

```
Simulate(Γ_{t+1}) = [Φ′ A′];
while (Φ′ ≠ ∅) do
    Create the matrix with Φ′ and A′;
    ∀a_i ∈ A′ : a_i → c_i, c_i = count(φ_i);
    Quicksort(c_1, ..c_k) = (c_x, ...);
    B = B ∪ a_x; {where a_x is the agent associated to c_x, that is the maximum counter of the marked cells}
    Simulate(Γ_{t+1} \ Γ|_{a_x}) = [Φ′ A′];
end while
```

matrix: in every column of the matrix there is one of the agents given by the simulation and in every row there is one of the formulas that became true during the simulation. We mark each cell that has as coordinates the agent that has variables in that formula, and then we eliminate the column that has more marked cells.[3] The corresponding agent is not eliminated, rather he is just blocked and his actions are not executed (by subtracting $\Gamma|_{a_x}$): the "dangerous" agents found in this way are put in a set $\mathcal{B}$ of blocked agents. The simulation is called again and so on, until the output of the simulation is an empty set of formulas, which means that by executing the remaining actions the system does not go in an insecure state. It is important to note that this method does not prevent the creation of hidden coalitions but can guarantee the system security from the attacks made by these coalitions.

The most important property of Algorithm 1 is that it never brings the system in an insecure state, as it blocks the actions of agents that can make the system insecure. We do not commit to a specific way that the blocking is actually done, as it depends on the particular observed systems and on the particular goals. For instance:

- Block the agent from changing the value of his variables until a precise instant of time. During this period, his variables are left unchanged or are controlled by the superuser/system administrator.
- Block the agent for an interval of time, which can be a default value or can be chosen in a random way, e.g. so that a hidden coalition doesn't know when the agent can be active and thus cannot organize another attack.
- Block the agent and remove his actions for that instant of time. At the next instant, the agent has the possibility to ask for his actions to be executed.
- Leave the variables unchanged, without making known to the agent if the value of the variables has been changed or not. This method can be improved by blocking the agent if he attempts to change the truth value of those variables again.

Other, more complex, blocking strategies can of course be given, e.g. by combining some the above.

Note also that, depending on the system considered, it could be that not all the requests for execution can be satisfied: *the maximum number n of actions that can be executed in an instant of time* can be chosen in different ways, with respect to the characteristics of the system. Here, we choose n to be the cardinality $|\Phi|$ of the critical formulas. The order used for taking these actions and executing them respects a FIFO queue, so the first n actions are executed.

Example 1. *As a concrete example of the application of the blocking method, consider a system $\mathcal{S}$ defined by the critical formulas*

$$\begin{aligned}\phi_1 &= v_1 \wedge v_2 \wedge (\neg v_3 \vee v_5 \vee \neg v_4)\\ \phi_2 &= (\neg v_5 \vee \neg v_3) \wedge \neg v_6\\ \phi_3 &= v_7 \wedge (\neg v_8 \vee \neg v_6)\\ \phi_4 &= (v_8 \vee v_5 \vee \neg v_9) \wedge v_2 \wedge v_1\end{aligned}$$

so that number of the action to be executed in an instant of time is $n = 4$ (the cardinality of the set of critical formulas that define the system), and let $Ag = \{a_1, a_2, a_3, a_4, a_5\}$ and $At = \{v_1, ..., v_9\}$. Further, consider the following distribution of the variables to the agents:

$$a_1 = \{v_1, v_7, v_8\} \quad a_2 = \{v_3\} \quad a_3 = \{v_2, v_6\} \quad a_4 = \{v_4, v_5\} \quad a_5 = \{v_9\}$$

