Non-collaborative Attackers and How and Where to Defend Flawed Security Protocols (Extended Version)

Michele Peroli[1], Luca Viganò[2(✉)], and Matteo Zavatteri[1]

[1] Dipartimento di Informatica, Università di Verona, Verona, Italy
[2] Department of Informatics, King's College London, London, UK
luca.vigano@kcl.ac.uk

Abstract. Security protocols are often found to be flawed after their deployment. We present an approach that aims at the neutralization or mitigation of the attacks to flawed protocols: it avoids the complete dismissal of the interested protocol and allows honest agents to continue to use it until a corrected version is released. Our approach is based on the knowledge of the network topology, which we model as a graph, and on the consequent possibility of creating an interference to an ongoing attack of a Dolev-Yao attacker, by means of non-collaboration actuated by ad-hoc benign attackers that play the role of network guardians. Such guardians, positioned in strategical points of the network, have the task of monitoring the messages in transit and discovering at runtime, through particular types of inference, whether an attack is ongoing, interrupting the run of the protocol in the positive case. We study not only how but also where we can attempt to defend flawed security protocols: we investigate the different network topologies that make security protocol defense feasible and illustrate our approach by means of concrete examples.

Keywords: Security protocols · Defense · Non-collaborative attackers · Attack interference · Attack mitigation · Topological advantage

1 Introduction

1.1 Context and Motivation

Security protocols are often found to be flawed after their deployment, which typically requires "dismissing" the protocol and hurrying up with the deployment of a new version hoping to be faster than those attempting to exploit the

This work was partially supported by the EU FP7 Project no. 257876, "SPaCIoS: Secure Provision and Consumption in the Internet of Services" (www.spacios.eu) and the PRIN 2010-11 project "Security Horizons". Much of this work was carried out while Luca Viganò was at the Università di Verona.

© Springer International Publishing Switzerland 2014

discovered flaw. We present an approach that aims at the neutralization or mitigation of the attacks to flawed protocols: it avoids the complete dismissal of the interested protocol and gives honest agents the chance to continue to use it until a corrected version is released.

The standard attacker model adopted in security protocol analysis is the one of [12]: 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 (and the consequent possible novel flaws) introduced in the implementation phase, the protocol can thus be safely employed in real-life networks, at least in principle.

A number of tools have been proposed for automated security protocol analysis (e.g., [1,5,11,13,19,20] to name just a few), all of which follow the classical approach for security protocol analysis in which there is a finite number of honest agents and only one DY dishonest agent, given the implicit assumption that in order to find attacks we can reduce n collaborative DY attackers to 1 (for a proof of this assumption see, e.g., [2]).

In this paper, we take a quite different approach: we exploit the fact that if in the network there are *multiple non-collaborative attackers*, then the interactions between them make it impossible to reduce their attack "power" to that of a single attacker. This paper is based on the network suitable for the study of non-collaborative scenarios defined in our previous works [14,15], in which we introduced a protocol-independent model for non-collaboration for the analysis of security protocols (inspired by the exploratory works [3,4] for "protocol life after attacks" and attack retaliation). In this model: (i) a protocol is run in the presence of multiple attackers, and (ii) attackers potentially have different capabilities, different knowledge and do not collaborate but rather may interfere with each other.

Interference between attackers has spawned the definition of an ad hoc attacker, called *guardian*, as a defense mechanism for flawed protocols: if two non-collaborative attackers can interfere with each other, then we can exploit this interference to neutralize or at least mitigate an ongoing attack (a detailed cost-effective analysis of this approach is left for future work).[1]

There is one fundamental catch, though. We know that a DY attacker actually cannot exist (e.g., how could he control the whole network?) but postulating

[1] It is interesting to note how this idea of "living with flaws" is becoming more and more widespread; see, e.g., [9] where runtime monitors are employed to warn users of android applications about "man in the middle" attacks on flawed implementations of SSL. Our approach is also related to signature-based intrusion detection systems, but we leave the detailed study of the relations of our approach with runtime monitors and signature-based intrusion detection systems for future work.

his existence allows us to consider the worst case analysis so that if we can prove a protocol secure under such an attacker, then we are guaranteed that the protocol will be secure also in the presence of weaker, more realistic attackers. A guardian, however, only makes sense if it really exists, i.e., if it is implemented to defend flawed protocols for real, but the attackers and the guardian presented in [14,15] are modeled in order to discover interactions between agents in non-collaborative scenarios rather than pushing for an implementation in the real-world.

1.2 Contributions

Since implementing a guardian with the full power of a DY attacker is impossible, we must investigate ways to make the guardian more feasible. In order to reduce the complexity of the possible implementation of such a defense mechanism, in this paper we relax the notion of guardian and ask him to defend only a subset of the communication channels of the network, which we put under his control.

Furthermore, not being obviously able to know where the competitor is, we investigate where we have to introduce this defense mechanism in the network from a topological perspective, i.e., how the guardian can dominate his competitor(s).[2] Modeling the network as a graph, we study how the topological position of an attacker E and a guardian G, with respect to each other and to honest agents of the protocol, can influence a protocol attack and, thus, the possible defense against it. We define six basic topological configurations and study the outcome of the introduction of a guardian in each specific position. We also introduce the concept of *topological advantage*, which guarantees that the guardian has an advantage with respect to his competitors, and can thus carry out inference on messages in transit in order to detect an ongoing attack and eventually mitigate or neutralize it.

The contributions of this paper thus extend, and in a sense are complementary to, the ones in our previous works [14,15]. In a nutshell: there we discussed the *how* we can defend flawed security protocols and here we discuss the *where*. More specifically, as we will describe in the following sections, in [14,15], we put the basis for the study of the interaction of two attackers in non-collaborative scenarios with the goal of understanding and finding the types of interference the guardian can use, and, in this paper, we give the means to understand how to exploit the interference from a topological point of view, thus bringing the guardian close to real implementation.

1.3 Organization

We proceed as follows. In Sect. 2, we summarize the main notions of attack interference in non-collaborative scenarios. In Sect. 3, we formalize the models of the network and of the guardian, with particular emphasis on the topological advantage that a guardian must have in order to defend against attacks. In Sect. 4,

[2] In the following, we focus on one competitor (i.e., one attacker), but it is quite straightforwardly possible to extend our work to multiple competitors.

we discuss, as a detailed proof-of-concept, how we can defend the ISO-SC 27 protocol and summarize the results we obtained for other case studies. In Sect. 5, we briefly summarize our results and discuss future work.

2 Attack Interference in Non-collaborative Networks

2.1 Network Agents

Let *Agents* be the set of all the network agents, which comprises of two disjoint subsets:

– the subset *Honest* of *honest agents* who always follow the steps of the security protocol they are executing in the hope of achieving the properties for which the protocol has been designed (such as authentication and secrecy), and
– the subset *Dishonest* of *dishonest agents* (a.k.a. *attackers*) who may eventually not follow the protocol to attack some (or all) security properties. In addition to being able to act as legitimate agents of the network, dishonest agents typically have far more capabilities than honest agents and follow the model of Dolev-Yao [12] that we summarized in the introduction.

