ATTACK INTERFERENCE IN NON-COLLABORATIVE SCENARIOS FOR SECURITY PROTOCOL ANALYSIS

M.-Camilla Fiazza, Michele Peroli and Luca Viganò
Department of Computer Science, University of Verona, Strada le Grazie 15, Verona, Italy

Keywords: Non-collaborative attackers, Attack interference, Dolev-Yao attacker, Attack mitigation, Security protocols.

Abstract: In security protocol analysis, the traditional choice to consider a single Dolev-Yao attacker is supported by the fact that models with multiple collaborating Dolev-Yao attackers have been shown to be reducible to models with one Dolev-Yao attacker. In this paper, we take a fundamentally different approach and investigate the case of multiple non-collaborating attackers. After formalizing the framework for multi-attacker scenarios, we show with a case study that concurrent competitive attacks can interfere with each other. We then present a new strategy to defend security protocols, based on active exploitation of attack interference. The paper can be seen as providing two proof-of-concept results: (i) it is possible to exploit interference to mitigate protocol vulnerabilities, thus providing a form of protection to protocols; (ii) the search for defense strategies requires scenarios with at least two attackers.

1 INTRODUCTION

1.1 Context

The typical attacker model adopted in security protocol analysis is the one of (Dolev and Yao, 1983): the *Dolev-Yao (DY) attacker* can compose, send and intercept messages at will, but, following the perfect cryptography assumption, he cannot break cryptography. The DY attacker is thus in complete control of the network — in fact, he is often formalized as being the network itself — and, with respect to network abilities, he is actually stronger than any attacker that can be implemented in real-life situations. Hence, if a protocol is proved to be secure under the DY attacker, it will also withstand attacks carried out by less powerful attackers; aside from deviations from the specification introduced in the implementation phase, the protocol can thus be safely employed in real-life networks, at least in principle.

Alternative attacker models have also been considered. On the one hand, *computational models* for protocol analysis consider attackers who can indeed break cryptography, as opposed to the *symbolic models* where cryptography is perfect (as we will assume in this paper). See, for instance, (Abadi et al., 2009) for a survey of models and proofs of protocol security, and (Basin and Cremers, 2010) for a protocol-security hierarchy in which protocols are classified by their relative strength against different forms of attacker compromise.

On the other hand, different symbolic models have been recently proposed that consider *multiple attackers*, instead of following the usual practice to consider a single DY attacker, a choice that is supported by the fact that models with multiple collaborating DY attackers have been shown to be reducible to models with one DY attacker (see, e.g., (Caleiro et al., 2005) for a detailed proof, as well as (Basin et al., 2011; Comon-Lundh and Cortier, 2003; Syverson et al., 2000) for general results on the reduction of the number of agents to be considered). For instance, (Basin et al., 2009; Schaller et al., 2009) extend the DY model to account for network topology, transmission delays, and node positions in the analysis of real-world security protocols, in particular for wireless networks. This results in a distributed attacker, or actually multiple distributed attackers, with restricted, but more realistic, communication capabilities than those of the standard DY attacker.

Multiple attackers are also considered in the models of (Arsac et al., 2009; Arsac et al., 2011; Bella et al., 2003; Bella et al., 2008), where each protocol participant is allowed to behave maliciously and intercept and forge messages. In fact, each agent may behave as a DY attacker, without colluding nor sharing knowledge with anyone else. The analysis of security protocols under this multi-attacker model allows one to consider scenarios of agents competing with each other for personal profit. Agents in this model may also carry out *retaliation attacks*, where an attack is followed by a counterattack, and *anticipation attacks*, where an agent's attack is anticipated, before its termination, by another attack by some other agent.

The features of the models of (Basin et al., 2009; Schaller et al., 2009) and of (Arsac et al., 2009; Arsac et al., 2011; Bella et al., 2003; Bella et al., 2008) rule out the applicability of the *n*-to-1 reducibility result for the DY attacker, as the attackers do not necessarily collaborate, and might actually possess different knowledge to launch their attacks. They might even attack each other. In fact, retaliation and anticipation allow protocols to cope with their own vulnerabilities, rather than eradicating them. This is possible because agents are capable of doing more than just executing the steps prescribed by a protocol: they can decide to anticipate an attack, or to counter-attack by acting even after the end of a protocol run (in which they have been attacked). Still, retaliation may nevertheless be too weak as honest agents can retaliate only *after* an attack has succeeded, and cannot defend the protocol during the attack itself.

1.2 Contributions

In this paper, we take a fundamentally different approach: we show that multiple non-collaborating DY attackers may interfere with each other in such a manner that it is possible to exploit interference to mitigate protocol vulnerabilities, thus providing a form of protection to flawed protocols.

In the approach we propose, instead of looking for attacks and reacting to the existence of one by redesigning the vulnerable protocol, we look for strategies for defending against existing known attacks. We would be performing protocol analysis to identify possible *defenses*, rather than attacks.

To investigate non-cooperation between attackers, we propose a (protocol-independent) model in which: (i) a protocol is run in the presence of multiple attackers, and (ii) attackers potentially have different capabilities, different knowledge and can interfere with each other. This, ultimately, allows us to create a benign attacker for system defense: honest agents can rely on a *network guardian*, an ad-hoc agent whose task is diminishing the frequency with which dishonest agents can succeed in attacking vulnerable protocols. This methodology moves the focus away from an attack-based view of security and towards a defense-based view.

We proceed as follows. In Section 2, we formalize models for the network and the agents, including, in particular, agent attitude, goals, and disposition. We then consider in Section 3 a vulnerable protocol from (Boyd and Mathuria, 2003) as a case study and focus on the interactions between attack procedures, interactions which cannot be observed in classical settings. In Section 4, we explain how interference between attacks leads to a methodology that can be used for defending vulnerable protocols against attacks. In Section 5, we conclude by discussing our approach and current and future work. The Appendix provides additional details about the case study.

2 SYSTEM MODELS: NETWORK, AGENTS, ATTITUDE

2.1 Goals of Modeling and Approach

Network models for security protocol analysis typically either replace the communication channel with a single attacker or build dedicated channels for each attacker (e.g. (Basin et al., 2011; Caleiro et al., 2005; Dilloway and Lowe, 2007; Kamil and Lowe, 2010; Syverson et al., 2000)). Traditional modeling strategies are not adequate to describe the non-collaborative scenario under consideration. The main shortcoming is the fact that the ability to spy the communication on a particular channel is hard-wired in the network model and may depend critically on network topology or attacker identity; the result is that an information-sharing mechanism (or a partial prohibition for it) is structurally encoded in the network. We would like, instead, to (i) abstract from positional advantages and focus solely on how attackers interfere *by attacking*; (ii) treat information-sharing (also as a result of spying) as a strategic choice of the agents.

For simplicity, in this paper we restrict our attention to *two* non-collaborative attackers (E_1 and E_2), in addition to the two honest agents A and B and a trusted third-party server S, whose presence is required by the protocol under consideration. In the following, let *Eves*$= \{E_1, E_2\}$ be the *set of attackers* and *Agents*$= \{A, B, E_1, E_2\}$ the *set of all network agents* (honest and dishonest, server excluded). Let X, Y, Z and W be variables varying in *Agents* and E a variable in *Eves*; j takes value in $\{1,2\}$, whereas $i \in \mathbb{N}$ is reserved for indexing states.

We are aware that, in situations with more than two (dis)honest agents, further types of interactions can arise; however, a full comprehension of the interactions depends on building a clear picture of interference. Such a picture necessarily starts with the elementary interaction between two attackers.

