Model checking detectability of attacks in multiagent systems

Ioana Boureanu
Department of Computing
Imperial College London, UK
ibourean@imperial.ac.uk

Mika Cohen
Department of Computing
Imperial College London, UK
mcohen1@imperial.ac.uk

Alessio Lomuscio
Department of Computing
Imperial College London, UK
a.lomuscio@imperial.ac.uk

ABSTRACT

Information security is vital to many multiagent system applications. In this paper we formalise the notion of detectability of attacks in a MAS setting and analyse its applicability. We introduce a taxonomy of detectability specifications expressed in temporal-epistemic logic. We illustrate the practical relevance of attack detectability in a case study applied to a variant of Kerberos protocol. We model-check attack detectability in automatically generated MAS models for security protocols.

Categories and Subject Descriptors

D.2.4 [**Software/Program Verification**]: Model checking

General Terms

Verification, Security

Keywords

Verification of MAS; Logics of Agency; Logic-based approaches and methods; Privacy, safety and security; Formal models of agency

1. INTRODUCTION

Security is an essential component to several multiagent system (MAS) applications including online trading [23], web-services [9], negotiation systems [19], self-healing systems, etc. Manually validating security requirements is a notoriously difficult and error-prone task. To surpass this difficulty tools have been developed to verify security protocols, to identify possible attacks [1, 2, 3, 4, 18, 10].

While being able to discover attacks automatically is obviously important, protocols that are known to be susceptible to attacks (e.g., IPSec, WEP, GSM, DNS, SSH1, Kerberos, etc.) remain in widespread use at least in some applications. In some cases the cost of protocol redeployment over large multi-agent distributed applications is too high to justify the benefit. Other times hardware, software, or standards constraints make it impossible to deploy other protocols on some applications.

Cite as: Model checking detectability of attacks in multiagent systems, I. Boureanu, M. Cohen, A. Lomuscio, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

If we are prepared to accept that in some cases the system will be compromised, it becomes of paramount importance *to detect* when this happens. Detecting that an attack has happened, or identifying some early signs of an attack being attempted, could be fundamental to limiting the damages to the system. Having the ability to detect attacks could itself be a deterrent for the attacks to be carried out. Even more so if the culprit of the attack could be identified. When are attacks detectable then? In security it is assumed that when an attack is successful the frauded party does not learn an attack has taken place. However, attacks may leave groups of agents with sufficient collective information to infer that an attack has taken place.

Just as verifying whether protocols are secure is a question competently answered by means of automatic tools, it turns out that checking the detectability of attacks is also difficult to check manually.

In this paper we show how attack detectability can be verified automatically in a MAS setting. To fix the technical details, here we focus on a class of widespread security protocols, e.g., authentication and key-establishment protocols and on attacks carried out on them. However our analysis can be extended to higher-level, dedicated protocols, e.g., protocols for web-services, e-voting, e-commerce etc. We only focus on the information-theoretic notion of detectability here (i.e., whether or not a group of agents has sufficient information to deduce an attack has taken place); we leave for further work the actual, protocol-dependent synthesis of communication steps to make this notion explicit.

The rest of the paper is structured as follows. Section 2 provides the required background on the formal MAS framework and logics used, on security protocols, and on a modelling of protocol executions as MAS as per [5]. Section 3 presents a taxonomy of specifications in temporal-epistemic logic formalising attack detectability in a group of agents. In Section 4 we illustrate the practical relevance of the introduced detectability specifications in a case study on KSL, a variant of the Kerberos protocol. Section 5 uses a recent toolkit put forward in [5] to verify automatically attack detectability for KSL and for other well-known security protocols. Finally Section 6 concludes and considers future work.

2. INTERPRETED SYSTEMS, CTLK AND SECURITY PROTOCOLS

Interpreted systems and temporal epistemic logic. The *interpreted systems* (IS) formalism [11] is a semantics for multiagent systems. We summarise its basic definitions. We assume a set $A = \{1, \dots, n\}$ of agents and a special agent called the *Environment*, abbreviated E. We associate to each agent $i \in A$ a set L_i of possible *local states* and a set Act_i of *local actions*. A *protocol function* $P_i : L_i \rightarrow 2^{Act_i}$ defines for each local state l_i the set of actions enabled at l_i. For the environment we associate similar sets L_E, Act_E, and a protocol function P_E. The local transitions of agent i are defined by the *evolution function* $t_i : Act_1 \times \ldots \times Act_n \times Act_E \rightarrow 2^{L_i \times L_i}$. The *evolution function* t_E of the environment is defined in a similar way. To describe the system as a whole we define a set of possible *global states* $G = \prod_{1 \leq i \leq n} L_i \times L_E$, a *joint protocol* $\overline{P} = (P_1, \ldots, P_n, P_E)$, *joint actions* $Act = Act_1 \times \ldots \times Act_n \times Act_E$, a *global evolution function* $\overline{t} = (t_1, \ldots, t_n, t_E)$ operating on global states.

An interpreted system is a tuple $\mathcal{I} = \langle G, \overline{P}, \overline{t}, I_0, V \rangle$, where $I_0 \subset G$ is a set of *initial global states* describing the initialization of the system, and $V : G \rightarrow 2^{AP}$ is a *valuation function* for a set of atomic propositions AP.

The joint protocol and the joint evolution function induce a *transition relation* between any two possible global states. A *(computation) path* is a sequence of global states in which each two adjacent states are in the transition relation.