The *knowledge* of an honest agent X is characterized by a proprietary dataset D_X, which contains all the information that X acquired during the protocol execution, and is closed under all cryptographic operations on message terms (e.g., an agent can decrypt an encrypted message that he knows provided that he knows also the corresponding decryption key). D_X is monotonic since an agent does not forget.

2.2 DY Attackers and the Network in a Non-collaborative Scenario

In this paper, we take the non-classical approach that leverages on the fact that the interactions between multiple non-collaborative attackers may lead to interference. We base our work on the network suitable for the study of non-collaborative scenarios defined in [14,15], which we now summarize quickly pointing to these two papers for more details.

Table 1 shows the model that we adopt to formalize a DY attacker E in a non-collaborative scenario in which different attacks may interfere with each other (we restrict the study of this type of interaction to two active attackers but it can be generalized to multiple ones). The knowledge base of E is encoded in the set D_E, whereas D_{net} is the proprietary dataset for the network (we will return to the network model below). The rules in the table describe the operations that an attacker can perform internally, how he can interact with the network and how the system (i.e., the network environment) is configured. It is important to note that the rules in Table 1 are transition rules rather than deduction rules, i.e., they describe knowledge acquisition from a given

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.

Composition:

$$\frac{m_1 \in D_E^i \quad m_2 \in D_E^i}{(m_1, m_2) \in D_E^i}$$

Encryption:

$$\frac{m \in D_E^i \quad k \in D_E^i}{\{m\}_k \in D_E^i}$$

Projection:

$$\frac{(m_1, m_2) \in D_E^i}{m_j \in D_E^i \text{ for } j \in \{1,2\}}$$

Decryption:

$$\frac{\{m\}_k \in D_E^i \quad k^{-1} \in D_E^i}{m \in D_E^i}$$

Inflow-Spy:

$$\frac{\mu \in D_{net}^i \quad \mathit{ofInterest}_E(X) \quad Y \in D_E^i \quad \psi}{m \in D_E^{i+1} \wedge \mathit{sender}(<X, m, Y>) \in D_E^{i+1}}$$

Outflow-Spy:

$$\frac{\mu \in D_{net}^i \quad \mathit{sender}(\mu) \in D_E^i \quad \mathit{ofInterest}_E(Y) \quad \psi}{m \in D_E^{i+1} \wedge Y \in D_E^{i+1}}$$

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

Injection:

$$\frac{m \in D_E^i \quad X \in D_E^i \quad Y \in D_E^i}{<E(X), m, Y> \in D_{net}^{i+1}}$$

Erase:

$$\frac{<X, m, Y> \in D_{net}^i \quad \mathit{sender}(<X, m, Y>) \in D_E^i}{<X, m, Y> \notin D_{net}^{i+1}}$$

True-sender-ID:

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

DecisionalProcess:

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

NetHandler:

$$\mathit{canSee}(<X, m, Y>, i)) = \{Z \in \mathit{Dishonest} \mid Z \text{ can spy } <X, m, Y> \text{ on } D_{net}^i\}$$

operation and a particular configuration rather than the reasoning about "only" the knowledge of the attacker.

As in the classic DY case, an attacker in this model can *send* and *receive* messages, derive new messages by *composing, decomposing, modifying, encrypting/decrypting* known messages (iff he has the right keys), and *intercept* or *remove* messages from the network. An attacker E may also masquerade as (i.e., impersonate) another agent X, which we denote by writing $E(X)$.

The most significant features of the attacker abilities are the two *spy* rules, which formalize the fact that attackers only pay attention to a selection of the traffic on the network (considering only selected target agents):[3,4]

– ***Inflow-Spy*:** the attacker pays attention to the incoming network traffic of a target agent and saves the identifiers of the sender agents,
– ***Outflow-Spy*:** the attacker pays attention to the traffic generated by a target agent.

The *target agent* X of the two spy-rules is defined through a decisional process (the function $\mathit{ofInterest}_E(X)$ in Table 1) in which each attacker decides if the traffic to/from the agent X is worth to be followed. This decision is made at run-time when a new agent identifier is discovered over the network (i.e., when a new agent starts sending messages on the channel monitored by the attacker). In this paper, we do not go into the details of how his decision is actually taken, but different strategies might be devised and we will investigate them in future work.

The network net is also formalized through a dataset, D_{net}, which is changed by the *actions* send, receive, inject and erase a message. We write D^i_{net} to denote the state of D_{net} after the i-th action. Messages transit on the network in the form of triplets of the type

$$\langle \mathit{sender\text{-}ID}, \mathit{message}, \mathit{receiver\text{-}ID} \rangle,$$

where, as in the classical approaches, both the attackers and the agents acquire knowledge only from the body of messages, i.e., *sender-ID* and *receiver-ID* are actually hidden to them and only used by the network system. As a consequence of message delivery or deletion, D_{net} is non-monotonic by construction.

In order to regulate the concurrent actions over the network, the model comprises a *NetHandler* whose task is to handle the network by selecting the next action and implementing the dependencies between selected actions and knowledge available to each attacker. That is, *NetHandler*: (i) notifies agents that the state of the network has changed with newly-inserted messages, (ii) polls agents for their next intended action, (iii) selects from the set of candidate actions the one that will be actually carried out, and (iv) informs agents of whether the computation they performed to propose an action is a consequence of a message that they did not have access to (i.e., for these agents a rollback might occur in

[3] If an attacker were omniscient and omnipotent (i.e., if he were to control the whole network) then there'd actually be no “space” for another attacker, and thus there'd be no interference. The more “adventurous” reader may want to compare this with the proof of the uniqueness of God by Leibniz, which was based on the arguments started by Anselm of Canterbury and was later further refined by Gödel.

[4] In this paper we only use the *inflow-spy* and the *outflow-spy* filters and not the *restricted-spy* filter used in the previous exploratory works. This is due to the fact that we can certainly know who we want to defend, but we cannot know who the attackers are and we want to have the possibility of intercepting *all* outgoing/incoming messages which leave/come from/to an agent X.

Table 2. The ISO-SC 27 protocol and a parallel session attack against it.

ISO-SC 27 protocol
(1) $A \to B : N_A$
(2) $B \to A : \{\!|N_A, N_B|\!\}_{K_{AB}}$
(3) $A \to B : N_B$

Attack
(1.1) $A \to E(B) : N_A$
(2.1) $E(B) \to A : N_A$
(2.2) $A \to E(B) : \{\!|N_A, N'_A|\!\}_{K_{AB}}$
(1.2) $E(B) \to A : \{\!|N_A, N'_A|\!\}_{K_{AB}}$
(1.3) $A \to E(B) : N'_A$
(2.3) $E(B) \to A : N'_A$

which all knowledge gained since the last confirmed action is deleted from the dataset, and internal operations that have occurred are cancelled).