Let us assume that the state S_t at time t is

$$\begin{aligned}&\theta^{S_t}(v_1) = \theta^{S_t}(v_5) = \bot\\ &\theta^{S_t}(v_2) = \theta^{S_t}(v_3) = \theta^{S_t}(v_4) = \theta^{S_t}(v_6) = \theta^{S_t}(v_7)\\ &\quad = \theta^{S_t}(v_8) = \theta^{S_t}(v_9) = \top\end{aligned}$$

and that we have the following actions Γ_{t+1} to be executed at time t in the FIFO queue:

$$\begin{aligned}&\theta^{S_{t+1}}(v_1) \hookleftarrow \top, \quad \theta^{S_{t+1}}(v_3) \hookleftarrow \bot,\\ &\theta^{S_{t+1}}(v_4) \hookleftarrow \bot, \quad \theta^{S_{t+1}}(v_6) \hookleftarrow \bot, \quad \ldots\end{aligned}$$

That is, v_1 should be set to $\top$ at state S_{t+1}, and so on. The algorithm simulates the first $n = 4$ actions, so that $\Phi' = \{\phi_1, \phi_2, \phi_3, \phi_4\}$ and $\mathcal{A}' = \{a_1, a_2, a_3, a_4\}$, and the matrix of Table 1 is constructed, which the

[3] It would be more efficient to consider only the variables of the formulas that become true, but if we take only these variables, we cannot prevent long-term strategies of hidden coalitions, consisting in the progressive reduction of the number of steps needed for making a security formula true. An optimization of the choice of variables to be considered in order to reduce the effectiveness of such long-term strategies will be subject of future work.

Table 1: Matrix constructed by the blocking algorithm for Example 1.

	a_1	a_2	a_3	a_4
ϕ_1	X	X	X	X
ϕ_2		X	X	X
ϕ_3	X		X	
ϕ_4	X		X	X

Table 2: Matrix of Table 1 sorted in a decreasing order of counters.

	a_3	a_1	a_4	a_2
ϕ_1	X	X	X	X
ϕ_2	X		X	X
ϕ_3	X	X		
ϕ_4	X	X	X	

algorithm sorts by the highest counter to produce the matrix in Table 2. a_3 is thus put into $\mathcal{B}$. The simulation takes place again, taking into account that we have blocked the value of the variables controlled by a_3 at the truth value of the instant of time t. The simulation gives as result the set $\Phi' = \{\phi_1, \phi_4\}$ e $\mathcal{A}' = \{\phi_1, \phi_2, \phi_4\}$. The matrix of Table 3 is created, which is already ordered (so the sorting will return the same matrix). So, we put in $\mathcal{B}$ the agent a_1, block its actions and make the simulation with the remaining actions. This simulation gives $\Phi' = \emptyset$, and thus the remaining actions can be executed without any risk for the system $\mathcal{S}$.

4 A NON-DETERMINISTIC BLOCKING METHOD

As we will see in §5, the above deterministic blocking method based on a greedy algorithm is sound and complete. However, this algorithm is not optimal as it cannot block the smallest set of potentially dangerous agents. We now introduce a non-deterministic method, which can be used for identifying optimal solutions. The method, which is implemented in Algorithms 2 and 3, is obtained by introducing an oracle (to determine the minimum set of agents to block so to ensure the security of the system) within the deterministic version, which makes the soundness and completeness results directly applicable to the non-deterministic version as well.

The idea is that the result given by the simulation is passed to the method *ScanOracle*, which creates all the subsets of the given set $\mathcal{A}'$ with cardinality $|\mathcal{A}' - 1|$ and finds the subsets with the maximum number of critical formulas that remain false, using the simulation. The simulation of all the subsets is done in parallel; the *ScanOracle* is the non-deterministic part of our algorithm. The result is passed to the main algorithm: if we find a subset of agents such that when executing their actions all critical formulas are false, then we have finished and we block the remaining agents that are not part of this subset; if not all the critical formulas remain false the result is passed recursively to *ScanOracle* until it is given a set of agents such that all the critical formulas stay false when simulating their actions. The rest of the agents in $\mathcal{A}'$ that are not part of the given subset are blocked. Using this method, we can have different best solutions but we choose one in a random way, where with "best solutions" we mean sets that have the same cardinality and are the biggest sets that make the critical formulas stay false, so that we block the smallest set of agents that make the critical formulas true.