We associate to each agent i an epistemic *indistinguishability relation* over global states $\sim_i \subseteq G \times G$ such that $g \sim_i g'$ iff $g_i = g'_i$, agent i is in the same local state in both g and g'. In more detail, the relation $\sim_i$ denotes that agent i cannot differentiate between states g and g' as it holds the same information g_i or g'_i at each of the two states. For groups of agents $G \subseteq A$, the indistinguishability relation is extended naturally to $g \sim_G g'$. Global state g is indistinguishable by group G from global state g' iff by using the information of all members of the group, the states (still) cannot be differentiated: $g \sim_G g'$ iff $g \sim_i g'$, for each agent i member of the group G.

We use a logic that combines a fragment of branching–time logic CTL and the modal system $S5_n$ with a knowledge operator K_i for each agent i and a group–knowledge operator D_G for each group G of agents. Formally, our language $\mathcal{L}$ is defined as follows:

$$\varphi, \psi := p \,|\, \neg\varphi \,|\, \varphi \wedge \psi \,|\, K_i\varphi \,|\, D_G\varphi \,|\, AG\,\varphi \,|\, AF\,\varphi \,|\, A(\varphi W \psi),$$

where p ranges over atomic propositions in AP. The knowledge operators K_i and D_G are read "agent i *knows* that" and "the group G of agents *collectively knows* that" respectively. The CTL quantifier over paths A is read "*along every* computation path". The path–specific quantifiers G, F and W are read "*always*", "*eventually*" and "*(weak) until*" or "*unless*" respectively.

We give some examples of interpreting formulae in the language $\mathcal{L}$ over the IS semantics. Consider an arbitrary IS model $\mathcal{I}$. For instance, satisfaction under $\mathcal{I}$ of the CTL formula $A(\phi\, W\, \psi)$ in $\mathcal{L}$ is as follows: $(\mathcal{I}, g) \models A(\phi\, W\, \psi)$ iff for all paths $g_0, g_1, \ldots$ in $\mathcal{I}$ with $g = g_0$ it holds that $(\mathcal{I}, g_i) \models \psi$ and $(\mathcal{I}, g_j) \models \phi$ for some $i \geq 0$ and for all $j \leq i$ or, it holds that $(\mathcal{I}, g_i) \models \phi$ for all $i \geq 0$. Another example of satisfaction under $\mathcal{I}$ is that of epistemic formula $D_G\phi$ in $\mathcal{L}$, as follows: $(\mathcal{I}, g) \models D_G\phi$ iff for all states g' such that $g \sim_G g'$ it holds that $(\mathcal{I}, g') \models \phi$.

Security protocols. A security protocol specifies a sequence of data exchanges aimed at delivering some security related goal through the use of cryptographic primitives. Security protocols are often described using informal *Alice&Bob notation* [22]. To illustrate, consider the Alice&Bob notation for the well–known Needham Schroeder Public Key (NSPK) protocol [20]:

1. $A \rightarrow B : \{A, N_A\}_{pub(B)}$
2. $B \rightarrow A : \{N_A, N_B\}_{pub(A)}$
3. $A \rightarrow B : \{N_B\}_{pub(B)}$

Two protocol *roles* are implied by the NSPK description: the role of A and the role of B. A participant playing the role of A encrypts its name and its nonce[1] N_A with the public key $pub(B)$ of a B–role participant, and sends the resulting encryption $\{A, N_A\}_{pub(B)}$ to this B–role participant. The participant playing the role of A then waits for a message encrypted with its own public key $pub(A)$, containing its own nonce N_A and a new nonce N_B. Following this, the participant proceeds in his A–role by encrypting the received nonce N_B with the public key $pub(B)$ and sending this encryption back to N_B's originator, the B–role player. Dually to an A–role, the first action of a participant playing a B–role is awaiting for a message encrypted with its own public key $pub(B)$, etc.

A protocol is equipped at design time with a number of *security requirements*. These depict the purpose of the protocol: authentication, secrecy, anonymity, etc. For example, a custom authentication requirement for NSPK is that when the participant playing the B–role has performed all its three steps it knows with whom it shares values N_A and N_B.

Interpreted system protocol–model. The Alice&Bob notation describes the sequence of messages exchanged during a single protocol session executed between honest participants. In contrast, participants in deployed protocols may engage simultaneously in multiple, parallel sessions with different interlocutors. In more detail, the same identity may be simultaneously a participant in several sessions, playing a distinct role in each, manipulating different data, multiple keys etc. Moreover, it is assumed that all deployed protocols are executed in a hostile environment. In order to subvert the protocol, an intruder can always intercept messages, manipulate data (e.g., encrypt, decrypt, sign and generate nonces), challenge participants to engage in multiple sessions and to persue certain actions etc.

Given the above, we view multisession executions of protocols as a multiagent system. Each *agent* is characterized by his *identity*, the principal whom he represents, and by his *role*, the protocol role that it plays. A model for a possible NSPK multisession execution includes two agents representing principal *alice* and two agents representing principal *bob*; one of the agents identified as *alice* plays the A–role while the other agent of *alice*'s plays the B–role. Similarly for *bob*'s agents. Along the lines of [5], we formalise this MAS model for protocol executions by associating an interpreted system semantics to it:

- The local state of an agent contains the variables that appear in the role played by the agent, e.g., variables A, B, N_A and N_B for the A-role of NSPK protocol. In addition, the local state includes a counter variable *Step* which indicates at what step in the protocol execution the agent is.

- The local protocol of an agent is defined according to the underlying role of the agent: at a certain step, the protocol selects an action either of sending or of receiving a message.

[1]"number once used", i.e., random, unpredictable value.

- The local evolution function specifies the update of the local state as messages are sent and received. For example in the case of NSPK, if agent ag is playing the role of A and ag receives the encryption $\{(v_1, v_2)\}_{v_3}$ at *step* 2, then her evolution function first dictates the check whether v_3 received is indeed her public key, then the check whether v_1 received is indeed the value previously sent by ag as N_A. If these checks succeed, the evolution function sets the value of ag's local variable N_B to the value v_2 received and increments her local *Step* variable.