The outcome of the process governed by the network handler is described through the function *canSee*, which returns a subset of dishonest agents, highlighting the identifier of attackers who can spy "before" the message is erased from D_{net}. In other words, when a message is deleted from the network, the network handler, through the function *canSee*, can decide if an attacker has spied (and saved in his dataset) the message or not. In our previous work we had the possibility of spying a message before its deletion (in this case, the attacker has to decide if the message has been received by the honest agent or deleted by another attacker) but in this paper we relax this assumption and decide that when a message is spied it remains in the dataset of the attacker. The function *canSee* is a configurable parameter of our network and it corresponds to configuring a particular network environment in which the agents are immersed: *canSee* is instantiated by the security analyst at the beginning of the analysis in order to model time-dependent accessibility, strategic decision-making and information-sharing, or to capture a particular network topology (in our framework the function *canSee* is necessary in order to model the topologies that we will introduce in Sect. 3.1).

2.3 Attack Interference (In the Case of the ISO-SC 27 Protocol)

As a concrete, albeit simple, example of security protocol, Table 2 shows the ISO-SC 27 protocol [16], which aims to achieve entity authentication (aliveness) between two honest agents A and B, by exchanging nonces, under the assumption that they already share a symmetric key K_{AB}. Since in the second message there is nothing that assures that the message actually comes from B, the protocol is subject to a parallel sessions attack (also shown in the table) in which the attacker E, who does not know K_{AB}, uses A as oracle against herself in order to provide to her a response that he cannot generate by himself: E masquerades as B intercepting A's first message and sending it back to her in a parallel session (messages (1.1) and (2.1)). When A receives the first message of the protocol from E, she thinks someone wants to talk with her in another instance of the protocol (she does not control the nonce), thus she replies to E generating another nonce N'_A and encrypting it together with N_A (message (2.2)). Now E has got everything he needs in order to complete the attack to the protocol (messages (1.2)). The last message is not mandatory as the session has already been attacked, thus E can omit it (message (2.3)). At the end of the protocol runs, A is fooled into believing that $E(B)$ is B.

If a protocol is flawed, a single DY attacker will succeed with certainty. However, if attacks to the same protocol are carried out in a more complex network environment, then success is not guaranteed since multiple non-collaborative attackers may interact, and actually interfere, with each other. The results of [14,15] show that it is possible, at least theoretically, to exploit interference between two non-collaborative attackers to mitigate protocol flaws, thus providing a form of defense to flawed protocols.

In the case of ISO-SC 27 protocol, which was not studied in [14,15][5], we can identify six cases for the possible interaction between two non-collaborative attackers E_1 and E_2:

1. E_1 and E_2 know each other as honest.
2. E_1 and E_2 know each other as attackers.
3. E_1 and E_2 are unaware of each other.
4. E_2 knows E_1 as honest.
5. E_2 knows E_1 as dishonest.
6. E_2 knows E_1, but he is unsure of E_1's honesty.

The traces corresponding to the interactions of E_1 and E_2 attacking the protocol are shown in Table 3. Attack traces of this type lead to three possible (mutually exclusive) situations: (i) E_1 dominates E_2 (i.e., E_1's attack succeeds while E_2's fails), or (ii) none of their attacks has success, or (iii) both achieve a situation of uncertainty, i.e., they do not know if their attacks have been successful or not.

In order to exploit the interference generated by multiple dishonest agents attacking the same protocol, we can construct an additional, but this time non-malicious, attacker: the *guardian* G.

To define the guardian as a network agent, we refine the previous definition of *Agents* to consider the subset of *benign dishonest agents*, i.e., $BenignDishonest \subseteq Dishonest \subseteq Agents$, where $X \in BenignDishonest$ means that X has attacker capabilities and may not follow the protocol but he "attacks" with the goal of "defending" the security properties not of attacking them. In other words:

Definition 1 (Guardian). *A* guardian *is a benign dishonest agent of the network, transparent to the other agents, whose main task is to establish a partial (or total) defense mechanism in order to mitigate (or neutralize) protocol attacks*

[5] In [14,15], we analyzed two protocols: (i) a key transport protocol described as an example in [6], which we thus called the Boyd-Mathuria Example (BME), and (ii) the Shamir-Rivest-Adleman Three-Pass protocol(SRA3P [8]), which has been proposed to transmit data securely on insecure channels, bypassing the difficulties connected to the absence of prior agreements between the agents A and B to establish a shared key.

Table 3. Traces for non-collaborative attacks against the ISO-SC 27. Traces are exhaustive: E_1 and E_2 have priority over honest agents. Arrows: relative order between $(2.1')$ and $(2.1'')$ is irrelevant in determining the outcome.

T1: cases 1, 3, 4	**T2: case 5**
$(1.1)\ A \rightarrow E_{1,2}(B) : N_A$	$(1.1\ )\ A \rightarrow E_{1,2}(B) : N_A$
$(2.1)\ E_{1,2}(B) \rightarrow A : N_A$	$\downarrow (2.1'\ )\ E_1(B) \rightarrow E_2(A) : N_A$
$(2.2)\ A \rightarrow E_{1,2}(B) : \{\lvert N_A, N'_A \rvert\}_{K_{AB}}$	$\uparrow (2.1'')\ E_2(B) \rightarrow A : N_A$
$(1.2)\ E_{1,2}(B) \rightarrow A : \{\lvert N_A, N'_A \rvert\}_{K_{AB}}$	$(2.2\ )\ A \rightarrow E_2(B) : \{\lvert N_A, N'_A \rvert\}_{K_{AB}}$
$(1.3)\ A \rightarrow E_{1,2}(B) : N'_A$	$(1.2\ )\ E_2(B) \rightarrow A : \{\lvert N_A, N'_A \rvert\}_{K_{AB}}$
$(2.3)\ E_{1,2}(B) \rightarrow A : N'_A$	$(1.3\ )\ A \rightarrow E_2(B) : N'_A$
	$(2.3\ )\ E_2(B) \rightarrow A : N'_A$

T3: case 2	**T4: case 6**
$(1.1\ )\ A \rightarrow E_{1,2}(B) : N_A$	$(1.1)\ A \rightarrow E_{1,2}(B) : N_A$
$\downarrow (2.1'\ )\ E_1(B) \rightarrow E_2(A) : N_A$	$(2.1)\ E_1(B) \rightarrow A : N_A$
$\uparrow (2.1'')\ E_2(B) \rightarrow E_1(A) : N_A$	+ steps of case 5

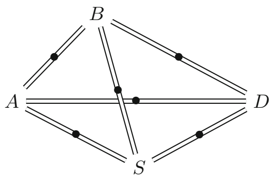
at execution time by means of attack-interference in non-collaborative scenarios. G is transparent to honest agents during their execution and becomes "visible" only in the case he has to report an ongoing attack.