In order to focus on the raw interference between two attackers, both directing their attack towards the same target, it is important for all attackers to have access to the same view of what is taking place with honest agents and possibly different views of what is taking place with the other attacker(s). If attackers do not all have the same information, it is possible to conceive of strategies in which some attackers can be mislead by others on purpose.

If the knowledge[1] available to an attacker affects his view of the system, attacker capabilities and effectiveness can be diversified, without needing to construct asymmetric attackers or hardwire constraints that may hold for some attackers and not for others. We find it relevant that a network model for non-collaborative scenarios — besides reflecting this stance — also support a form of competition for access to messages, especially if attacks rely on erasing messages.

If it is possible in principle to actively interfere with an attack, it should be possible to do so even if all attackers have the same knowledge. However, differentiating attackers with respect to their understanding of the situation — in particular with respect to awareness of other attackers — may bring into focus the conditions, if any, that allow an attacker to interfere with another without being interfered with.

We diversify the activity of our attackers by admitting that attackers may choose to selectively ignore some messages, on the basis of the sender's and receiver's identifiers. This choice reflects actual situations in which attackers pay attention to only a subset of the traffic through a network, focusing on the activity of some agents of interest. Regardless of whether this selection is caused by computational constraints or by actual interest, real attackers filter messages on the basis of the sender's or receiver's identity. In the following, we will use the set $Attend_E$ to model the agents to which attacker E is attentive; the predicate $ofInterest_E(X)$ (see Table 1) models the decisional process of attacker E as he considers whether he wishes to augment $Attend_E$ with X, i.e. $ofInterest_E(X)$ implies that X is added into $Attend_E$.

Honest agents are interested in *security properties* (such as authentication or secrecy) being upheld through the use of protocols. Dishonest agents, on the other hand, are interested in changing or negating such properties. The characteristic feature of the attackers we consider is their attitude. In particular, in the case study presented in Section 3, dishonest agents wish to attack the security protocol and are ready, should they encounter unforeseen interference, to take countermeasures with respect to the interference as well. In a sense, each attacker is exclusively focused on attacking the protocol and becomes aware of other attackers through their effect on his success.

Our target is capturing the behavior of *equal-opportunity* dishonest agents that do not cooperate in the classical sense. By equal-opportunity attackers we mean agents that have the same attack power and that may differ with respect to the information content of their knowledge bases. Such differentiation arises out of attentional choices and not out of intrinsic constraints. Strategic and attitude considerations should not be derivable explicitly from the attacker model — rather, they should configure it.

The driving hypothesis of our work is that studying non-collaboration requires a complex notion of attacker, whose full specification involves attentional choices, decisional processes pertaining to the network environment and to other agents, cooperation-related choices and decisional processes pertaining to the attack strategy. To support this type of attacker, we extend the usual notions of protocol and role by introducing a control — a mechanism to regulate the execution of the steps prescribed by the attack trace in accordance with the attacker's strategy. In our model, honest agents perform a controlled execution of the protocol as well, so as to support in-protocol detection of attacks. Honest agents behave according to the protocol's prescription, expect things to go exactly in accordance with the protocol and interpret deviations in terms of the activity of dishonest agents.

2.2 Agent Model

Agent knowledge is characterized in terms of a proprietary dataset. To each X in *Agents*, we associate the dataset D_X, which we assume to be monotonically non-decreasing. Our agents, in particular dishonest agents, collect information but do not forget it. When it is important to highlight that the dataset is to be considered at a particular moment, we will use D^i_X instead.

The network *net* is also formalized through a dataset, which is named D_{net} and indexed in the same manner as D^i_X. A dataset is a simple network model that can be configured to support complex attackers; we believe it can successfully meet all of our modeling requirements for non-collaboration. We postpone to Section 2.3 the discussion of how datasets evolve and how indexing and evolution are related to actions and message transmission.

We adapt the notion of DY attacker (Dolev and

[1] Note that we do not attach any epistemic interpretation to the knowledge we consider in this paper. We simply consider the information initially available to the agents, together with the information they acquire during protocol executions.

Table 1: Dolev-Yao attacker model for non-collaborative scenarios: internal operations (synthesis and analysis of messages), network operations (*spy*, *inject*, *erase*) and system configuration (*True-Sender-ID*, *DecisionalProcess*, *NetHandler*). *NetHandler* describes the set of attackers who are allowed to spy by applying one of the *spy* rules. We omit the usual rules for conjunction. The rules employed in the case study are marked in boldface.

$$\frac{m_1 \in D^i_E \quad m_2 \in D^i_E}{(m_1, m_2) \in D^i_E} \ \textbf{\textit{(Comp)}} \qquad \frac{m \in D^i_E \quad k \in D^i_E}{\{m\}_k \in D^i_E} \ \textbf{\textit{(Encr)}}$$

$$\frac{(m_1, m_2) \in D^i_E}{m_j \in D^i_E \text{ for } j \in \{1,2\}} \ \textbf{\textit{(Proj)}} \qquad \frac{\{m\}_k \in D^i_E \quad k^{-1} \in D^i_E}{m \in D^i_E} \ \textbf{\textit{(Decr)}}$$

$$\frac{< X, m, Y > \in D^i_{net} \quad \text{sender}(< X, m, Y >) \in D^i_E \quad Y \in D^i_E \quad \psi}{m \in D^{i+1}_E} \ \textbf{\textit{(Restricted-Spy)}}$$

$$\frac{< X, m, Y > \in D^i_{net} \quad \mathit{ofInterest}_E(X) \quad Y \in D^i_E \quad \psi}{m \in D^{i+1}_E \wedge \mathit{sender}(< X, m, Y >) \in D^{i+1}_E} \ (\mathit{Inflow\text{-}Spy})$$

$$\frac{< X, m, Y > \in D^i_{net} \quad \mathit{sender}(< X, m, Y >) \in D^i_E \quad \mathit{ofInterest}_E(Y) \quad \psi}{m \in D^{i+1}_E \wedge Y \in D^{i+1}_E} \ (\mathit{Outflow\text{-}Spy})$$

$$\text{where } \psi = E \in \mathit{canSee}(< X, m, Y >, i)$$

$$\frac{m \in D^i_E \quad X \in D^i_E \quad Y \in D^i_E}{< E(X), m, Y > \in D^{i+1}_{net}} \ \textbf{\textit{(Injection)}}$$

$$\frac{< X, m, Y > \in D^i_{net} \quad \mathit{sender}(< X, m, Y >) \in D^i_E}{< X, m, Y > \notin D^{i+1}_{net}} \ \textbf{\textit{(Erase)}}$$

$$\mathit{sender}(< X, m, Y >) = \begin{cases} E & \text{if there exists } Z \text{ such that } X = E(Z) \\ X & \text{otherwise} \end{cases} \ \textbf{\textit{(True-sender-ID)}}$$

$$\mathit{ofInterest}_E(X) = \begin{cases} \mathit{true} & \text{if } E \text{ decides to pay attention to } X \\ \mathit{false} & \text{otherwise} \end{cases} \ (\mathit{DecisionalProcess})$$

$$\mathit{canSee}(< X, m, Y >, i) = \{Z \in \mathit{Eves} \mid Z \text{ can spy } < X, m, Y > \text{ on } D^i_{net}\} \ \textbf{\textit{(NetHandler)}}$$

Yao, 1983) to capture a non-collaborative scenario. We show in Table 1 how one such attacker is formalized within our model, writing rules for attacker E with respect to the knowledge base D_E and the network model D_{net}. Let us specify that the rules in Table 1 are transition rules, rather than deduction rules. Taken altogether, they construct a *transition system* – which describes a computation by describing the states that are upheld as a result of the transition. We do not intend to carry out in this paper logical inference to identify defenses against attacks; rather, we recognize in the system's evolution what in our eyes corresponds to a defense.