- The Environment encodes a Dolev-Yao [7] intruder that controls the communication medium: the intruder can intercept and insert messages on the medium, but can only perform cryptographic operations (encrypt, decrypt, sign, etc.) if it has the relevant keys.

Security requirements and attacks. Security requirements are most often formalised as simple invariants (state predicates). In this paper, we assume that each security requirement of interest appears as an atomic proposition p in the interpreted system protocol model, but we leave the interpretation open. We emphasise however that the detectability specifications introduced in the next section apply equally to more complex, temporal–epistemic requirements for privacy, receipt-freeness, etc.

`Authentication requirement.` A standard formulation for an authentication requirement is: whenever an agent i has performed (at least) N execution steps, there is a different agent j that agrees with agent i on the values of variables in a selected set Ξ

$$auth@N(i, \Xi) =_{df} i.Step \geq N \rightarrow \bigvee_{j} agree(i, j, \Xi)$$

where j ranges over agents different from i and $agree(i, j, \Xi)$ abbreviates $\bigwedge_{X \in \Xi} (i.X = j.X)$. We will implicitly assume that Ξ includes variables for the intended communication partners. For example, for NSPK $auth@3(i, \{N_A\})$ abbreviates $auth@3(i, \{A, B, N_A\})$. This denotes that once all three execution steps of agent i have been completed, the agent shares the value $i.N_A$ with its intended communication partner.

An *attack* against a security requirement p is an execution trace that refutes p. We say that an *attack on* p has occurred whenever p fails in the above sense.

`Attack on authentication.` An attack on the authentication requirement $auth@N(i, \Xi)$ has occurred if and only if there is an execution where agent i has completed N steps and there is no other agent j that agrees with i on the variables Ξ:

$$i.Step \geq N \wedge \neg \bigvee_{j} agree(i, j, \Xi)$$

Intended goals and pre–goals. For each protocol a set of security requirements is explicitly specified at design time. These stipulate the *intended goals* of the protocol. In this sense, an attack on any of these is an attack on the protocol. Typically the intended goals of a protocol concern the final state of agents in some execution, e.g., the requirement $auth@3(i, \{N_A, N_B\})$ illustrated above for NSPK.

However, intended goals are achieved gradually throughout a protocol execution. If an intended goal holds at a final step of an agent in an execution, it is expected that meaningful parts of that goal held at some intermediate steps of the agent, i.e., previously in that execution. We use *pre-goals* to denote (implicit) requirements that express parts of intended goals considered at intermediate execution steps. We illustrate below a possible derivation method for pre–goals given the protocol description and the intended goals.

`Derivation of pre-goals.` Assume the set of intended goals includes the authentication requirement $auth@N(i, \Xi)$. We derive a pre–goal $auth@N'(i, \Xi')$ for each $N' \leq N$ and each subset $\Xi' \subseteq \Xi$ of variables which are set in the local states of agent i at step N'. For example, assume that $auth@3(i, \{N_A, N_B\})$ is given as an intended goal for NSPK. Then $auth@1(i, \{N_A\})$ is a pre–goal for the protocol derived as above. It denotes that once the first step of agent i has been completed, the agent is expected to share the nonce N_A with its intended communication partner.

We will assume that if a pre–goal fails it is impossible that the protocol will eventually complete successfully. In more detail, for a protocol with the intended goals Γ and one pre–goal p we assume that:

$$AG\,(\neg\, p \rightarrow\ AG\,(end \rightarrow \neg\Gamma)) \tag{1}$$

where *end* says that agents have completed all their protocol steps, and $\neg\Gamma$ says that at least one requirement in Γ fails. In light of property (1) it seems reasonable to consider also failures of pre–goals as protocol failures.

3. DETECTABILITY OF PROTOCOL FAILURES

It is not uncommon to find insecure protocols deployed in applications. For instance, IPSec, WEP, GSM, DNS, SSH1, Kerberos have all been shown to be susceptible to attacks, but are still widely used. Often, the cost of updating already deployed systems might outweigh the risk of attacks. Once a security protocol is found to be open to attacks a new question arises: are attacks *detectable*? In security it is assumed that an attack on a protocol leaves the agent subject to the assault without sufficient information to determine an attack has taken place. However, attacks may leave the group G of agents with sufficient collective information to infer an attack has taken place.

The relevant group G of information-sharing agents depends on the particular scenario. The most natural candidate is a group of agents that represent the same principal. This group models the concurrent protocol sessions run by the same user on the same machine. But other groups may also be of interest. For instance, one might wish to consider only the subgroup of agents that represent the same principal and play the same protocol role. The group may model a software client that collects information from different sessions (e.g., a user running the SSH protocol on the client side in two different terminals). In fact, one may even consider a group of agents that represent different principals. These groups are most relevant from the point of view of a protocol attacker. The attacker would be interested in a guarantee that he will remain undetectable no matter what information is subsequently exchanged between honest participants. Indeed, attacks that are in principle detectable by a certain group can lead to retaliations undesired by the attacker. As argued in [12], this fact can act as deterrence.

In this section we use temporal–epistemic logic to formalise subtle differences in the ability of a group G to detect attacks and other protocol failures. Are attacks eventually detectable on all possible future paths, or just on some? Are attacks detectable without the intruder knowing this? Can attack–related failures be immediately detected and can attacks therefore be detected prior to their actual completion? In future sections, the meaning of these specifications will be explicitly denoted in MAS environments and their corresponding formulae will be checked against formal MAS models for protocol executions.

Attack detectability.