3 Modeling the Network and the Guardian

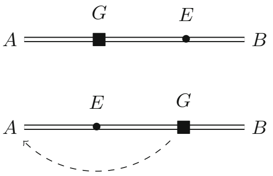
In the previous section, we have seen how the interaction between multiple non-collaborative dishonest agents attacking the same protocol can interfere with both attacks, thus providing a form of defense. As we remarked in the introduction, even if the idea of having a guardian defending honest agents from attacks seems thrilling, the existence of a guardian agent makes sense only with his implementation in the real world. In order to reduce the complexity of such an implementation, we will now investigate where we have to introduce this defense mechanism in the network from a topological perspective (i.e., how the guardian can dominate his competitor(s)). Modeling the network as a graph, we study how the topological position of an attacker E and a guardian G, with respect to each other and to honest agents of the protocol, can influence a protocol attack and, thus, the possible defense against it.

We say that the outcome of the introduction of the guardian on the network for a particular protocol yields a:

- *false positive* if, for some reason, a normal run of the protocol is considered as an attack,
- *false negative* if, for some reason, an attack is considered as a normal run of the protocol,



(a) An example of network as a graph; vertices represent agents, edges represent communication channels and the bullets • represent the presence of a DY-attacker E.



(b) Two possible allocations, on a channel between A and B that is controlled by an attacker E, for the guardian G when he defends an honest agent A. For both cases (the above one is implicit), we assume the presence of an authentic and resilient communication channel between G and A (dashed line).

Fig. 1. Model of the network and possible allocations of the guardian on a channel.

- *partial defense* iff it admits false negatives,
- *total defense* iff it does not admit false negatives.

Our objective is to realize a defense mechanism that admits as few false negatives as possible, while limiting also the number of false positives, by investigating the position that gives the guardian a topological advantage (see Definition 4 of defense mechanism and the ensuing Theorem 1).

3.1 A Network for Topological Advantage

We model the network as a graph (an example is depicted in Fig. 1a), where vertices represent the agents of the network and edges represent communication channels (we assume no properties of these channels, which are standard insecure channels over which messages are sent as specified by the security protocols). Since, as we remarked above, it would be unfeasible for the guardian to defend the traffic on all network channels, we investigate which of these channels the guardian should be best positioned on.

Security protocols typically involve two honest agents A and B, who sometimes enroll also a honest and trusted third party S (we could, of course, consider protocols with more agents). As depicted in Fig. 1a, the DY-attacker E is in control of all the communication channels of the network, thus, in the case of a ping-pong protocol between A and B, E controls also the communication channel between A and B. If we were to allocate a guardian G on such a channel in order to defend the honest agent A, it could only be in one of two locations: as shown in Fig. 1b, either the guardian G is between the initiator A and the attacker E, or G is between the attacker E and the responder B. In the following, these two cases will be used as a base of network topologies to be considered during the analysis. We will see in the next section that the guardian should have the possibility of alerting A of the ongoing attack without being detected by the attacker; in such a case (especially as highlighted in the lower topology in Fig. 1b), we thus assume the presence of an authentic and resilient communication channel (confidentiality can be enforced but it is not mandatory) between G and A.[6] In the following, this channel will be omitted from the notation and the figures for the sake of readability.

If the network topologies for two-agent protocols are simple (Fig. 2a and b), for the case where a trusted third party S (or another agent) is present on the network, we have to make some assumptions about the position of the attacker E (the attack power of the attacker is never questioned). In this paper, we consider four main base cases of network topologies for three-agent protocols, where, for every case, we consider which channel(s) the guardian is defending:

- Fig. 2c: the channel between A and S (we assume that the attacker is not present over these channels[7] and the guardian acts like a proxy),
- Fig. 2d: the channel between B and S (this is the specular scenario with respect to the previous case),
- Fig. 2e: A's communication channel (the guardian acts as a proxy for A), and
- Fig. 2f: B's communication channel (the guardian acts as a proxy for B).

These basic topologies abstract the communication channels of a complex network (e.g., a LAN) in a way that permits one to reason about the position of agents without introducing additional parameters in the process (e.g., additional agents that start the protocol at the same time, or multiple network paths relaxed in one link).

In general, we cannot state that a base case is the right one or the wrong one as this actually depends on both the analyzed protocol and the agent we want to defend. In order to implement the right guardian, we should consider the protocol defense possible in each of these cases. We conjecture that all other network topologies with two or three agents can be reduced to the base cases introduced above, but leave a formal proof for future work.

3.2 Network Guardian in Practice

Attacks leverage protocol-dependent features, and thus attack traces always contain particular messages that we can use as signals for ongoing attacks. As messages transit continuously through the network, we assume that the guardian has a way to distinguish them (otherwise, we cannot guarantee any type of defense).

[6] This channel could be a digital or a physical channel, say a text message sent to a mobile (as in some two-factor authentication or e-banking systems), a phone call (as in burglar alarm systems), or even a flag raised (as is done on some beaches to signal the presence of sharks). We do not investigate the features of this channel further but simply assume, as done in all the above three examples of runtime guarding (monitoring) systems, that such a channel actually exists.

[7] We do not make assumptions on the real topology of the network between A and S (i.e., there could be more than one channel) but only consider the fact that the communications from E are received by G.

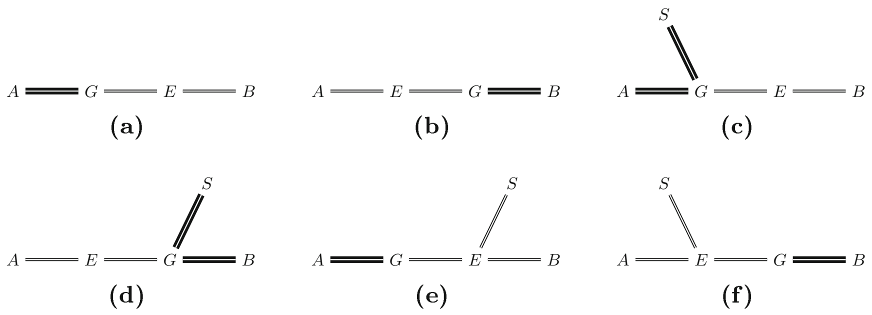


Fig. 2. Base cases of network topologies for protocols between two agents (a, b) and three agents (c, d, e, f). We denote with double stretched lines (in boldface) the channels for which we assume that the attacker is not present.