Attackers are legitimate network agents that can *send* and *receive* messages, derive new messages by analyzing (e.g. decomposing) known messages, obtain messages transiting on the network (*spy*) and remove them so that they do not reach their intended receiver (*erase*). Attackers can also partially impersonate other agents, by *inject*ing messages under a false identity; we represent impersonification with the notation $E(X)$, where E is the impersonator and X is the identifier of the impersonated agent. This set of abilities describes agents who have control over almost all facets of a communication; their characteristic limitation is that they cannot violate cryptography (we assume perfect cryptography). Note that further rules could be added in Table 1 for other forms of encryption, digital signatures, hashing, creation of nonces and other fresh data, and so on. For conceptual clarity, we explicitly pair an *erase*-rule with the *injection*-rule, to emphasize that an attacker can modify messages (by erasing them and injecting a substitute) or send messages under a false identity (partial impersonification).

The most significant feature concerns spying. Our attackers can employ three different *spy* rules, adapted to formalize the fact that attackers do not pay attention to all of the traffic on the network. The *spy* rules rely on an interpretation for "send" that is modified with respect to the denotational semantics in (Caleiro et al., 2006), to reflect the attentional focus of attackers. The default *spy* is the *Restricted-Spy*: only the messages involving known agents in both sender and receiver roles, regardless of hypotheses on their honesty, become part of the attacker's dataset. Note that in our model what matters is the actual sender and not the declared sender (*True-Sender-ID*). This mechanism prevents total impersonification and allows filtering messages on the basis of the agent's attentional choices.

The attentional filter we use is meant as a choice of the agents and not as a constraint to which they are subject; therefore, it must be possible to expand the set of agents of interest. This role is fulfilled by the two exploratory *spy* rules in Table 1, *Inflow-Spy* and *Outflow-Spy*. Attackers have the option of accepting or rejecting the newly discovered identifier X, on the basis of the predicate $\mathit{ofInterest}_E(X)$, which models the decisional process for attention.

Note that an attacker cannot apply any of the *spy* rules to obtain the message m without knowing the identifier of at least one between m's sender and m's intended receiver. By not providing a "generalized spy" rule to waive this requirement, we ensure that $(D^0_E \cap \mathit{Agents} = \emptyset)$ implies that for all i, $(D^i_E \cap \mathit{Agents} = \emptyset)$. Although E can augment its knowledge base D_E indefinitely — through internal message generation and the synthesis rules *Comp* and *Encr* —, E's network activity is in fact null. One such E is a *dummy attacker*, whose usefulness becomes apparent when considering that proof of reductions for non-collaboration can involve progressively migrating identifiers from an attacker's dataset, until the attacker himself reduces to the dummy attacker.

An attacker's dataset D_E consists of (i) messages that have transited through the network and that have been successfully received, analyzed or spied and (ii) identifiers of the agents to whom the attacker is attentive. The set $Attend_E$ of identifiers of interest to E is further partitioned into three sets: the set H_E of agents believed[2] to be honest, the set A_E of agents believed to be attackers, and the set U_E of agents whose attitude is unknown in E's eyes. Note that, differently from D_{net}, agent datasets do not contain triplets ($\langle$*sender-ID*, *message*, *receiver-ID*$\rangle$), but only messages or identifiers.

Once a new identifier X enters the knowledge base of attacker E, E establishes a belief about the honesty of X and places the identifier in one of the sets H_E, A_E or U_E. We do not enter details on how the agents initially build their knowledge base and establish their belief about the attitude of other known agents. In fact, this classification is meant to be dynamic. Agents are on the watch for suspicious messages, which may indicate that an attack is ongoing or may reveal that a certain agent is dishonest. Dynamically adapting their beliefs about the honesty of other agents allows the agents to gather important information during single protocol runs. The agents we wish to consider are *smart*: they always employ the available strategic information.

Attackers do not have automatic access to triplets that relate sender, message and receiver. They must infer this key piece of information on the basis of the identifiers of the agents to which they are attentive, and attempt to relate the identifiers to the messages they spy. Inference is easier if attackers use only the *Restricted-Spy* rule and keep the set $Attend_E$ of known agents small.

2.3 Network Model

All the operations that can change the state of the network dataset D_{net} (*send*, *receive*, *inject* and *erase*) are termed *actions*, whereas we consider *spy* simply as an operation: although it requires interacting with the network, it does not change its state. Messages in transit are inserted in the network dataset D_{net}, where attackers can spy them before they are delivered to their intended receivers. Contextually to delivery, the message is removed from the dataset. Messages transit on the network dataset in the form of triplets of the type $\langle$*sender-ID*, *message*, *receiver-ID*$\rangle$. As a consequence of message delivery or deletion, D_{net} is non-monotonic by construction.

The sequence of actions that takes place during a protocol run is enumerated and used to index the evolution of the network dataset D_{net}; the index of D^i_{net} is shared with all the proprietary datasets D^i_X, whose states are synchronized accordingly. D^i_{net} is the state of the network dataset *after the i-th action*.

Customarily, evolutions are indexed per transition (per rule application), rather than per action. Our chosen indexing strategy reflects three needs: (1) allowing agents to fully analyze newly acquired messages without having to keep track of the number of internal operations performed; (2) supporting a form of competition between attackers for access to the network; (3) supporting a form of concurrence.

Ideally, all attackers act concurrently. However, the state transitions for the network must be well-defined at all times, even if attackers try to perform conflicting actions, such as spying and deleting the same message in transit. To impose a measure of order, we introduce a *network handler*, whose task is to regulate the selection of the next action and implement the dependencies between selected action and knowledge available to each attacker. Through the network handler, it is also possible to keep the system evolution in accordance with additional constraints, e.g. modeling information sharing within specific subsets of agents or modeling network topology.

As soon as the state of the network changes (e.g. as a result of *inject* or *send*), the network handler passes the new triplet to each attacker, who then *simulates* spying and decides on whether to request erasing the message or injecting a new one as a consequence, in accordance with his strategy. The network handler interprets the application of the inject-rule and of the erase-rule as requests and selects the next action from the set of requests. Message deletion, when requested by any attacker, is always successful.

The outcome of the process governed by the network handler is described through the function *canSee*(), which returns a subset of *Eves*, highlighting the identifiers of the attackers who can spy "before" the message is erased from D_{net}. The set of agents described by *canSee*() contains at least the identifier of the attacker whose erase/inject request was served.

If the network handler does not receive any erase- or inject- requests, all attentive attackers can acquire the message. If one or more erase-requests are present, the network handler erases the message and confirms success in spying only for a subset of attentive attackers. If an attacker is not in *canSee*(),

[2] We do not attach any doxastic interpretation to the beliefs we consider in this paper.

the prior (simulated) spy is subject to rollback, along with all internal operations that have occurred since the last confirmed action. If no requests are received from attackers, the network handler oversees message delivery or selects actions requested by honest agents.

Although the formulation of *canSee*() in terms of access time is intuitive, the reason why we favor this mechanism is that time-dependent accessibility is not the only situation it can model. The function can be instantiated to model strategic decision-making and information-sharing, or to capture a particular network topology. In realistic attack scenarios, knowledge of a message that has been erased may depend more on cooperation and information-sharing than on timing. For example, if E_j is sharing information with E_k (but not vice versa), whenever E_j's erase requests are served E_k is automatically in *canSee*().

The network handler is not an intelligent agent. Specifying its behavior and instantiating the function *canSee*() corresponds to configuring the particular network environment in which the agents are immersed (i.e. *canSee*() is a configurable parameter of our model).