We consider attack detectability specifications of the form:

$$AG\,(\beta \rightarrow \Box\,\beta) \tag{2}$$

where the *modality* $\Box$ is generated by the grammar:

$$\Box \quad ::= \quad D_G \mid AF\,D_G \mid EF\,D_G \mid \neg\Box$$

and the *condition* β expresses that a "bad" state has been reached. In the most basic case is $\beta ::= \neg p$ and it states that there has been an attack on an intended security goal p. Below we consider more complex conditions β.

We say that condition β is *instantly detectable by group G* if whenever β holds the group G knows this:

$$AG\,(\beta \rightarrow D_G\,\beta) \tag{3}$$

We say that β is *eventually detectable* if whenever β holds the group G will eventually know this:[2]

$$AG\,(\beta \rightarrow AF\,D_G\,\beta) \tag{4}$$

We say that β is *possibly detectable* if whenever β holds it is possible that the group G will eventually know this:

$$AG\,(\beta \rightarrow EF\,D_G\,\beta) \tag{5}$$

We say that β is *instantly undetectable* if whenever β holds the group G does not know this:

$$AG\,(\beta \rightarrow (\neg D_G)\,\beta) \tag{6}$$

We say that β is *possibly undetectable* if whenever β holds it is possible that the group G will never know this:

$$AG\,(\beta \rightarrow (\,\neg(AF\,D_G)\,)\,\beta) \tag{7}$$

We say that β is *forever undetectable* if whenever β holds the group G will never know this:

$$AG\,(\beta \rightarrow (\,\neg(EF\,D_G)\,)\,\beta) \tag{8}$$

Specifications (3) - (8) exhaust the modalities $\Box$ up to logical equivalence. Clearly, the logical–strength decreases in specifications (3) to (5) while it increases in (6) to (8). Moreover, for detectability specifications (3) - (5) the logical–strength decreases with larger groups G. On the other hand, for (un)detectability specifications (6) - (8) the logical–strength decreases with more inclusive groups G.

[2]It is a customary assumption in the Dolev–Yao model that the intruder is able to block messages. However, this is not reasonable in all protocols, e.g., wireless protocols. Thus, we sometimes restrict the path quantifiers to paths that are *fair* in the sense that agents complete all their protocol steps.

Simple attack–detectability. The most basic conditions β are of type $\beta ::= \neg\,\Gamma$. This denotes that some security requirement p from a selected set Γ of requirements fails. Note that in specifications we identify the set Γ with the conjunction of the requirements included in the set. The detectability schema (2) then becomes:

$$AG\,(\neg\,\Gamma \rightarrow \Box\,\neg\,\Gamma)$$

stating that attacks on the security requirements Γ are instantly/eventually/possibly detectable or instantly/possibly/forever undetectable by the group G.

The selected set Γ of requirements might include only the intended security goals of the protocol being analysed. However, it may be of interest to include also pre–goals in Γ; failure of an such goal signals that the intruder is mounting an attack and the protocol will not complete successfully.

Culprit–detectability. Some attacks on security protocols require that the intruder is an "insider": the intruder knows from the outset the private keys and/or signatures of some corrupt principal. If honest agents can detect the identity of this corrupt principal then retaliation actions can be more precisely targeted, strengthening the deterrence.

We consider therefore instances of (3) - (8) in which the condition β states that a certain principal identity v is corrupt: $\beta ::= \neg\,\Gamma \wedge \mathit{corrupt}(v)$. The atomic proposition $\mathit{corrupt}(v)$ holds if the intruder started the current run knowing the private key of the principal with identity v. [3]

In particular, the condition $(\neg\,\Gamma \wedge \mathit{corrupt}(v))$ is possibly detectable if:

$$AG\,(\neg\,\Gamma \wedge \mathit{corrupt}(v) \rightarrow EF\,D_G\,(\neg\,\Gamma \wedge \mathit{corrupt}(v))) \tag{9}$$

i.e., whenever requirements Γ are attacked and v is corrupt it is possible for group G to know this eventually.

Nested attack–detectability. It seems reasonable to assume that possible culprit detectability (9), or even the weaker possible attack–detectability (5) for $\beta ::= \neg\,\Gamma$ achieve an adequate deterrence in many possible scenarios. Moreover, the perspective of being able to know whether his attack have been detected can influce future actions of the intruder. Namely, the attacker could prepare for possible retaliations. In other words, if the attacker is assured that he will always know when he is detectable, he may be more willing to conduct attacks.

Thus, we consider instances of schema (2) where the "bad" states defined by β are those states in which an attack is detectable:

$$\beta \quad ::= \quad \Box\,\neg\,\Gamma \mid \Box(\neg\,\Gamma \wedge \mathit{corrupt}(v))$$

where the attack–modality $\Box$ includes no negations. Intuitively, β defines states that are "bad" from the point of view of an attacker (i.e., his actions are detectable). To illustrate, consider the states given by $\beta ::= D_G\neg\,\Gamma$, "bad" states in which an attack on requirements Γ is detectable by a group G of "honest" agents. Then, condition β itself is instantly detectable by another group G' of "hostile" agents if:

$$AG\,(D_G\,\neg\,\Gamma \rightarrow D_{G'}\,D_G\,\neg\,\Gamma)$$

[3]For ease of presentation, we assume that there is at most one corrupt principal in any run of the system $\mathcal{I}$. Moreover, we assume that the name of the corrupt principal is randomised in each run. In this way the corrupt identity is initially unknown to honest agents.

i.e., whenever the "honest" group G knows that there is an attack on requirements Γ, the "hostile" group G' knows that the "honest" group G knows this. Another more complex instance of schema (2) views the states given by $\beta ::= AF\, D_G \neg\, \Gamma$. These are "bad" states in which an attack on requirements Γ is eventually detectable by the group G of "honest" agents. Then, condition β is instantly detectable by another group $G^{'}$ of "hostile" agents if:

$$AG\,(AF\, D_G \neg\, \Gamma \rightarrow D_{G'}\, AF\, D_G\, \neg\, \Gamma)$$

i.e., whenever it is the case that the "honest" group G will eventually know that there is an attack on requirements Γ, the "hostile" group $G^{'}$ knows that the "honest" group G knows this.