In order to operate, the network guardian needs to interact with the messages transiting over the network. The two modules that we define in the architecture of the guardian are: (i) the *Identification Module*, and (ii) the *Control Module*. Both modules operate separately, do not interact with each other (even though they share the guardian's dataset D_G), and are meant to (i) distinguish the messages that belong to the protocol[8] that they are defending and (ii) detect ongoing attacks.

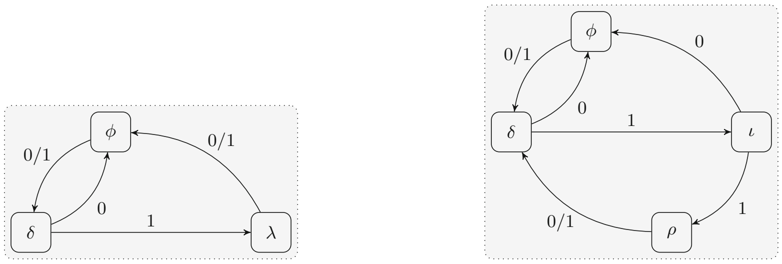
These features are achieved by means of two *distinguishers* Δ_{Id} and Δ_C, two probabilistic polynomial time algorithms. Δ_{Id} returns 1 if it believes that a message m belongs to the protocol and 0 otherwise. We use the distinguisher Δ_C in order to detect, from the run of a security protocol $\mathcal{P}$ (identified by the other module), those messages m that are considered *critical*, i.e., that can be used to attack $\mathcal{P}$.

For a concrete example of critical message, we can refer to Table 2. The nonce N_A exchanged in message (1.1) is the first information that the attacker uses in order to perform the reply attack against the ISO-SC 27 protocol, so this message must be considered critical. Even though the nonce is sent as a plaintext, the use of the distinguisher Δ_C overcomes the problem with encrypted messages.

Identification Module. Figure 3a shows the graphical representation of the *Identification Module*. The guardian uses this module, together with the distinguisher Δ_{Id}, to detect those messages m that belong to the protocol and label them as part of $\mathcal{P}$ in the dataset D_G in order to do inference subsequently.

We can see the Identification Module as a finite state machine where the transition from state to state depends on the spied messages. When a message

[8] We deliberately wrote "protocol" instead of "protocols" since, for now, we are not going to consider multi-protocol attacks or protocol composition, e.g., [7,10,17]. As future work, we envision a distinguisher able to distinguish between messages belonging to different protocols and thus consider also the attacks that occur when messages from one protocol may be confused with messages from another protocol.



(a) Identification Module: Assuming that m is the message spied from the spy filter, δ is the state where the distinguisher is invoked on input m, ϕ the "forward state", λ the "label state" in which the message m is labeled in the dataset D_G as part of the protocol $\mathcal{P}$.

(b) Control Module: Assuming that m is the message spied from the spy filter, δ is the state where the distinguisher is invoked on input m, ι the state where the attack invariant is invoked on m, ϕ represents the "forward state", ρ the "interference state".

Fig. 3. Identification and Control Modules implemented in the guardian.

m is spied by the spy filter (see Table 1 for the two available spy filters), the Identification Module of the guardian invokes the distinguisher $\varDelta_{Id}(m)$ to establish whether the message belongs to the protocol or not.

If $\varDelta_{Id}(m) = 0$, the message is not considered useful and the guardian moves to the forward state ϕ, which will let the message go, and subsequently goes back, without checking any condition, to the initial state δ in order to wait for the next message. If $\varDelta_{Id}(m) = 1$, then m belongs to the protocol and the guardian moves to the "identification state" λ, where the message is labeled in the dataset D_G. After the message has been labeled, the Identification Module goes back to the initial state δ in order to wait for the next message.

From now on, when we do an operation (spy-filters excluded) on the dataset, we mean (slightly abusing notation) the subset of the labeled messages.

Control Module. Figure 3b shows the graphical representation of the Control Module. The guardian uses this module, together with the distinguisher $\varDelta_C$, in order to deal with those messages m that he must control in order to be able to do inference (i.e., check if an attack is ongoing) and eventually interfere with the attacker; we call these messages *critical.*

Once the distinguisher, implemented in the Control Module, believes that m is critical (at time i), the *attack invariant* $Inv(m, i)$ is tested to discover (or exclude) an ongoing attack. $Inv(m, i)$ is a protocol-dependent Boolean condition; formally, it is a first-order logic formula on a critical message of the protocol (which can be straightforwardly extended to a set of messages) tested at time i (i.e., after i actions on the dataset D_{net}; in order to define more complex functions, more than two parameters can be used):

$$Inv(m,i) = \begin{cases} 1 & \text{if } m\text{, at time } i\text{, characterizes an ongoing attack or a false positive} \\ 0 & \text{if } m\text{, at time } i\text{, characterizes a normal run or a false negative} \end{cases}$$

If the computation of the invariant returns 1, then the guardian G carries out the appropriate defense for the attack making the victim abort the current run of the protocol and, eventually, mislead the attacker and/or induce him to abort the attack. We give an example of invariant in Sect. 4.1 when we return to our case study.

When a message m is spied by the spy filter, the Control Module is in the initial state δ, and then the message is passed as input to the distinguisher Δ_C, whose task is to establish whether the message is critical or not. If the result of the distinguisher is $\Delta_C(m) = 0$, the message is not considered critical and the guardian moves to the forward state ϕ, which will let the message go, and subsequently goes back, without checking any condition, to the initial state δ in order to wait for the next message. Instead, if $\Delta_C(m) = 1$, then a critical message has just been distinguished from the others; the guardian moves to the invariant state ι passing the message as input to the attack invariant formula $Inv(m,i)$, whose task is to establish whether an attack is actually ongoing or not (the invariant is computed using the labeled messages in D_G respecting the temporal constraints). If $Inv(m,i) = 0$, then either an attack is not ongoing or a false negative has just happened (i.e., the defense mechanism is partial); thus, the guardian goes to the forward state ϕ, which will let the message go, and subsequently goes back without checking any condition to the initial state δ. Instead, if $Inv(m,i) = 1$ either an attack is ongoing or a false positive has just happened, independently of the used defense mechanism; thus, the guardian moves to interference state ρ to carry out the appropriate countermeasures and subsequently goes back, without checking any condition, to the initial state.

As the Δ_{Id} is needed in order to detect the messages that belong to the protocol $\mathcal{P}$, we envision Δ_C to be useful in the case of protocols with a large number of messages in order to lighten the computation load of $Inv(m,i)$, i.e., we compute $Inv(m,i)$ on a subset of the protocol messages:

$$Critical \quad \subseteq \quad \mathcal{P}_{labeled} \quad \subseteq \quad Messages$$

where *Messages* are all the messages saved in the dataset by a spy-filter, $\mathcal{P}_{labeled}$ are the messages that Δ_{Id} labeled as part of the protocol $\mathcal{P}$ and *Critical* are the messages that Δ_C believes may be used to attack $\mathcal{P}$.