Table 3: Matrix of Table 2 after the block of agent a_3.

	a_1	a_2	a_4
ϕ_1	X	X	
ϕ_4	X		X

Algorithm 2: A NON-DETERMINISTIC BLOCKING METHOD.

1: $Simulate(\Gamma_n) = [\Phi' \mathcal{A}']$;
2: $I = \mathcal{A}'$; $j = 0$;
3: **while** ($\Phi' \neq \emptyset$ & $I \neq \emptyset$ & $j < |\mathcal{A}'|$) **do**
4: $I' = ScanOracle(I)$;
5: For a random $I_i \in I'$
6: **if** $|\{\phi_i \,|\not\models \phi_i$ at the current state $\}| = |\Phi'|$ **then**
7: $I = \emptyset$;
8: **else**
9: $I = I'$;
10: **end if**
11: $j++$
12: **end while**
13: Choose a subset $I_i \in I'$ and put $\mathcal{A}' \backslash I_i$ in $\mathcal{B}$

Algorithm 3: SCANORACLE.

1: Generate the subset of I with cardinality $|I| - 1$
2: Execute the simulation in parallel for each subset I_i, where $i \in \{1, ..., |I|\}$
3: Take the I_i with the maximum number of $\{\phi_i \,|\not\models \phi_i\}$ and put them in I'
4: $I' = I' \cup I_i$
5: Eliminate the duplicates in I'
6: Return I'

Example 2. *As a concrete example of the application of Algorithm 2 (and of Algorithm 3), consider again the system of Example 1, with the same data. The simulation of the first* 4 *actions yields again*

$\mathcal{A}' = \{a_1, a_2, a_3, a_4\}$, *which is passed to ScanOracle, which in turn creates the subsets:*

$$I_1 = \{a_1, a_2, a_3\} \qquad I_2 = \{a_1, a_3, a_4\}$$
$$I_3 = \{a_1, a_2, a_4\} \qquad I_4 = \{a_2, a_3, a_4\}$$

The oracle takes these subsets and gives as results at the current state

$$\begin{array}{lllll} I_1: & \models \phi_1, & \models \phi_2, & \models \phi_3, & \models \phi_4. \\ I_2: & \models \phi_1, & \models \phi_2, & \models \phi_3, & \models \phi_4. \\ I_3: & \models \phi_1, & \not\models \phi_2, & \not\models \phi_3, & \models \phi_4. \\ I_4: & \not\models \phi_1, & \models \phi_2, & \models \phi_3, & \not\models \phi_4. \end{array}$$

The two subsets with maximum number of false critical formulas are I_3 and I_4, so $I' = I_3 \cup I_4$. Note that, since I_3 and I_4 have the same number of false formulas, it is enough to test just one of them to see if all the formulas are false or not; in this case, we have just two formulas. The ScanOracle is then called again with $I' = I_3 \cup I_4$ and it yields the subsets

$$I_5 = \{a_1, a_2\} \qquad I_6 = \{a_1, a_4\} \qquad I_7 = \{a_2, a_4\}$$
$$I_8 = \{a_2, a_3\} \qquad I_9 = \{a_2, a_4\} \qquad I_{10} = \{a_3, a_4\}$$

and thus the following results

$$\begin{array}{lllll} I_5: & \models \phi_1, & \not\models \phi_2, & \not\models \phi_3, & \models \phi_4. \\ I_6: & \not\models \phi_1, & \not\models \phi_2, & \not\models \phi_3, & \models \phi_4. \\ I_7: & \not\models \phi_1, & \not\models \phi_2, & \not\models \phi_3, & \not\models \phi_4. \\ I_8: & \not\models \phi_1, & \models \phi_2, & \models \phi_3, & \not\models \phi_4. \\ I_9: & \not\models \phi_1, & \not\models \phi_2, & \not\models \phi_3, & \not\models \phi_4. \\ I_{10}: & \not\models \phi_1, & \models \phi_2, & \models \phi_3, & \not\models \phi_4. \end{array}$$

Then $I' = I_7 \cup I_9 = I_7$ as these two subsets are identical. Using I', all the critical formulas are false, so it is the maximum subset of agents with which the system is secure. Hence, we block the remaining agents in $\mathcal{A}'$, which is the minimum set of agents for the blocking of which the system remains secure: $\{a_1, a_2, a_3, a_4\} \setminus \{a_2, a_4\} = \{a_1, a_3\}$.