Above we illustrated on the "hostile" group G' being able to detect states that are "bad" from their point of view. However, it is of interest to consider whether the "hostile" group G' can also detect states that are "good" from their perspective. For instance, consider the states in which attacks are undetectable by the "honest" group G. To this end we assume conditions β with a negative attack–modality $\Box$ (i.e., a modality with an odd number of negations). For example, consider the states given by the condition $\beta ::= (\,\neg(EF\, D_G)\,)\neg\, \Gamma$, "good" states in which an attack on requirements Γ is forever undetectable by the group G of "honest" agents. Then, this condition β is eventually detectable by the group $G^{'}$ of "hostile" agents if:

$$AG\,(\,\neg(EF\, D_G)\,\neg\, \Gamma \rightarrow AF\, D_{G'}\ \neg(EF\, D_G)\ \neg\, \Gamma)$$

i.e., whenever it is the case that the "honest" group G will never know that there is an attack on requirements Γ, the "hostile" group G' eventually knows that the "honest" group G will never know this.

Attack–launch detectability.

It might be the case that neither attacks on intended goals Γ nor attacks on pre–goals Γ' are detectable, not even possibly detectable. Nevertheless, attacks on the intended goals Γ might be observable in a weaker sense: an attack on the intended goals Γ cannot be mounted without the group learning in the meanwhile that some pre–goal in $\Gamma^{'}$ fails.

We say that *attacks on pre–goals* $\Gamma^{'}$ *signal attacks on intended goals* Γ *to group* G if requirements Γ hold at least until the group G knows that requirements $\Gamma^{'}$ fail:

$$A\,(\Gamma\ W\ D_G \neg\, \Gamma') \tag{10}$$

Note that attack–launch detectability (10) does not reduce to an attack detectability specification (2). In particular (10) may hold even if attacks on the pre–goals $\Gamma^{'}$ are not *possibly detectable* (5) by the group G.

If attacks on the pre–goals Γ' signal attacks on the intended goals in Γ we can conclude that the flawed protocol can be "patched" with the epistemic test $D_G \neg\, \Gamma'$. For instance, a possible fix is for agents in G to abort as soon as $D_G \neg\, \Gamma'$. Several flawed protocols have been "patched" in a similar way [17], but with non-epistemic tests that compare values across agents representing the same principal. Finding the appropriate non-epistemic tests can be non–trivial. By contrast, the relevant set $\Gamma^{'}$ of pre–goals can be derived automatically with little cost from the protocol description, as illustrated in Section 2.

4. CASE STUDY: KSL

In this section we illustrate a case study on detectability of assaults against KSL [14], a variant of the Kerberos protocol. The next section reports on a framework for automatic analysis of detectability specifications. Some of the results in this section were provided by the automatic methodology aforementioned.

The KSL protocol was designed with two distinct levels. In the first level, a *key establishment level*, two parties A and B use a trusted server S to establish a session key Kab and a ticket. We refer to [14] for details. The key and ticket are employed in the second level in order to mutually authenticate A and B in three simple message exchanges. The second level is a *repeated authentication level* since it can be run several times, until the ticket expires.

```
1. A -> B : Ma, {Tb,A,Kab}Kbb
2. B -> A : {Ma}Kab, Mb
3. A -> B : {Mb}Kab
```

In step 1 principal A sends to B a nonce challenge Ma and the ticket $\{Tb, A, Kab\}Kbb$ established during the first level. This ticket contains a timestamp Tb, the identity A and the session key Kab also established in level one. All these are locked with Kbb, a key known only by B. If the timestamp Tb inside the ticket received by B has not expired yet, then B responds in step 2 by encrypting the nonce Ma using the preestablished session key Kab. B also sends along a nonce challenge Mb of her own, which A then returns encrypted with the session key Kab in step 3.

The purpose of KSL's second–level is to ensure that when an agent playing the role of B completes all its steps it then shares the nonces Ma and Mb with its intended communication partner. Therefore, we consider the intended authentication goal $auth@3(ag, \{Ma, Mb\})$ for each agent ag playing the B–role in the second-level.

The Hwang attack.

The following attack on the repeated authentication level is due to Hwang [13]:

```
i.1.   I(A) -> B : Ma, {Tb,A,Kab}Kbb
i.2.   B  ->  I(A): {Ma}Kab, Mb
ii.1.  I(A) -> B :  Mb, {Tb,A,Kab}Kbb
ii.2.  B -> I(A) :  {Mb}Kab, Mb'
i.3.   I(A)  -> B :  {Mb}Kab
```

This attack assumes a first–level session completed between principals *alice* and *bob*, playing an A–role and a B–role respectively. On these grounds, the intruder impersonates *alice* and initiates a second–level session i with *bob*. The intruder has previously intercepted a B–role ticket in the precedent first–level communication and in step $i.1$ sends this ticket together with a nonce Ma to *bob*. In step $i.2$, *bob* responds according to his honest B–role by sending the nonce Ma encrypted with the session key Kab and the challenge–nonce Mb to his assumed A–role interlocutor. Instead of completing this session, the intruder impersonates *alice* again and initiates yet another second–level session ii with *bob*. In this second session ii, the intruder uses *bob* as an oracle to encrypt his own challenge–nonce Mb from the initial i session. The intruder finally inserts the oracled encryption $\{Mb\}Kab$ back into session i as if it were coming from *alice*, thus completing *bob*'s initial session in step $i.3$.