3.3 Topological Advantage

To defend protocols against attacks, a guardian should be “near” one of the agents involved in the protocol executions; otherwise the guardian could be useless: if he does not see (and thus cannot control) messages belonging to the protocol in transit from these agents, then he cannot carry out the interference/defense.

Definition 2 (Topological Advantage). *Let $X \in Agents$ be the agent that the guardian $G \in BenignDishonest$ is defending in a particular protocol (with set Messages of messages), and $Y \in Agents$ the other agent. We say that G is in* topological advantage *with respect to the attacker E if*

$$\begin{aligned}&\forall m \in Messages.\ \exists i \in \mathbb{N}.\\ &\qquad G \in canSee(<X, m, Y>, i)) \ \vee\ G \in canSee(<Y, m, X>, i)) \ \vee\\ &\qquad G \in canSee(<E(X), m, Y>, i)) \ \vee\ G \in canSee(<Y, m, E(X)>, i))\end{aligned}$$

Definition 2 states that for a guardian to be in topological advantage, he must be collocated over the network in one of the configurations of Fig. 2 so that he can spy (and eventually modify) all the transiting messages to and/or from the agent that he is defending, even in the case that they are forged.

In order to define what a defense mechanism is, we have to formalize how an attack can be formalized based on a parametric function that the attacker computes during his execution.

Let $E \in Dishonest$, $X \in Honest$, s be the number of steps composing the attack trace, m_s the message spied over the network or present in the attacker dataset D_E at step s, $Func = \{Erase, Injection, Duplicate, \ldots\}$ a set of functionalities that E can use on the messages. Note that the names of the functionalities quite intuitively denote their meaning; not all of the functionalities are used in this paper and many more could be defined. The functionalities in *Func* have domain in the messages belonging to a given protocol, whereas the codomain is defined as the union of all the possible transformations of the messages in the domain that give (i) messages "acceptable" by the protocol (i.e., that can be sent/received by the protocol's agents) or (ii) an empty message. The codomain is thus a set of messages. We use $func_s$ to denote a functionality in *Func* used at step s.

Definition 3 (Attack Function). *The attack function $f(m, s)$ selects a functionality $func_s$ to be used on the message m at step s and returns the result of the $func_s$ with argument m ($func_s(m)$).*

As a concrete example, the attack function of the attack in Table 2 is reported in Table 4.

Of course, more complex attack functions could (and sometimes even should) be defined, especially for more complex protocols. Since the attack function is but one parameter, we believe that our definitions and results are general enough and can be quite easily adapted to such more complex functions.

Having formalized how an attack can be seen as a parametric function, we can also assume the existence of an inverse function $f^{-1}(m, s)$ of the attack function (i.e., the function that from a message m such that $m = f(m', s)$, and a step s, computes m'). In this paper, we will not discuss how to formalize the inverse attack function; we leave a definition for future work and for now assume that, during the implementation of the framework, a security analyst can take care of this matter.

Table 4. Example of attack function for a parallel session attack against the ISO-SC 27 protocol.

s	m	$func_s$	$f(s,m)$
1	N_A	*Erase*	$\emptyset$
2	N_A	*Injection*	N_A
3	$\{\lvert N_A, N'_A \rvert\}_{K_{AB}}$	*Erase*	$\emptyset$
4	$\{\lvert N_A, N'_A \rvert\}_{K_{AB}}$	*Injection*	$\{\lvert N_A, N'_A \rvert\}_{K_{AB}}$
5	N'_A	*Erase*	$\emptyset$
6	N'_A	*Injection*	N'_A

Definition 4 (Defense Mechanism). *Let $X \in Agents$ be the agent that the guardian $G \in BenignDishonest$ is defending in a particular protocol (with set $Critical$ of critical messages), let $E \in Dishonest$ be the attacker, and s be the number of steps composing E's attack trace. We say that G is a* defense mechanism *if he knows E's attack function $f(m, s)$ and can compute the inverse function $f^{-1}(m, s)$ in order to enforce the following:*

$$\nexists m \in Critical.\, \forall i \in \mathbb{N}.\ \exists p, j \in \mathbb{N}.\ j > i \ \wedge\ 1 \leq p \leq s\ \wedge$$
$$m \in D^i_{net}\ \wedge f^{-1}(f(m,p),p) = m\ \wedge$$
$$(G \notin canSee(<E, f(m,s), X>, j)) \vee G \notin canSee(<E(Y), f(m,s), X>, j)))$$

If G can compute the inverse attack function, then G has knowledge of the possible attacks against the protocol carried out through the attack function and can detect the critical messages even if the attacker modifies/deletes them.

Thus, we can state the following theorem (which can be quite straightforwardly generalized to multiple attackers):

Theorem 1. *A guardian $G \in BenignDishonest$ is a defense mechanism for an agent $X \in Agents$ in a protocol $\mathcal{P}$, if he is in topological advantage with respect to an attacker $E \in Dishonest$ who is attacking X in $\mathcal{P}$.*

As a proof sketch, let $X \in Agents$ be the agent that G is defending, $Y \in Agents$, $E \in Dishonest$ with attack function $f(m,p)$, $m \in Critical$, f^{-1} known to G, G in topological advantage with respect to the attacker E, s the number of steps composing E's attack trace, and $1 \leq p \leq s$. Then, since $f(m,p) \in Messages$, we have that: $\exists i \in \mathbb{N}.\ G \in canSee(<X, f(m,p), Y>, i)) \ \vee\ G \in canSee(<Y, f(m,p), X>, i)) \ \vee\ G \in canSee(<E(X), f(m,p), Y>, i)) \ \vee\ G \in canSee(<Y, f(m,p), E(X)>, i))$. In order to have a defense mechanism, we have to enforce the following: $\nexists m \in Critical.\ \forall i \in \mathbb{N}.\ \exists p, j \in \mathbb{N}.\ j > i \ \wedge\ 1 \leq p \leq s \wedge m \in D^i_{net} \ \wedge\ G \notin canSee(<E, f(m,p), X>, j)) \wedge f^{-1}(f(m,p),p) = m$. Since $f(m,p) \in Critical \subseteq Messages$, only $f^{-1}(f(m,p),p) = m$ must be enforced, but it is known to G by assumption.

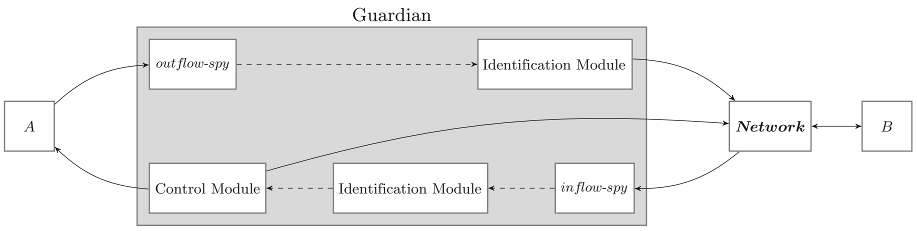


Fig. 4. Guardian configuration for the ISO-SC 27 protocol. With a dashed arrow we describe the fact that the execution flow (not the spied message) continues with the next module.