As a result of the network handler and of our chosen indexing strategy, several internal operations can occur in a proprietary dataset between consecutive states, whereas only a single action separates consecutive states of the network dataset. Attackers determine the next state of the network dataset with priority with respect to the actions of honest agents.

In Table 2, we formalize within our model operations in the Alice&Bob notation used in Section 3; we write $E_I(Y)$ to denote the subset of *Eves* who spy message m addressed to Y, at least one of which has requested m to be erased.

With reference to Table 2, note that the $(i+1)^{\text{th}}$ action is requested when the state of the network is D^i_{net} and agent datasets are D^i_X; thus, the sender X must already know in D^i_X both the message m and the identifier of the intended recipient Y. The message correctly transits on D^{i+1}_{net}, immediately after being sent. The $(i+2)^{\text{th}}$ action is either *receive* (first two cases) or *erase* (last case). The availability of m to attackers is conclusively decided after the network handler selects the $(i+2)^{\text{th}}$ action, and thus pertains to D^{i+2}_W.

2.4 Attacker Goals and Disposition

The notion of cooperation between agents can be viewed from at least two perspectives of interest: sharing of information and sharing of success. The notion of attacker cooperation classically employed in protocol analysis encompasses both aspects, as it states the first while assuming that the second holds.

Table 2: Representation of operations in Alice&Bob notation.

$(i+1)^{\text{th}}$ action	Formalization
$X \rightarrow Y : m$	$m \in D^i_X$ and $Y \in D^i_X$
	$< X, m, Y > \in D^{i+1}_{net}$ and $< X, m, Y > \notin D^{i+2}_{net}$
	$m \notin D^{i+2}_W$, where $W \notin canSee(< X, m, Y >, i+1)$
	$m \in D^{i+2}_Y$
$E(X) \rightarrow Y : m$	$m \in D^i_E$ and $X \in D^i_E$ and $Y \in D^i_E$
	$< E(X), m, Y > \in D^{i+1}_{net}$ and $< E(X), m, Y > \notin D^{i+2}_{net}$
	$m \notin D^{i+2}_W$, where $W \notin canSee(< X, m, Y >, i+1)$
	$m \in D^{i+2}_Y$
$X \rightarrow E_I(Y) : m$	$m \in D^i_X$ and $Y \in D^i_X$
	$< X, m, Y > \in D^{i+1}_{net}$ and $< X, m, Y > \notin D^{i+2}_{net}$
	$m \in D^{i+2}_W$, where $W \in I$ and $I \subseteq canSee(< X, m, Y >, i+1)$

In this paper, we examine attackers that exhibit, with respect to cooperation, the behavior we call *complete non-collaboration*: agents voluntarily abstain from sharing information and do not consider their goals as met if they do not succeed in attacking. The *disposition* of attacker E_1 towards E_2 belongs to one of the following basic classes: active collaboration, passive collaboration, competition and conflict[3]. The focus of this paper is on competition – a situation in which the goal is successfully attacking the protocol, regardless of the disposition of other agents. From the perspective of a competitive attacker, other attackers are not of interest per se: they are relevant factors because they are sources of interference. If some interference is detected while carrying out an attack, a competitive attacker will take countermeasures, attempting to negate potentially adverse effects.

Our scenario of interest is composed by a set of two agents that are homogeneous with respect to their (competitive) disposition.

3 A CASE STUDY

A dishonest agent, aware that other independent attackers may be active on the network, will seek to devise suitable novel attacks, so as to grant himself an edge on unsuspecting competitors. As the mechanics of interaction and interference between attackers have not been exhaustively studied in literature yet, it is not known a priori how to systematically derive an attack behavior of this type.

In the following case study, we start from a simple protocol for which a vulnerability is known; we devise for the known ("classical") attack a variant that

[3]In active and passive collaboration there is a common goal to be pursued; the difference lies in choosing a strategy that helps another vs. choosing a strategy that does not hinder another. In conflict scenarios, the primary focus of interest is the attackers, rather than the protocol.

Table 3: The Boyd-Mathuria Example protocol and a masquerading attack against it.

BME			Classical Attack		
			(1)	$A \to E(S)$	$: A, B$
(1)	$A \to S$	$: A, B$	(1′)	$E(A) \to S$	$: A, E$
(2)	$S \to A$	$: \{\vert k_{AB} \vert\}_{k_{AS}}, \{\vert k_{AB} \vert\}_{k_{BS}}$	(2)	$S \to A$	$: \{\vert k_{AE} \vert\}_{k_{AS}}, \{\vert k_{AE} \vert\}_{k_{ES}}$
(3)	$A \to B$	$: \{\vert k_{AB} \vert\}_{k_{BS}}$	(3)	$A \to E(B)$	$: \{\vert k_{AE} \vert\}_{k_{ES}}$

explicitly considers the possibility of ongoing independent attacks. We describe a possible reasoning for a competitive attacker in the context of the protocol's main features. Due to space limitations, we give additional details about the case study in the Appendix.

The protocol we consider as a case study is a key transport protocol described as an example in (Boyd and Mathuria, 2003); we name it as the Boyd-Mathuria Example (BME), and present it in Table 3 together with a classical attack against it. BME relies on the existence of a trusted third-party server S to generate a session key k_{AB} for agents A and B, where each agent X is assumed to share a symmetric secret key k_{XS} with S.

A is subject to a masquerading attack in which, at the end of a run of BME, A thinks that he shares a session key with the honest agent B, while in fact he shares it with the attacker E. Subsequent communication from A addressed to B is seen by E through the spy-rule and removed with an erase request: E has successfully taken B's place. This attack prevents B from receiving *any* communication from A. Should the two agents have prior agreement that such a communication was to take place, B is in the position of detecting that something has gone wrong. E can prevent detection by staging a dual man-in-the-middle attack.

If more than one attacker is active during a given protocol run, simultaneous execution of the classical attack could lead to A receiving multiple session keys as a response to his (single) request to the server. This situation clearly indicates to A that an attack is ongoing. A competitive attacker E_1, wishing to prevent this situation from occurring, could try removing from the network all the responses from S to A that do not pertain to his own request. However, the characteristics of the (non-redundant) cryptographic methods employed here do not allow distinguishing $M_1 = \left(\{|k_{AE_1}|\}_{k_{AS}}, \{|k_{AE_1}|\}_{k_{E_1S}}\right)$ (to let through) from $M_2 = \left(\{|k_{AE_2}|\}_{k_{AS}}, \{|k_{AE_2}|\}_{k_{E_2S}}\right)$ (to block). E_1 can recognize the format of M_1 and M_2 and can successfully decrypt M_1 to recover k_{AE_1}; by decrypting M_2 with the key k_{E_1S}, E_1 can still recover a value, but different from the previous one. Not knowing k_{AE_1} a priori, the attacker is not able to distinguish which of M_1 and M_2 contains the answer to his request for a key with A.

As a consequence, the attacker E_1 is not able to know which messages to remove in order to ensure that A accepts k_{AE_1} as a session key to communicate with B. Competitive attackers cannot rely on step (2) to enforce their attacks at the expense of their competitors; furthermore, the probability of erasing all competing messages (while letting one's own pass) decreases with the number of active attackers. In this situation, it becomes fundamental for a competitive attacker to gain exclusive access to the first message — to gain control over the messages that reach S, as opposed to the messages coming from S [4].

After spying the initiator's opening message, a competitive attacker E_1 will therefore attempt to mount the classical attack, while keeping watch for other messages that may be interpreted as attack traces. Any transiting message of the type (A, E_m) for which $E_m \in A_{E_1}$ is interpreted as another active attack; E_1 counters by requesting that the message be erased. If E_m is in H_{E_1}, the message may be understood either as a message from A — who would be initiating a parallel session of the protocol to obtain a second session key — or as an indication that E_m has been incorrectly labeled as honest. In the first case, E_1 will let the message through, as he has chosen to target specifically the session key for the communication between A and B; in the second case, he will protect his attack by erasing the message. If E_m is in U_{E_1}, E_1 can choose to either play conservatively and hypothesize the dishonesty of E_m or let the message through and interpret E_m as the culprit in case the current attack fails.