Informally, *bob* is fooled into believing that in session i he is sharing values Ma and Mb respectively with *alice*, while in fact he has been interacting with the intruder impersonating *alice*. Formally, KSL's authentication goal $auth@3(ag, \{Ma, Mb\})$ fails for *bob*'s agent ag engaged in session i.

Following the formalism in [5] and summarised in Section 2, we consider a MAS model $\mathcal{I}$ to formalise KSL multi-session executions.

Attack–launch detectability. On the MAS model $\mathcal{I}$, we can show that principal *bob* can detect preparations for the attack prior to its actual completion. More precisely, attacks on the pre–goal of authentication upon Ma at step 1 signals to *bob* attacks on the intended goal of authentication upon Ma and Mb at step 3. Formally, the intended authentication goal holds at least until *bob*'s agents collectively have sufficient information to infer that in one session, one of these agents received a replayed nonce for an expectedly fresh Ma. In other words, the corresponding attack–launch detectability formula that holds on the IS model $\mathcal{I}$ is:

$$A\,(auth@3(\{Ma, Mb\})\ W\ D_{bob}\,\neg\,auth@1(Ma)) \qquad (11)$$

where D_{bob} is the distributed knowledge modality D_G for the group G of agents representing *bob* and playing the B–role in the second level, $auth@3(\{Ma, Mb\})$ abbreviates the the set $\{auth@3(ag, \{Ma, Mb\}) \mid ag \in G\}$ of intended goals, and $auth@1(Ma)$ operates analogously for the pre–goals. To explain, the group G of *bob*'s agents can compare the value of Mb in each group member with the value of Ma in each other group member; there has been an attack on $auth@1(Ma)$ if two values collide. Particularly, the epistemic test $D_{bob}\,\neg\,auth@1(Ma)$ can be reduced to the non–epistemic test:

$$\exists ag, ag' \in G \mid ag.Mb = ag'.Mb$$

which in turn translates (11) into a purely temporal property. However, the reduction of $D_{bob}\,\neg\,auth@1(Ma)$ to a non-epistemic test is specific to a particular protocol and may be non-trivial to determine. By contrast, attack–launch specifications like (11) can be generated automatically as explained in Section 3 and later shown in Section 5.

Eventual attack–detectability. Despite attack–launch detectability specification (11), attacks on the intended goals $auth@3(\{Ma, Mb\})$ are not instantly detectable by *bob* in the sense of specification (3). In the MAS system $\mathcal{I}$, the attack–trace $i.1$ – $i.3$ for the Hwang attack above is indistinguishable to *bob* from an execution trace in which the intruder does not impersonate *alice*. This counterexample trace of system $\mathcal{I}$ is shown below:

`Counterexample E1`
Trace of $\mathcal{I}$, counterexample for $D_G \neg auth@3(\{Ma, Mb\}))$

```
i'.1   A -> B: {Tb,A,Kab}Kbb, Ma
i'.2  B -> A: Mb, {Ma}Kab
ii'.1  I(A) -> B:  {Tb,A,Kab}Kbb, Mb
ii'.2   B -> I(A): Mb', {Mb}Kab
i'.3  A -> B: {Mb}Kab
```

We briefly explain the above execution exhibited in the model $\mathcal{I}$. As in the Hwang attack, the nonce Mb of *bob*'s agent from the first session i' is replayed to the agent representing *bob* in another session ii'. The trace is therefore indistinguishable to *bob* from the Hwang attack. Moreover, note that in session i' of the counterexample–execution principal *bob* authenticates himself correctly upon values Ma and Mb with *alice*.

Nonetheless, we can show that attacks on the intended goals $auth@3(\{Ma, Mb\})$ are eventually detectable by *bob*. In other words, the following attack–launch detectability formula holds on the IS model $\mathcal{I}$:

$$AG\,(\neg\,auth@3(\{Ma, Mb\}) \rightarrow AF\,D_{bob}\neg\,auth@3(\{Ma, Mb\}))$$

where the AF–modality quantifies only over execution paths that are fair in the sense that each agent eventually completes all its steps.

We briefly explain the eventual detectability by *bob* in the above. Assume that *bob*'s agent used as an oracle in session ii of the Hwang attack has completed his execution. By comparing the nonce received in this session with the nonce generated in his first session i by another agent of his, principal *bob* can infer than the nonce received did not originate from *alice* as it purports to do. In other words, *bob*'s agents collectively know that *bob* has completed session ii without agreeing with *alice*, i.e., *bob* knows that the intended authentication goal has been attacked.

The Lowe attack.

The discussion above assumes that a KSL principal does not engage in the second level prior to completing all the steps of the first level. However, KSL has also been interpreted [17] as a protocol that allows a principal playing the B–role to engage in a second–level session as soon as a ticket has been established, but before actually completing the final challenge–response step of the protocol's first–level. Again we refer to [14] for details.

As shown by Lowe, this permits more attacks against KSL. For instance, *alice* can be fooled into generating a B–role ticket $\{Ta, bob, Kab\}Kaa$ and *bob* also be fooled into generating another B–role ticket $\{Tb, alice, Kab\}Kbb$ with both tickets based on the same session key Kab. These can then be used by the intruder for an impersonation attack on the second level:

```
i.1.  I(A) -> B :  {Tb,A,Kab}Kbb, Ma
i.2.  B -> I(A) :  Mb, {Ma}Kab
ii.1.  I(B) -> A : {Ta,B,Kab}Kaa, Mb
ii.2. A -> I(B) :  Ma', {Mb}Kab
i.3.  I(A) -> B :  {Mb}Kab
```

The attack begins like the Hwang attack above: the intruder impersonates *alice* to enter second–level session i with *bob*. However, in order to encrypt the nonce challenge Mb with the session–key Kab the intruder does not use another agent representing *bob* as oracle like in Hwang attack. Instead, in step $ii.1$ the intruder engages an agent representing *alice* and playing the B–role. The intruder finally completes session i with *bob* in step $i.3$ by forwarding nonce Mb as previously encrypted by *alice* in $ii.2$. We refer to the [13, 6] for details. Like in the Hwang attack the agent representing *bob* in session i is fooled into believing that he is sharing the values Ma and Mb with *alice*.