4 Case Studies

4.1 The ISO-SC 27 Protocol

Even though the ISO-SC 27 protocol is subject to the parallel sessions attack shown in Table 2, we can defend it by means of a guardian G. Since the victim is A, for the defense to be possible, it is necessary that G is in the configuration in Fig. 2a, i.e., between A and the rest of the network agents, so that he can identify/control all of A's incoming and outgoing messages (by Definition 2, in this configuration the guardian is in topological advantage), whereas in the configuration in Fig. 2b he can be completely excluded by an attacker E. In the following, we give as an example the successful case and a brief explanation for the unsuccessful one.

In order to defend the ISO-SC 27 protocol, we have set up the guardian G with the two spy-filters shown in Fig. 4: an outflow-spy filter in order to record in his dataset D_G all of A's outgoing messages, and an inflow-spy filter in order to record and control A's incoming messages.

Even if G does not know the symmetric key K_{AB}, he can become aware that the protocol has been attacked when he spies via the *inflow-spy* filter a message of the same form of the message (1) in Table 2 (i.e., N_A; the guardian knows that the attacker will reply the first message because he knows the attack function of Definition 3) between those that have previously been identified as such: if an attack is ongoing, then the message that has been identified by the Control Module as critical (i.e., is one of the first messages of the protocol) "has already been seen" by G. We formalize this concept by means of the invariant $Inv(m, i)$:

$$\exists m' \in D_G^{i-1}.\ \Delta_C(m) = 1 \ \wedge \ \Delta_C(m') = 1 \ \wedge \ m = m'.$$

That is, if an attack is ongoing and m is the message spied by guardian's inflow-spy filter, labeled by the Identification Module, and in the Control Module the distinguisher Δ_C believes that it is critical, then the guardian's dataset D_G^i must contain another message m' seen before such that $m = m'$ (the implementation of D_G must be done with respect to the temporal constraints of the invariant Inv, but in this paper we do not discuss the implementation details). Since the

Table 5. Guardian's interference for the ISO-SC 27 protocol.

Interference		
(1.1)	A → E(B)	: N_A
(2.1)	E(B) → G(A)	: N_A
(2.1$_1$)	G(B) → A	: N_{fake}
(2.2)	A → E(B)	: $\{\!\|N_{fake}, N'_A\|\!\}_{K_{AB}}$
(1.2)	E(B) → A	: $\{\!\|N_{fake}, N'_A\|\!\}_{K_{AB}}$
(2.2)	G raises A's flag for abort	

guardian knows that the attacker can use a replay attack, by Definition 4, he has to define the inverse of the attack function as the identity function (the use of the identity function is also reflected in the definition of the invariant).[9]

Let us assume, following [14,15], that each honest agent defended by the guardian G has a set of flags that G can modify in order to make the agent he is defending abort the protocol. Once he has detected such an ongoing attack, G can defend it carrying out the interference. He modifies the content (i.e., he alters the nonce N_A) of the first message in the parallel session (see Table 5 for the complete execution trace, and Table 6 for the corresponding dataset evolution). At this point, the guardian already knows that an attack is ongoing, but we choose to finish the two sessions of the protocol (G changes A's "abort flag" only at the end) in order to show that we can also deliver false information to the attacker and that the Control Module (shown in Table 6) checks the invariant only once since the replayed message in (1.2) is not seen as critical (i.e., it has not the form of the first message). More specifically, Table 5 shows the interference attack that G can use against the attacker E, and Table 6 the evolution of the dataset and the inference during the protocol execution.

To measure the defense mechanism implemented by the guardian for the parallel sessions attack against the ISO-SC 27 protocol, we consider false positives and negatives.

False positives: False positives are possible if, after A completes a protocol run as initiator, B restarts the protocol with A (i.e., they change roles) using (in the first message) a nonce N_B that is already contained in G's dataset. If N_B is represented through a k-bit length string, then the probability of this event is equal to the probability of guessing a nonce amongst those belonging to D^i_G (i.e., G's dataset after i actions):

$$Pr[N_B \in_R \{0,1\}^k, N_B \in D^i_G] = \frac{|D^i_G|}{2^k}$$

So, this probability is negligible if k is large enough (e.g., $k = 1024$).

[9] Formally, for the ISO-SC 27 we have: $f^{-1}(f(N_A, 2), 2) = f^{-1}(N_A, 2) = N_A$ (where $s = 2$ refers to message (2) in Table 4 or, equivalently, message (1.2) in Table 2).

Table 6. Dataset evolution and inference for the ISO-SC 27 protocol. $\{(x.y)\}$ refers to the message sent in step $(x.y)$ (we omit the repeated messages) and to the configuration in Fig. 2a.

i	Protocol message	D_G^i	Identification module	Control module	
			$\Delta_{Id}(m)$	$\Delta_C(m)$	$Inv(m, i)$
0	–	$\{\ \}$	–	–	–
1	(1.1) $A \rightarrow E(B) : N_A$	$\{(1.1)\}$	1	–	–
2	(2.1) $E(B) \rightarrow G(A) : N_A$	$\{(1.1)\}$	1	1	1
3	(2.1$_1$) $G(A) \rightarrow A : N_{fake}$	$\{(1.1), (2.1_1)\}$	–	–	–
4	(2.2) $A \rightarrow E(B) : \{\|N_{fake}, N'_A\|\}_{K_{AB}}$	$\{(1.1), (2.1_1), (2.2)\}$	1	–	–
5	(1.2) $E(B) \rightarrow A : \{\|N_{fake}, N'_A\|\}_{K_{AB}}$	$\{(1.1), (2.1_1), (2.2)\}$	1	0	–
6	(2.2) G raises A's flag for abort	–	–	–	–

False negatives: False negatives are not possible, since not knowing K_{AB} the only way to attack the protocol with the classical attack (Table 2) is to reflect A's messages in a parallel session; but if this situation happens, then the guardian has already seen the message that is coming back to A, and thus he can detect (and afterwards defeat) the ongoing attack. Since G does not admit false negatives for this scenario, G is a total defense mechanism when he is in a topological advantage with respect to his competitor(s), i.e., when he is defending A.

Now that we have seen the successful case, let us focus on the configuration of Fig. 2b. In this configuration, a guardian would not work because B's participation is not mandatory to attack the protocol and thus E can easily exclude G from the run of the protocol; thus there are no false positives and there are only false negatives. In this case, the presence of the resilient channels does not help because G is completely excluded from seeing the execution of the protocol and the attack.