5 COMPUTATIONAL COST, SOUNDNESS AND COMPLETENESS

In this section, we list some results for the deterministic and non-deterministic methods, which are proved in (Cristani et al., 2010). Recall that the maximum number *n* of actions that can be executed in an instant of time corresponds to the cardinality of the formulas in Φ. So, in the worst case, at each instant of time, there are *n* different agents that want to change the value of *n* different variables.

Theorem 1. *The computational cost of the greedy blocking method, Algorithm 1, and of the non-deterministic blocking method, Algorithm 2, is $O(n^3)$.*

We say that a blocking method, and thus the corresponding algorithm, is *sound* if it does not allow a system to go in an insecure state when it starts from a secure state S_0. It is not difficult to prove that:

Theorem 2. *The greedy blocking method, Algorithm 1, is sound.*

Let us now define the notion of a *state graph*. Recall that every state of the system is defined by an assignment of truth values to the variables, and a state is secure if it falsifies all security-related formulas. As a very simple example, in Figure 1 we give the state graph of a system with two variables $\{A, B\}$.

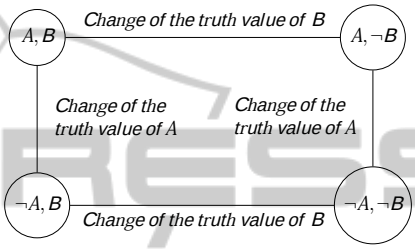

Figure 1: A state graph for a system with two variables.

In general, to denote the transitions executed by a system, we build a state graph as follows: every state is represented as a vertex of the graph, and every pair of vertices is connected by an edge when and only when the two edges differ by the truth value of one single variable, where the edge is labeled by the name of the agent that controls that variable. The resulting graph is *indirected*. In Figure 2, we give an example of such a graph, where we omit to specify all the values of the variables for readability, but instead denote with gray vertices the insecure states and with white vertices the secure ones.

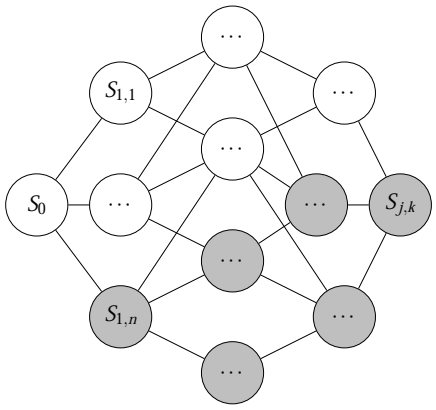

Figure 2: A state graph with secure and insecure states.

We say that a blocking method, and thus the corresponding algorithm, is *complete* if every secure state

can be reached from any secure state. To prove the completeness of the greedy blocking method, we pursue the following strategy:

1. First we prove that the state graph of the system is connected.
2. We prove that the subgraph formed by the vertices representing secure states is connected when the security formulas can be written as a set of Horn clauses.
3. We prove that every formula that we consider can be written as a disjunction of Horn clauses.
4. We show that two secure states, whose security formulas can be written as Horn clauses, are connected if and only there is a path of secure states in the above mentioned subgraph that can be traversed by the algorithm.
5. We show that the set of agents that have to be blocked, defined by a rewriting into Horn clauses of a security formula, is the union of agents that control variables occurring in one single Horn clause, and that can modify the value of the formula.
6. We show that the set of agents blocked by the greedy algorithm is a superset of the set of agents that control variables occurring in one single Horn clause in any rewriting of the formula.