BME is such that at most one attacker E_d can successfully mislead A into accepting the key k_{AE_d} as a session key to communicate with B. Therefore, a successful attack automatically entails exclusivity of success. An attack is successful if it goes undetected by the initiator A. Our honest agents are intelligent and they make use of all information available to perform in-protocol detection of attacks. With respect to BME, a clear indication for A consists in receiving multiple responses from S after a single session key request; if A receives multiple responses, he concludes that there has been a security violation and thus does not employ any of the keys so received in his later communications with B – choosing to try a fresh run of the protocol instead. From the attackers'

[4]Of course, E_1 could guess which message(s) to erase, but he would have the added difficulty of having to decide whether to let the first message pass without knowing how many other messages will transit, if any at all, and how many session keys were requested by A (as opposed to by his competitor(s)).

perspective, an ongoing attack can be detected by observing a message of the type (A,X) transiting on the network; however, the attack trace is ambiguous to spying attackers and has to be interpreted on the basis of current beliefs concerning the honesty of X. A last feature of interest is that BME is rather friendly for attacker labeling. Decisional processes can rely on at least some conclusive information on the identity of the agents involved, because identifiers transit in the clear; attackers would have to infer them otherwise.

We examine the outcome of attacks carried out in a non-collaborative environment in six cases, corresponding to different conditions of knowledge and belief for E_1 and E_2. Cases and attack traces are summarized in Table 4. In order to completely specify agent behavior, we posit the following:

1. If an attacker E spies (A,E_m) with $E_m \in H_E$ or $E_m \in U_E$, he will not request that the message be erased. In the latter case, if E's attack fails, E_m is immediately placed in A_E.
2. Both E_1 and E_2 spy the opening message and are interested in attacking the current protocol run; this allows us to leave aside the trivial cases in which only one attacker is active for a given protocol run.
3. Due to space constraints, we detail only the cases in which *canSee* for step (3) yields $\{E_1,E_2\}$. Cases in which only one of the attackers can access A's response can be found in the Appendix.

Case 1: E_1 and E_2 Know each other as Honest. E_1 and E_2 know each other's identifiers (i.e. they are paying attention to each other: $E_1 \in D_{E_2}$ and $E_2 \in D_{E_1}$), but they are both mistaken in that they have labeled the other as honest ($E_1 \in H_{E_2}$ and $E_2 \in H_{E_1}$). E_1 and E_2 are unaware of active competitors and mount the classical attack in steps (1_1) and (1_2). When the attackers spy two requests to the server transiting on the network, they both believe that A wishes to request keys with the honest agents B and E_j.

(1.T1). *S sends two messages before A can address a message to B.* With the messages in steps (2_1) and (2_2), A receives two keys instead of the single key requested. A now knows that at least one attacker is active and abandons the protocol without sending a message to B. The attackers do not spy the message they were hoping for (timeout) and acquire the certainty that at least another active attacker is around. The attackers can employ ad-hoc strategies to search for the mislabeled or unknown attacker. If the attackers are careful to keep track of the messages (A,X) pertaining to a given session, they can make informed guesses as to whom, amongst the known agents, they might have mislabeled.

(1.T2). *A receives a reply from S, answers B and stops listening.* A receives the messages he expects and closes the current session before receiving the second response from S. E_1 is successful in his attack, whereas E_2 believes that he has succeeded when he has, in fact, decrypted the wrong key. None of the agents have an opportunity for detection.

(1.T3). *A receives a reply from S, answers B and keeps listening.* A replies with the message in step (3), resulting in both E_1 and E_2 believing that they have succeeded. However, after receiving (2_2), A detects the attack and abstains from employing k_{AE_1} in his future communications with B. Thus, even if for different reasons, both attackers in fact fail. Furthermore, they both continue to hold their mistaken belief that the other attacker is in fact honest.

Case 2: E_1 and E_2 Know each other as Attackers. E_1 and E_2 know each other's identifier ($E_1 \in D_{E_2}$ and $E_2 \in D_{E_1}$) and have correctly understood that the other is behaving as a dishonest agent ($E_1 \in A_{E_2}$ and $E_2 \in A_{E_1}$). Each attacker is aware of the presence of a competitor, which they have correctly labeled. Each attacker is attempting to gain exclusive access to the initial communication towards S and to ensure that only his request reaches S. E_1 and E_2 erase each other's request to S. Within our model, no attacker can be certain that his message has been received by its intended receiver; the attackers may wish to replay step (1_1) and (1_2) if a message of the type $\{|k_{AE_j}|\}_{k_{AS}}, \{|k_{AE_j}|\}_{k_{E_jS}}$ is not spied on the network within a reasonable time. This option is marked with $(\cdot)^+$ in Table 4. However, the active presence of the competitor ensures that no message reaches S. A notices that an anomalous situation is occurring, because his request to the server is not being served in a reasonable time. A interprets the situation as a denial-of-service attack and abandons the protocol.

Case 3: E_1 and E_2 are Unaware of each other. E_1 and E_2 are unaware of the other's presence – i.e. they are not paying attention to the other's activity ($E_1 \notin D_{E_2}$ and $E_2 \notin D_{E_1}$). Subcases follow closely those described for case 1 above. The only significant difference concerns detection for trace T1: here the attackers must employ exploratory strategies (*Inflow-Spy* or *Outflow-Spy*), because they failed to spy an additional message of type (A,E_m) transiting on the network. The failure to observe such a message is a strong indicator that the competitor's identifier is unknown. In 2-attacker scenarios this is the only legitimate conclu-

Table 4: Traces for non-collaborative attacks against BME. Traces are exhaustive: E_1 and E_2 have priority over honest agents and S is honest. Arrows: relative order between (1_1) and (1_2) is irrelevant in determining the outcome.

T1: cases 1, 3, 4, 6			**T2 and [T3]: cases 1, 3, 4, 6**		
$(1\ )$	$A \to E_{1,2}(S)$	$: A, B$	$(1\ )$	$A \to E_{1,2}(S)$	$: A, B$
$\downarrow (1_1)$	$E_1(A) \to S$	$: A, E_1$	$\downarrow (1_1)$	$E_1(A) \to S$	$: A, E_1$
$\uparrow (1_2)$	$E_2(A) \to S$	$: A, E_2$	$\uparrow (1_2)$	$E_2(A) \to S$	$: A, E_2$
(2_1)	$S \to A$	$: M_1$	(2_1)	$S \to A$	$: M_1$
(2_2)	$S \to A$	$: M_2$	$(3\ )$	$A \to E_{1,2}(B)$	$: \{\lvert k_{AE_1} \rvert\}_{k_{E_1 S}}$
			$[(2_2)$	$S \to A$	$: M_2]$

T4: case 2			**T5: case 5**		
$(1\ )$	$A \to E_{1,2}(S)$	$: A, B$	$(1\ )$	$A \to E_{1,2}(S)$	$: A, B$
$\downarrow (1_1)^+$	$E_1(A) \to E_2(S)$	$: A, E_1$	$\downarrow (1_1)$	$E_1(A) \to E_2(S)$	$: A, E_1$
$\uparrow (1_2)^+$	$E_2(A) \to E_1(S)$	$: A, E_2$	$\uparrow (1_2)$	$E_2(A) \to S$	$: A, E_2$
			$(2\ )$	$S \to A$	$: M_2$
			$(3\ )$	$A \to E_{1,2}(B)$	$: \{\lvert k_{AE_2} \rvert\}_{k_{E_2 S}}$

Where: $M_1 = \{\lvert k_{AE_1} \rvert\}_{k_{AS}}, \{\lvert k_{AE_1} \rvert\}_{k_{E_1 S}}$, $M_2 = \{\lvert k_{AE_2} \rvert\}_{k_{AS}}, \{\lvert k_{AE_2} \rvert\}_{k_{E_2 S}}$

sion, whereas with three or more attackers this situation may also arise from the interplay between erase and spy operations.