We consider now a MAS model $\mathcal{I}$ formalising KSL multisession executions upon the relaxed interpretation in [17] described above, i.e., possible initiation of second–level KSL runs without full completion of prior KSL first level sessions.

Instant attack–detectability. On the MAS model $\mathcal{I}$ we can show that the Lowe attack is instantly detectable by a subgroup of agents representing principals *bob* and *alice* together. More precisely, the group G of all second–level agents can instantly detect attacks on the intended authentication goals. In other words, the following detectability formula holds on the IS model $\mathcal{I}$:

$$AG\,(\neg auth@3(\{Ma, Mb\}) \rightarrow D_G\, \neg auth@3(\{Ma, Mb\}))$$

In other words, by comparing values between group members the group can (trivially) deduce whether any one group member is fooled into completing the second-level without agreeing with any other agent.

Possible attack–undetectability. On the MAS model $\mathcal{I}$ we can show that the Lowe attack is possibly undetectable by principal *bob* alone. More precisely, attacks on the intended goals $auth@3(\{Ma, Mb\})$ for *bob*'s agents are possibly undetectable by the common effort of all *bob*'s agents. In other words, the following (un)detectability formula holds on the IS model $\mathcal{I}$:

$$AG\,(\neg auth@3(\{Ma, Mb\}) \rightarrow \neg AF D_{bob}\, \neg auth@3(\{Ma, Mb\}))$$

Under the MAS model $\mathcal{I}$ if the intruder conducts the Lowe attack, then the resulting trace is indistinguishable to the group of *bob*'s agents from a trace in which there is no attack on the intended authentication goal. Unlike during the Hwang attack, in the Lowe attack the challenge nonce Mb of one of *bob*'s agents is replayed to an agent representing *alice* rather than back to one of *bob*'s agents. Thus, there is no way for *bob* to observe this replay even at later stages.

However it is not the case that attacks on the intended goals $auth@3(\{Ma, Mb\})$ are forever undetectable by *bob* in the sense of (8). Since the Hwang attack is possible also under this relaxed interpretation of KSL, there exists a path where *bob*'s undetection is eventually refuted.

Nested detectability. On the MAS model $\mathcal{I}$ we can show that even though attacks on the intended goals are not always undetectable by *bob*, whenever they are undetectable the intruder knows this is so. In other words, the following (nested) detectability formula holds on the IS model $\mathcal{I}$:

$$\begin{aligned} &AG\,(\neg D_{bob}\, \neg auth@3(\{Ma, Mb\}) \rightarrow \\ &\qquad D_I\, \neg D_{bob}\, \neg auth@3(\{Ma, Mb\})) \end{aligned}$$

stating that the "good–from–the–intruder–point–of–view" states given by $\neg D_{bob}\, \neg auth@3(\{Ma, Mb\})$ are instantly detectable by the intruder.

5. EXPERIMENTS

In this section we report on an automatic approach to checking attack detectability in a variety of protocols. The results here presented generalise the ones discussed for KSL in the preceeding section.

Tools employed. PD2IS (Protocol Descriptions to Interpreted Systems) [5] is an open source compiler from CAPSL [8] protocol descriptions into ISPL (Interpreted System Programming Language), the input language to the model checker MCMAS [15]. MCMAS is a BDD-based symbolic model checker for multiagent systems supporting temporal epistemic specifications.

In addition to CTLK security requirements which PD2IS already supported, we extended the tool to generate CTLK formulae encoding attack, attack-launch and culprit (un)detectability specifications described in section 3.

By doing so, we have produced an extended toolkit [21] that automatically analyses attack detection properties for protocols given in the CAPSL language.

Experiment design and results. We selected a set of well known authentication and key establishment protocols from the SPORE repository [6]. Given a CAPSL protocol description we used PD2IS to generate a corresponding ISPL file for each instantiation scenario with four or fewer sessions; each generated ISPL file was then passed to MCMAS for verification. For each protocol we selected the models that exhibited attacks and analysed the output of MCMAS for what concerns satisfiability of the detectability specifications. Some of these results are summarised in Table 5.

		Det	Undet	Launch	Culprit
Scenario	Groups				
NSPK	a	F	possible	F	possible
$a^1 \rightarrow I$	b^1	F	forever	F	F
$a^1 \leftarrow b^1$	a^1, b	eventual	possible	T	eventual
$a^2 \rightarrow b^1$	a, b	eventual	possible	T	eventual
WMF	a	F	forever	N/A	N/A
$a^1 \rightarrow S^1$	b	F	forever	N/A	N/A
$S^1 \rightarrow b^1$	a, b	eventual	instant	N/A	N/A
$I \rightarrow b^1$	S, b	eventual	possible	T	eventual
A. S-RPC	a	instant	F	F	N/A
$a^1 \rightarrow b^1$	a^2	F	possible	N/A	N/A
$a^2 \rightarrow b^2$	b	F	forever	F	N/A
	a^2, b^2	instant	F	T	N/A
KSL	a	F	forever	F	N/A
$a^1 \xrightarrow{1} b^1$	a^1	F	forever	F	N/A
$a^1 \xrightarrow{1} S$	b	F	possible	T	N/A
$a^1 \xleftarrow{2} b^1$	a^1, b	eventual	instant	T	N/A
$a^1 \xleftarrow{2} b^2$	a^2, b	F	possible	T	N/A
$a^2 \xrightarrow{2} b^3$	a, b	eventual	F	T	N/A

Table 1: Attack detectability results.