In particular, we write $\mathcal{H}(\Phi,\Phi')$ to denote the set of the agents that control at least one variable of one Horn clause in one rewriting $\Phi'$of the security formula Φ, in such a way that by changing the value of one of these variables the value of the security formula can pass from $\bot$ to $\top$.

Let us observe a few simple facts that will be useful in the following. First of all, every secure state corresponds to a formula, obtained as the conjunction of the literals representing the truth values of the variables in that state. Since the single elements of the set of security formulas have to be false for the system to be secure, we can describe this situation directly by the set of secure states. Indeed, guaranteeing falseness of each security formula corresponds to falsifying the disjunction of the logical expressions representing the secure states. By the definition of state graph, we immediately have:

Lemma 1. *The state graph is connected.*

It would be tempting to presume that not only the set of states is connected, but also the set of secure states. However, this is untrue. Consider namely the case in which the system has two variables, A and B, so that there are four states as shown in Figure 1. Suppose that the security formula is $(\neg A \wedge B) \vee (A \wedge \neg B)$. The set of secure states is formed by the state in which both A and B are false and the state in which they are both true. Clearly the set of secure states is then disconnected.

Conversely, if the set of secure states is connected, the security formula can be written as a Horn clause (or a set of clauses, which is equivalent). To do so, we introduce the notion of *Horn rewriting* of a formula: a propositional formula is a *Horn clause* iff it can be written in *Conjunctive Normal Form* (i.e. as a conjunction of disjunctions of literals) in which every conjunct is formed by at most one positive literal (This is the standard notion of Horn clause, which we recall for preserving self-containedness.) It is well-know that every propositional formula can be written as a disjunction of Horn clauses.

A *Horn labeling* λ of the states of a system is an assignment of the system variables to one of the corresponding literals. Whenever, in a Horn labeling, $\lambda(v) = v$ for a variable v, the literal v will be considered positive by that labeling, and consequently literal $\neg v$ will be considered negative. If $\lambda(v) = \neg v$, then the literal v will be considered negative, and consequently literal $\neg v$ will be considered positive. We henceforth generalize the notion of Horn clause, by stating that a formula is a Horn clause when there exists a Horn labeling for it that makes it a Horn clause.

In the above example, the formula can be rewritten, by applying the distributive property, as $(A \vee B) \wedge (\neg A \wedge \neg B)$ and there exists a Horn labeling that makes the formula a Horn clause: $\lambda(A) = \neg A$ and $\lambda(B) = B$.

Lemma 2. *If the set of states that correspond to a security formula is connected, then the security formula is a Horn clause.*

Let us now consider a generic set of states that is not connected. As we show in Figure 1, this may anyhow correspond to a valid Horn labeling. This, however, does not occur for every security formula. Conversely, every set of states can be written as the intersection of connected sets of states. Therefore, given any security formula, we can represent it as the disjunction of the Horn clauses that are obtained by the sets of connected states.

The purpose of Algorithm 1 is to block the agents that apply for changing variables so to make true a critical formula. Since a critical formula can be made true by making true one of its disjuncts, Lemma 2 can be used directly to prove the following Lemma 3.

More specifically, the greedy blocking works by blocking agents when they apply for the modification of the truth value of a variable, where the blocking condition is: an agent cannot perform an action when this performance brings the system in an insecure state. The synchronization proposed by the algorithm is based on application time: the system simu-

lates the result of performing all (up to the maximum) actions that agents applied for at that instant of time. The system denies the execution to those agents that modify variables involved in the transition of the system into an insecure state. Since this may correspond to more than one combination, the resulting blocked agent set may be larger than needed. We can assume, therefore, without loss of generality, that the algorithm blocks all the agents that applied for modifying variables that bring the system into an insecure state. This assumption is sufficient to employ fruitfully the generalization of Lemma 2 to generic formulas. Remember that $\mathcal{H}(\Phi,\Phi')$ denotes the set of the agents that control variables of one Horn clause in the rewriting $\Phi'$of Φ and bring the system into an insecure state.