Case 4: E_2 Knows E_1 as Honest. Only one out of the two attackers E_1 and E_2 is paying attention to the other and knows his identifier. Here we consider $E_1 \in H_{E_2}$ and $E_2 \notin D_{E_1}$. Regardless of the order in which steps (1_1) and (1_2) occur, the attacker in disadvantage E_1 does not spy the message at step (1_2); E_2 does spy (1_1) but, trusting his judgement on E_1's honesty, does not request it to be erased. As a consequence, similarly to case 1, the traces follow schemes T1, T2 and T3. Significant differences concern detection in T1: E_1 detects the presence of an unknown attacker, whereas E_2 learns of a mislabeled or unknown attacker. The successful attackers in traces T2 and T3 are those whose requests to S are served first; knowledge does not affect the outcome.

Case 5: E_2 Knows E_1 as Dishonest. Only one out of the two attackers E_1 and E_2 is paying attention to the other and knows his identifier. Here we consider $E_1 \in A_{E_2}$ and $E_2 \notin D_{E_1}$ Regardless of the order in which steps (1_1) and (1_2) occur, E_1 does not spy the message at step (1_2) and E_2 uses a direct attack against the competitor. E_2 removes E_1's request to the server and remains the only attacker in play, leading A into accepting k_{AE_2} as a session key. E_1 does not have an opportunity to detect the competitor.

Case 6: E_2 Knows E_1, but he is Unsure of E_1's Honesty. Only one out of the two attackers E_1 and E_2 is paying attention to the other and knows his identifier. Here we consider $E_1 \in U_{E_2}$ and $E_2 \notin D_{E_1}$. This case reduces to case 4, with the only difference that E_2 is testing the dishonesty of E_1, instead of believing his honesty. Whenever E_2 realizes that he has failed his attack, he adds E_1 into A_{E_2} and deletes it from U_{E_2}.

General Considerations. In traces T2 and T3, the winning attacker is the one whose request is served first by S. S is an honest agent but it is not constrained to answering requests in the exact order in which they are received. Attackers do not have control over which requests are served first, although this factor determines whether they cannot do better than acquire the wrong key. Attackers realize in-protocol that they have failed only when they cannot spy a response from A, i.e. when they do not acquire any keys. Post-protocol detection, on the other hand, can occur also when an attacker with a wrong key attempts to decrypt the later communications addressed by A to B.

The case study highlights that, if A keeps the session open for a reasonable time after step (3), he can improve his chances of discovering that the key is compromised. This is a simple strategy that is beneficial and does not depend on the particular protocol. Furthermore, when A receives two answers from S in response to his single request, he now has two keys – at least one of which is shared with an attacker. If honest agents are immersed in a retaliatory framework (Bella et al., 2003; Bella et al., 2008), such keys can be used to identify attackers, to feed them false information or, in general, to launch well-aimed retaliatory attacks.

4 DEFENDING PROTOCOLS

Key exchange protocols are amongst the most used cryptographic protocols. It is a common security practice to establish a secure channel by first exchanging a session key and then using it to authenticate and encrypt the data with symmetric cryptography. The security of all communications occurring during a session rests on the integrity of the key. In this context, it is not important per se that a key has been acquired by an attacker: what matters is whether a compromised key is used. Rather than on preventing the acquisition of a session key from ever occurring, the focus is on detecting that the key has been compromised – so as to prevent an attack from spreading to the entire session traffic.

If a protocol is vulnerable, a single DY attacker will succeed with certainty. However, if attacks to the same protocol are carried out in a more complex network environment, success is not guaranteed.

As shown in the case study, in competitive scenarios with equal-opportunity attackers it is not possible for a given attacker to ensure that an attack is successful under all circumstances. The outcome depends on the strategy and knowledge conditions of all active agents, on the visibility of erased messages to other attackers (*canSee* $\neq \{E_1, E_2\}$) and on the order in which S processes requests. In a sense, the presence of an independent active attacker constrains the success of otherwise sure-fire attacks.

In order to make use of the emergent interference between concurrent attacks, it is necessary to ensure that attacks mounted by a malicious attacker are immersed in a multi-agent scenario. To this end, we construct an additional non-malicious attacker, who carries out attacks against the protocol, discards the "security secret" whenever he acquires it and reasons on the basis of what he can observe, to assist honest agents in detection tasks. In a sense, we are manipulating the *execution environment* of malicious attacks to gain a chance to thwart them.

The presence of a non-malicious agent that behaves as an attacker can be exploited to facilitate detection of attacks against vulnerable protocols. Honest agents should not, in principle, be informed of the specific attack trace to which they are vulnerable. Hence, if honest agents can perform detection at all, it has to be on the basis of flags that are independent of the specific attack trace – and, in general, independent also of the protocol in use. Such flags encode *local* defense criteria and can be as simple as realizing that no answer has arrived within a time considered reasonable or realizing that two (different) answers have been sent in response to a single request.

The basic idea is constructing a network agent that causes protocol-independent flags to be raised – via deliberate interference with ongoing attacks. In addition, one such *guardian agent* is formally an attacker, and can therefore be configured with knowledge of the attack trace(s). The guardian's task can be formulated as raising protocol-independent flags in correspondence to protocol-dependent indicators.

By using such an ad-hoc competitor as defense, it is possible, in some cases, to allow detection of otherwise-undetectable attacks. If no flag is raised for A, the guardian may be the only attacker at work. In this case, no ill-intentioned attacker has successfully concluded an attack; from the standpoint of A, actual security is not affected. In Table 5, we show the effects of introducing a guardian G for BME, configured as the attackers in the case study.

A guardian is a practical solution even when it is not all-powerful: any attack detected by A thanks to the guardian's active presence is an improvement

Table 5: Effects of introducing a guardian G for BME when attacker E is active. G operates according to the same strategy as the attackers in the case study. G's active interference results in A detecting attacks always ($\surd$), sometimes ($\sim$), always if A commits to listening after step (3) ($^+$). The guardian is progressively more effective the more his beliefs and knowledge reflect the actual set of attackers. G can be effective even when he is not aware of E's presence.

canSee step(3)	**Cases 1,3,4,6**	**Case 2**	**Case 5:** $E \in A_G$	**Case 5:** $G \in A_E$
$\{E, G\}$	$\sim^+$	$\surd$	$\surd$	
$\{G\}$	$\surd$	$\surd$	$\surd$	$\surd$
$\{E\}$	$\sim^+$	$\surd$	$\surd$	

in security. It is not necessary to demand that the guardian monitor all traffic – which is unrealistic at best; on the other hand, all monitored traffic enjoys partial protection.

Attacks failing are, by themselves, markers that there are other dishonest agents at work; this fact can be used by the guardian G as a basis for further detection, possibly on behalf of honest agents. Employing guess-and-test strategies can then lead to an understanding of the second attacker's identity; a rudimentary example is the strategy used by our attackers for BME when they spy (A, E_m) and $E_m \in U$. Across multiple iterations of the attack procedure and under different hypotheses concerning (H_G, A_G, U_G), the attacker's identity will eventually be revealed.