In the table the first column denotes the scenario considered (see below); the second column denotes the detectability group G considered; the third column reports on whether attacks were found to be detectable instantly, eventually, possibly, or never detectable ("F") by the group G considered; the forth column reports on whether attacks were found to be undetectable instantly, possibly, or forever detectable; the fifth column states the satisfaction of detectability of attacks on pre-goals where it applies; the last column reports the results for satisfaction of culprit detectability formulae. "N/A" signifies that the corresponding detectability specification does not apply for the corresponding protocol or the corresponding group.

In more detail, in the first column the notation $a^1 \rightarrow I$ indicates that the NSPK scenario considered includes an agent a^1 representing principal *alice* in session 1 communicating with a corrupt insider I. Analogously, $a^2 \rightarrow b^2$ indicates that the scenario also includes an agent a^2 representing principal *alice* in session 2 communicating with an agent representing *bob*. The notation $\xrightarrow{i}$ in the KSL scenario denotes a communication direction within level i of the protocol. [4] The

[4]We assume the classical interpretation of level interleaving in KSL (which does not allow for the Lowe attack).

letter S in the WMF scenario represents the server identity. The letters a and b in the second column denote the groups of all agents representing *alice* and *bob* respectively.

The table does not report the verification time of the toolkit. These were found not to be problematic for the protocols considered whenever the number of instances was kept to a practically useful number. As an example, checking NSPK [16] with 4 agents in a communication setting enabling the Lowe attack against 218 specifications took approximately 21 seconds on an Intel Core 2 Duo clocked at 2.26GHz with 2.9 GB of memory, running Linux kernel 2.6.27.7. The corresponding number of reachable global states was in the region of 10^3.

6. CONCLUSIONS

In this paper we have formalised the notion of detectability of attacks against security protocols in a MAS setting. We showed the applicability of different flavours of detectability on a MAS model for KSL, a variant of the Kerberos protocol. We reported on the results of model-checking automatically generated MAS models for security protocols against a taxonomy of detectability formulae (generated inline with the models). The positive results demonstrate that detectability of attacks in MAS inspired security applications is a viable concept that can be used in more complex protocols. At a technical level, while our discussion was grounded on authentication and key–establishment protocols, much can be extended to attacks on higher-level protocols, such as e-voting.

Acknowledgments. The research described in this paper is partly supported by EPSRC funded project EP/E035655. The authors would like to thank Mads Dam for valuable comments on earlier drafts of this paper.

7. REFERENCES

[1] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. Drielsma, P. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA tool for the automated validation of internet security protocols and applications. In *Proc. of the 17th International Conference on Computer Aided Verification*, pages 281–285, 2005.

[2] G. Bella, C. Longo, and L. Paulson. Verifying second-level security protocols. In *Theorem Proving in Higher Order Logics*, pages 352–366. Springer, 2003.

[3] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proc. of the 14th IEEE Computer Security Foundations Workshop*, pages 82–96. IEEE Computer Society Press, 2001.

[4] M. Boreale and M. Buscemi. A framework for the analysis of security protocols. In *Proc. of the 13th International Conference on Concurrency Theory*, pages 483–498, 2002.

[5] I. Boureanu, M. Cohen, and A. Lomuscio. A compilation method for the verification of temporal-epistemic properties of cryptographic protocols. In *Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security*, 2009.

[6] LSV ENS Cachan. Security Protocols Open Repository. `http://www.lsv.ens-cachan.fr/spore`.

[7] D.Dolev and A.Yao. On the security of public-key protocols. *IEEE Transactionson Information Theory 29*, pages 198–208, 1983.

[8] G. Denker and J. Millen. CAPSL integrated protocol environment. In *In Proc. of DARPA Information Survivability Conference*, pages 207–221. IEEE Computer Society, 2000.

[9] J. Estevez-Tapiador. Moving web services to the secure side. *IEEE Distributed Systems Online*, 5(1), 2004.

[10] T. Fábrega, J. Herzog, and J. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(1):191–230, 1999.

[11] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge.* MIT Press, 1995.

[12] G.Bella, S.Bistarelli, and F.Massacci. Retaliation against protocol attacks. *Journal of Information Assurance and Security*, 3:89–102, 2008.

[13] T. Hwang, N. Lee, C. Li, M. Ko, and Y. Chen. Two attacks on neuman-stubblebine authentication protocols. *Information Processing Letters*, 53(2):103–107, 1995.

[14] A. Kehne, J. Schőnwālder, and H. Langendőrfer. Multiple authentications with a nonce-based protocol using generalized timestamps. In *International Conference on Computer Communications*, 1992.

[15] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. *Proc. of the 21th International Conference on Computer Aided Verification*, 5643, 2009.

[16] G. Lowe. An attack on the Needham-Schroeder Public-Key authentication protocol. *Information Processing Letters*, 56(3):131–133, 1995.

[17] G. Lowe. Some new attacks upon security protocols. In , pages 162–169. Society Press, 1996.

[18] G. Lowe. Casper: A compiler for the analysis of security protocols. In *Proc. of the 10th IEEE Computer Security Foundations Workshop*, pages 18–30, 1997.

[19] A. Nacho and E. Aïmeur. Building a multi-agent system for automatic negotiation in web service applications. *Autonomous Agents and Multiagent Systems, International Joint Conference on*, 3:1466–1467, 2004.

[20] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21:393–399, 1978.

[21] PD2IS. Protocol compilation into interpreted systems, 0.90. `http://pc2is.sourceforge.net`, 2009.

[22] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[23] E. Shakshuki and S. Abu-Draz. Multi-agent system architecture to trading systems. *Journal of Interconnection Networks*, 6(3):283–302, 2005.