Lemma 3. *If no agent in $\mathcal{H}(\Phi,\Phi')$ modifies variables occurring in Φ, and Φ is false, then Φ is false after the modifications.*

Since the agents blocked by the algorithm are all those that bring the system into an insecure state, then every agent controls variables that certainly occur in at least one disjunct of Φ. If we rewrite Φ as a disjunction of Horn clauses (following the standard notion of formula rewriting into disjunction of Horn clauses) $\Phi' = \Phi'_1 \vee \Phi'_2 \vee ... \vee \Phi'_k$, then, by definition of this rewriting, every variable controlled by a that occurs in Φ, occurs in at least one of these disjuncts. If an agent that controls one variable is blocked by our algorithm, then, by definition of the simulation, at least one of the conjuncts in which the variable occurs in Φ' is true. This means that given any pair of secure states s and s', the algorithm never blocks an agent that brings the system directly from s to s'. The extension of this property to paths is proved in the following theorem.

Theorem 3. *The greedy blocking method, Algorithm 1, is complete.*

Soundness and completeness of the deterministic algorithm directly extend to the non-deterministic one.

Theorem 4. *The non-deterministic blocking method, Algorithm 2, is sound and complete.*

Let us call *optimal* a method that blocks the smallest sets of agents to ensure the security of the system. The greedy blocking method guarantees just one of the optimality properties, i.e. security, but it cannot guarantee to block the smallest sets of agents. We thus say that the greedy blocking method is a *sub-optimal solution*. What can further be proved is that the comparison of the solutions computed in the non-deterministic method generates an optimal solution. This is quite obvious, since the solutions computed are all the possible combinations, and thus the best solution is included in this set. What the algorithm does is find the smallest set of agents that need to be blocked.[4]

Theorem 5. *Algorithm 2 computes an optimal solution.*

We consider here the specific problem of blocking underhand attacks as the problem of keeping the security formula false when agents apply for changing variables. The computational complexity of a problem is defined as the cost of the best solution. In this case, we cannot claim that the solution is optimal and therefore we only have an upper bound result.

Theorem 6. *Blockage of underhand attacks is a polynomially solvable problem on deterministic machines.*

Analogously, the next result is a consequence of the results about soundness, completeness and cost of Algorithm 2, again in form of an upper bound.

Theorem 7. *Optimal blockage of underhand attacks is a polynomially solvable problem on non-deterministic machines.*

6 CONCLUSIONS

We have given a first approach to hidden coalitions by introducing a deterministic method that blocks the actions of potentially dangerous agents, i.e. possibly belonging to such coalitions. We have also given a non-deterministic version of this method that blocks the smallest set of potentially dangerous agents. Our two blocking methods are sound and complete, and we have calculated their computational cost.

The starting point of our approach to model multi-agent systems is Coalition Logic (Pauly, 2001; Pauly and Parikh, 2003), a cooperation Logic that implements ideas of Game Theory. Another cooperation logic that works with coalitions is the Alternating-time Temporal Logic, e.g. (Alur et al., 1998). A widely used logic, specifically thought for dealing with strategies and multi-agent systems, is the Quantified Coalition Logic (Ågotnes et al., 2008). A specific extension, also used for agents in multi-agent systems is CL-PC (van der Hoek and Wooldridge, 2005; Troquard et al., 2009), and this is indeed the version of Coalition Logic that we started from.