In actual scenarios, protocols are implemented through programs in the users' computers. Protocols with vulnerabilities have been in use in the past and it is easy to anticipate that this situation will occur in the future as well. Even some deployments of important protocol standards such as Kerberos or Single-Sign On have been shown to be vulnerable — but only after entering mass-scale use. Vulnerabilities are known to attackers or security engineers well before the awareness that the protocol is flawed reaches the users.

It is very difficult to force users to stop using a protocol as soon as a vulnerability is discovered. The more widespread the protocol, the more difficult it is to ensure that it quickly goes out of use. Two aspects are important: that every user (a) is informed of the new vulnerability and (b) takes action in switching to a secure protocol. Statistics on software upgrades are an unfortunate example of this type of issue.

By designing the user-end software to inform the user of a security failure whenever protocol-independent flags are raised, a guardian can help solve the notification issue as well as raise the likelihood that the user will take action and upgrade. When the weakness in the protocol is understood, it may be a cost-effective investment to design a guardian with an effective interference strategy, so as to facilitate restoring network security.

5 CONCLUSIONS

The traditional goal of protocol analysis is discovering attacks, to prompt replacing a vulnerable protocol with an improved and more secure one. Reductions are centered on attacks, either to reduce the search space for attacks (e.g. (Basin et al., 2011; Millen and Denker, 2002; Mödersheim et al., 2010)) or to reduce the number of agents (e.g. (Basin et al., 2011; Comon-Lundh and Cortier, 2003)). In particular, if there exists an attack involving *n* collaborating attackers, then there exists an "equivalent" attack involving only one. Within this perspective, it is known that *n*-DY attackers equal in attack power a single DY attacker, and that the same can be said of Machiavelli-type attackers (Syverson et al., 2000). As a result, an exhaustive search for attacks can be performed in a reduced-complexity model.

With vulnerable protocols, in a single-attacker situation there is no protocol-independent indicator that could be used by honest agents to become aware that security has been compromised. If there is a single attacker, no simple defense is possible and the protocol inevitably fails its security goals. On the other hand, by deploying an additional ad-hoc competitor (the guardian) as defense, in certain conditions we can successfully raise protocol-independent indicators of ongoing attacks and protect the system. Introducing an appropriate guardian procedure as soon as new attacks are discovered can mitigate the consequences of flawed protocols still being in use.

In the case study, we have shown a counterexample to the statement: "if there exists a defense against an attack in a 2-attacker scenario, then there exists an equivalent defense in a 1-attacker scenario". This statement mirrors the classical result on *n*-to-1 reducibility and the counterexample shows that exhaustive searches for (guardian-based) defenses cannot be carried out in reduced-complexity settings, as they require at least two attackers. Within our proposed approach, the goal of analysis is finding a strategy to defend the system against existing attacks, rather than identifying vulnerabilities to prompt redesigning the protocol. We would be performing protocol analysis to identify possible *defenses*, rather than attacks.

Future Work. Along the line of work presented in this paper, we have investigated an additional simple protocol, the Shamir-Rivest-Adleman three-pass protocol, which differs significantly from BME in that success is not necessarily exclusive. The case study is available as additional material in (Fiazza et al., 2011).

So far, we have formalized a framework for non-collaboration, described the notion of competitive attacker, shown a proof-of-concept result on concurrent attacks giving rise to interference, delineated a novel strategy for defending protocols and presented a proof-of-concept result on the effectiveness of a network guardian configured as a competitive attacker.

We are currently working on realizing an implementation of our framework for non-collaboration. One of the key issues is how to systematically generate competitive attack behaviors, given a vulnerable protocol and a base ("classical") attack. In the case studies we have explored so far, this step was addressed by taking the point of view of an attacker and observing our reasoning. The ability to construct competitive attack behaviors rests on our intuitive understanding of key features in both the protocol and the attack, as well as on our ability to reason at a high level of abstraction to anticipate the consequences of an action. Our chosen approach to enable protocol analysis for defense identification involves recruiting techniques from AI and robotics, fields that traditionally have a complex notion of agent.

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" (www.avantssar.eu) and by the FP7-ICT-2009-5 Project no. 257876, "SPaCIoS: Secure Provision and Consumption in the Internet of Services" (www.spacios.eu). We thank Davide Guardini for his constructive comments.

REFERENCES

Abadi, M., Blanchet, B., and Comon-Lundh, H. (2009). Models and proofs of protocol security: A progress report. In *Proceedings of CAV'09*, LNCS 5643, pages 35–49. Springer.

Arsac, W., Bella, G., Chantry, X., and Compagna, L. (2009). Validating Security Protocols under the General Attacker. In *Proceedings of ARSPA-WITS 2009*, LNCS 5511, pages 34–51. Springer.

Arsac, W., Bella, G., Chantry, X., and Compagna, L.

(2011). Multi-attacker protocol validation. *Journal of Automated Reasoning*, 46(3-4):353–388.

Basin, D., Caleiro, C., Ramos, J., and Viganò, L. (2011). Distributed temporal logic for the analysis of security protocol models. *Theoretical Computer Science*. To appear.

Basin, D., Capkun, S., Schaller, P., and Schmidt, B. (2009). Let's get physical: Models and methods for real-world security protocols. In *Proceedings of TPHOLs'09*, LNCS 5674, pages 1–22. Springer.

Basin, D. and Cremers, C. (2010). Modeling and analyzing security in the presence of compromising adversaries. In *Proceedings of ESORICS 2010*, LNCS 6345, pages 340–356. Springer.

Bella, G., Bistarelli, S., and Massacci, F. (2003). A protocol's life after attacks. In *Proceedings of 11th International Workshop on Security Protocols*, LNCS 3364, pages 3–18. Springer.

Bella, G., Bistarelli, S., and Massacci, F. (2008). Retaliation against protocol attacks. *Journal of Information Assurance and Security*, 3:313–325.

Boyd, C. and Mathuria, A. (2003). *Protocols for Authentication and Key Establishment*. Springer.

Caleiro, C., Viganò, L., and Basin, D. (2005). Metareasoning about security protocols using distributed temporal logic. *Electronic Notes in Theoretical Computer Science*, 125(1):67–89.

Caleiro, C., Viganò, L., and Basin, D. (2006). On the semantics of Alice & Bob specifications of security protocols. *Theoretical Computer Science*, 367(1-2):88 – 122.

Comon-Lundh, H. and Cortier, V. (2003). Security properties: two agents are sufficient. In *Proceedings of ESOP'2003*, LNCS 2618, pages 99–113. Springer.

Dilloway, C. and Lowe, G. (2007). On the specification of secure channels. In *Proceedings of WITS'07*.

Dolev, D. and Yao, A. C. (1983). On the security of public key protocols. *IEEE Trans. Inform. Theory*, 29(2):198–208.

Fiazza, M. C., Peroli, M., and Viganò, L. (2011). Attack Interference in Non-Collaborative Scenarios for Security Protocol Analysis (extended version). Available at www.arxiv.org.

Kamil, A. and Lowe, G. (2010). Specifying and modelling secure channels in strand spaces. In *Proceedings of FAST'09*, LNCS 5983, pages 233–247. Springer.

Millen, J. K. and Denker, G. (2002). Capsl and mucapsl. *Journal of Telecommunications and Information Technology*, 4:16–27.

Mödersheim, S., Viganò, L., and Basin, D. A. (2010). Constraint differentiation: Search-space reduction for the constraint-based analysis of security protocols. *Journal of Computer Security*, 18(4):575–618.

Schaller, P., Schmidt, B., Basin, D., and Capkun, S. (2009). Modeling and verifying physical properties of security protocols for wireless networks. In *Proceedings of CSF'09*. IEEE Computer Society.