The notion of hidden coalition is a novelty, and more generally, to the best of our knowledge, no specific investigation exists that deals with security in open systems by means of a notion of underhand attack. The system presented here is a multi-agent one,

[4]There may exist more than one solution with the smallest number of agents blocked. The approach of Algorithm 2 is to compare everything with everything, so the chosen solution is the last examined one.

where we did not discuss how these coalitions are formed or the negotiations that can take place before the creation of the coalitions (Sandholm, 2004; Kraus, 1997). For future work, it will be interesting to consider in more detail the non-monotonic aspects underlying the problem of underhand attacks by hidden coalitions, e.g. to formalize: the mental attitudes and properties of the intelligent agents that compose the system, how agents enter/exit/are banned from a system or enter/exit a hidden coalition, and the negotiations between the agents for establishing the common goal and synchronizing/organizing their actions. In this work, we give a way to protect the system, without making a distinction between the case in which the agents that make the attack are actual members of a coalition or not. If the system is equipped with explicit/implicit coalition test methods, this can make up a significant difference in terms of usefulness of our approach.

A specific analysis of the computational properties of our blocking methods, in particular an analysis of worst, average, and practical cases, will be subject of future work. Results of lower bound for the blocking problem and the optimal blocking problem, and the computational cost of the optimal blocking problem on deterministic machines are in particular important aspects to be investigated.

ACKNOWLEDGEMENTS

The work presented in this paper was partially supported by the FP7-ICT-2007-1 Project no. 216471, “AVANTSSAR: Automated Validation of Trust and Security of Service-oriented Architectures”.

REFERENCES

Ågotnes, T., van der Hoek, W., and Wooldridge, M. (2008). Quantified coalition logic. *Synthese*, 165(2).

Alur, R., Henzinger, T. A., and Kupferman, O. (1998). Alternating-time temporal logic. In *Compositionality: The Significant Difference*, pages 23–60. Springer.

Aumann, R. J. and Hart, S., editors (1994). *Handbook of Game Theory*, volume 2. Elsevier.

Chaum, D. (1981). Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM*, 24(2).

Cristani, M., Karafili, E., and Viganò, L. (2010). Blocking Underhand Attacks by Hidden Coalitions (Extended Version). http://arxiv.org/abs/1010.4786.

Davidsson, P. (2001). Multiagent based simulation: beyond social simulation. *In Multi-Agent-Based Simulation*, 1979:97–107.

Hahn, C., Fley, B., Florian, M., Spresny, D., and Fischer, K. (2007). Social reputation: a mechanism for flexible self-regulation of multiagent systems. *Journal of Artificial Societies and Social Simulation*, 10.

Kraus, S. (1997). Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 94:79–97.

Oravec, V. and Fogel, J. (2006). Coalition logic of propositional control based multi agent system modeling. *Proceedings of IEEE Conference on Mechatronics*, pages 288–291.

Pauly, M. (2001). *Logic for social software*. PhD thesis, Institute for Logic Language and Computation, University of Amsterdam.

Pauly, M. and Parikh, R. (2003). Game logic - an overview. *Studia Logica*, 75(2):165–182.

Rafaeli, S. and Hutchison, D. (2003). A survey of key management for secure group communication. *ACM Computing Surveys (CSUR)*, 35(3):309–329.

Sandholm, T. (2004). Agents in electronic commerce: Component technologies for automated negotiation and coalition formation. *Autonomous Agents and Multi-Agent Systems*, pages 73–96.

Sorniotti, A. and Molva, R. (2010). Secret Interest Groups (SIGs) in social networks with an implementation on Facebook. In *Proceedings of SAC 2010*. ACM Press.

Troquard, N., van der Hoek, W., and Wooldridge, M. (2009). A logic of games and propositional control. In *Proceedings of AAMAS’09*, pages 961–968. ACM Press.

van der Hoek, W., Jamroga, W., and Wooldridge, M. (2005). A logic for strategic reasoning. In *Proceedings of AAMAS’05*, pages 701–708. ACM Press.

van der Hoek, W. and Wooldridge, M. (2005). On the logic of cooperation and propositional control. *Artificial Intelligence*, 164(1-2):81–119.

Wiehler, G. (2004). *Mobility, Security and Web Services*. Publicis Corporate Publishing.

Wooldridge, M. and Jennings, N. (1995). Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2).

Wooldridge, M. and Jennings, N. (1998). Pitfalls of agent-oriented development. In *Proceedings of Agents’98*, pages 385–391. ACM Press.