Syverson, P., Meadows, C., and Cervesato, I. (2000). Dolev-Yao is no better than Machiavelli. In *Proceedings of WITS'00*, pages 87–92.

APPENDIX

In this appendix, we present a detailed view of the outcome of an attack carried out against BME and involving only the non-collaborative attackers E_1 and E_2. Refer to Section 3 for definitions of BME, attacker behavior against BME, attack traces and cases.

Note that in cases 1, 2 and 3 (shown in Table 6), E_j's request is the j-th served by S. In cases 4, 5 and 6, E_2 is the attacker with knowledge advantage. For clarity, for cases 4 and 6 (see Table 6) we mark as E_j* the case in which E_j's request is served *first* by S. In case 5, E_2's request is the only served and the distinction is unnecessary.

A competitive attacker E attacking BME can:

- succeed and compromise a key that A will use;
- fail and realize it (by timeout);
- fail without realizing it, by acquiring the wrong key;
- fail without realizing it, even though E acquired the right key.

Honest agents under attack can:

- detect the attack and abandon the protocol before carrying out step (3);
- realize that the key has been compromised and keep safe by not using it;
- fail to detect an attack but use their keys safely, because all attackers have failed to acquire the correct key;
- use a compromised key.

Attackers who realize their failure can infer the following:

α *Mislabeled or Unknown Attacker.* The attacker spies two messages from S and none from A in response; he deduces that A had opened a single session and that at least one request to S (in addition to his own) was an attack. The attacker realizes that he has either mislabeled as honest one of the active attackers or that an unknown competitor is active.

β *Unknown Attacker.* The attacker spies two messages from S and none from A in response; he deduces that A had opened a single session and that at least one request to S (in addition to his own) was an attack. However, he has seen no additional requests of the type (A,X) transit on the network; the attacker realizes that an unknown competitor is active on the network.

γ *Missed Message: Mislabeled or Unknown Attacker.* The attacker spies only one message from S

Table 6: Outcomes of a competitive attack against BME involving the attackers E_1 and E_2 and the honest initiator A. In cases 4 and 6 E_j*: E_j's request at step (1_i) is served by S first. Traces are described in Table 4; *canSee*$()$ describes the set of attackers who spy the message sent by A at step (3); for each role, we report the actual result of the attack (result), if the agent believes he has succeeded or failed (belief) and whether he has acquired the right key, the wrong key or no key at all (key). When attackers realize their failure, they can infer the reason for failing as shown in the column Detection; the honest agent A can detect ongoing attacks by receiving two answers from S or none. In the last column, we show the result of introducing a guardian agent playing the role in the corresponding row against an attacker playing the other role.

Trace	canSee	Agent	Result	Belief	Key	Detection		Guardian
T1						**Case 1**	**Case 3**	
	–	E_1	failure	failure	none	α	β	of help
		E_2	failure	failure	none	α	β	of help
		A	safe	attack	not used	2 keys	2 keys	
T2	**step (3)**					**Case 1**	**Case 3**	
	$\{E_1,E_2\}$	E_1	success	success	right	none	none	of help
		E_2	failure	success	wrong	none	none	no effect
		A	attacked	safe	broken	none	none	
	$\{E_1\}$	E_1	success	success	right	none	none	of help
		E_2	failure	failure	none	γ	γ	no effect
		A	attacked	safe	broken	none	none	
	$\{E_2\}$	E_1	failure	failure	none	γ	γ	of help
		E_2	failure	success	wrong	none	none	of help
		A	safe	safe	used	none	none	
T3	**step (3)**					**Case 1**	**Case 3**	
	$\{E_1,E_2\}$	E_1	failure	success	right	none	none	of help
		E_2	failure	success	wrong	none	none	of help
		A	safe	attack	not used	2 keys	2 keys	
	$\{E_1\}$	E_1	failure	success	right	none	none	of help
		E_2	failure	failure	none	γ	γ	of help
		A	safe	attack	not used	2 keys	2 keys	
	$\{E_2\}$	E_1	failure	failure	none	γ	γ	of help
		E_2	failure	success	wrong	none	none	of help
		A	safe	attack	not used	2 keys	2 keys	
T4	–					**Case 2**		
		E_1	failure	failure	none	correct understanding		of help
		E_2	failure	failure	none	correct understanding		of help
		A	safe	attack	none	no answer: DoS		
T5	**step (3)**					**Case 5**		
	$\{E_1,E_2\}$	E_1	failure	success	wrong	none		no effect
		E_2	success	success	right	correct understanding		of help
		A	attacked	safe	broken	none		
	$\{E_1\}$	E_1	failure	success	wrong	none		of help
		E_2	failure	failure	none	δ		of help
		A	safe	safe	in use	none		
	$\{E_2\}$	E_1	failure	failure	none	γ		no effect
		E_2	success	success	right	correct understanding		of help
		A	attacked	safe	broken	none		

Trace	canSee	Agent	Result	Belief	Key	Detection		Guardian
T1	–					**Case 4**	**Case 6**	
		E_1*	failure	failure	none	β	β	of help
		E_1	failure	failure	none	β	β	of help
		E_2*	failure	failure	none	α	ε	of help
		E_2	failure	failure	none	α	ε	of help
		A	safe	attack	not used	2 keys	2 keys	
T2	**step (3)**					**Case 4**	**Case 6**	
	$\{E_1,E_2\}$	E_1*	success	success	right	none	none	of help
		E_1	failure	success	wrong	none	none	no effect
		E_2*	success	success	right	none	none	of help
		E_2	failure	success	wrong	none	none	no effect
		A	attacked	safe	broken	none	none	
	$\{E_1\}$	E_1*	success	success	right	none	none	of help
		E_1	failure	success	wrong	none	none	of help
		E_2*	failure	failure	none	γ	ε	of help
		E_2	failure	failure	none	γ	ε	no effect
		A	attacked	safe	broken	none	none	
	$\{E_2\}$	E_1*	failure	failure	none	γ	γ	of help
		E_1	failure	failure	none	γ	γ	no effect
		E_2*	success	success	right	none	none	of help
		E_2	failure	success	wrong	none	none	of help
		A	safe	safe	used	none	none	
T3	**step (3)**					**Case 4**	**Case 6**	
	$\{E_1,E_2\}$	E_1*	failure	success	right	none	none	of help
		E_1	failure	success	wrong	none	none	of help
		E_2*	failure	success	right	none	none	of help
		E_2	failure	success	wrong	none	none	of help
		A	safe	attack	not used	2 keys	2 keys	
	$\{E_1\}$	E_1*	failure	success	right	none	none	of help
		E_1	failure	success	wrong	none	none	of help
		E_2*	failure	failure	none	γ	ε	of help
		E_2	failure	failure	none	γ	ε	of help
		A	safe	attack	not used	2 keys	2 keys	
	$\{E_2\}$	E_1*	failure	failure	none	γ	γ	of help
		E_1	failure	failure	none	γ	γ	of help
		E_2*	failure	success	right	none	none	of help
		E_2	failure	success	wrong	none	none	of help
		A	safe	attack	not used	2 keys	2 keys	

but no reply from A; all messages from S that successfully reach A are seen, so the attacker deduces that he has missed A's response. Thus, an active competitor (mislabeled or unknown) has erased it, preventing the attacker from acquiring it through the spy rule.

δ *Missed Message.* Similar to case γ. The attacker does not draw further conclusions because he is already aware of an active attacker that may have erased the message.

ε *Suspect Condemned.* The attacker E has put to test the dishonesty of an agent X in U_E (the suspect). Failing the attack is interpreted as a confirmation that the suspect is dishonest: X is placed into A_E.