Summing up the analysis of the case study, we have seen how a flawed protocol as the ISO-SC 27 can be defended through the use of a guardian. The first step of our analysis was the attack typically found via model checking and the classical approach. We used the classical attack in order to select the critical messages that the attacker exploits during the attacks. Knowing the critical messages allows us to formalize the invariant, which is also used in order to set up filters and module configurations in the guardian architecture. Finally, we have investigated the different outcomes with respect to the position of the guardian in the network topology.

4.2 Other Protocols

We have applied our approach also to a number of other security protocols. Table 7 summarizes our results, while a more detailed analysis can be found in [18]. For each protocol, in the table we report if the defense is total or partial, which agent is being defended, and the topologies that permit the defense.

Table 7. Other case studies. See [6,8] for details on the protocols.

Protocol	Defense	Agent defended	Topology
ISO-SC 27	Total	A	Fig. 2a
SRA3P	Total	A	Fig. 2a
Andrew Secure RPC	Partial	A	Fig. 2a
Otway-Rees	Total	A	Fig. 2c, e
Encrypted Key Exchange	Total	A	Fig. 2a
SPLICE/AS	Total	A	Fig. 2c
Modified BME	Partial	B	Fig. 2d

In Table 7, we show only the successful results for each protocol in the given task (i.e., defending one of the agents for the corresponding protocol). The outcome of the analysis of these 7 (4 two-agent and 3 three-agent) protocols is quite promising since we have a total defense in 5 cases and a partial defense in the remaining 2 cases.

5 Conclusions and Future Work

Discovering an attack to an already largely deployed security protocol remains nowadays a difficult problem. Typically, the discovery of an attack forces us to make a difficult decision: either we accept to use the protocol even when knowing that every execution can potentially be attacked and thus the security properties for which the protocol has been designed can be compromised at any time, or we do not (generating consequently, kind of a self denial of service). Both choices are extreme, and typically the classical (and conservative) mindset prefers to "dismiss" the protocol and hurry up with the deployment of a new version hoping to be faster than those who are attempting to exploit the discovered flaw.

The above results contribute to showing, we believe, that non-collaborative attacker scenarios, through the introduction of a guardian, provide the basis for the active defense of flawed security protocols rather than discarding them when the attack is found. Regarding the concrete applicability of this approach to security protocols, on one hand, we can use our previous work [14,15] as an approach for discovering how two attackers interact in non-collaborative scenarios and what type of interference the guardian can use, and, on the other hand, in this paper we have given the means to understand how to exploit the interference from a topological point of view, thus bringing the guardian close to real implementation, which is the main objective of our current work.

We are also working on a number of relevant issues, such as how the content of, and the meaning that the honest agents assign to, critical messages may have an influence on the defense mechanisms enforced by the guardian, or such as how to define general attack functions and their inverses. We are also investigating criteria that will allow us to reason about the minimal and/or optimal configurations for protocol defenses. For instance, to show that no further configurations are possible (by showing how m possible configurations can be reduced to $n < m$ base ones, such as the 6 we considered here) or that the considered configuration is optimal for the desired defense (and thus for the implementation of the guardian). It seems obvious, for example, that Fig. 2a is the optimal configuration for defending the initiator A in the majority of two-agent protocols. Similarly, our intuition is that a guardian (with an appropriate defense for a particular protocol) put in configuration of Fig. 2e is also valid for the configuration of Fig. 2c (and similarly for configuration of Fig. 2f with respect to configuration of Fig. 2d).

We envision the some general, protocol-independent results might be possible but that ultimately both the notion (and agents' understanding) of critical message and that of defense configuration will depend on the details of the protocol under consideration and of the attack to be defended against. Our hope is thus to obtain parametric results that can then be instantiated with the fine details of each protocol and attack.

References

1. Armando, A., Arsac, W., Avanesov, T., Barletta, M., Calvi, A., Cappai, A., Carbone, R., Chevalier, Y., Compagna, L., Cuéllar, J., Erzse, G., Frau, S., Minea, M., Mödersheim, S., von Oheimb, D., Pellegrino, G., Ponta, S.E., Rocchetto, M., Rusinowitch, M., Torabi Dashti, M., Turuani, M., Viganò, L.: The AVANTSSAR platform for the automated validation of trust and security of service-oriented architectures. In: Flanagan, C., König, B. (eds.) TACAS 2012. LNCS, vol. 7214, pp. 267–282. Springer, Heidelberg (2012)
2. Basin, D., Caleiro, C., Ramos, J., Viganò, L.: Distributed temporal logic for the analysis of security protocol models. Theor. Comput. Sci. **412**(31), 4007–4043 (2011)
3. Bella, G.: A protocol's life after attacks. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2003. LNCS, vol. 3364, pp. 11–18. Springer, Heidelberg (2005)
4. Bella, G., Bistarelli, S., Massacci, F.: Retaliation against protocol attacks. J. Inf. Assur. Secur. **3**, 313–325 (2008)
5. Blanchet, B.: An efficient cryptographic protocol verifier based on Prolog rules. In: Proceedings of CSF'01, pp. 82–96. IEEE CS Press (2001)
6. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Information Security and Cryptography. Springer, Heidelberg (2003)
7. Ciobâcă, S., Cortier, V.: Protocol composition for arbitrary primitives. In: Proceedings of CSF'10. IEEE CS Press (2010)
8. Clark, J., Jacob, J.: A survey of authentication protocol literature: Version 1.0 (1997)
9. Conti, M., Dragoni, N., Gottardo, S.: MITHYS: mind the hand you shake - protecting mobile devices from SSL usage vulnerabilities. In: Accorsi, R., Ranise, S. (eds.) STM 2013. LNCS, vol. 8203, pp. 65–81. Springer, Heidelberg (2013)
10. Cortier, V., Delaune, S.: Safely composing security protocols. Form. Methods Syst. Des. **34**(1), 1–36 (2009)

11. Cremers, C.J.F.: The Scyther tool: verification, falsification, and analysis of security protocols. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 414–418. Springer, Heidelberg (2008)
12. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Trans. Inf. Theor. **29**(2), 198–208 (1983)
13. Escobar, S., Meadows, C., Meseguer, J.: Maude-NPA: cryptographic protocol analysis modulo equational properties. In: Aldini, A., Barthe, G., Gorrieri, R. (eds.) FOSAD 2007/2008/2009. LNCS, vol. 5705, pp. 1–50. Springer, Heidelberg (2007)
14. Fiazza, M.-C., Peroli, M., Viganò, L.: Attack interference: a path to defending security protocols. In: Obaidat, M.S., Sevillano, J.L., Filipe, J. (eds.) ICETE 2011. CCIS, vol. 314, pp. 296–314. Springer, Heidelberg (2012)
15. Fiazza, M.-C., Peroli, M., Vigano, L.: An environmental paradigm for defending security protocols. In: 2012 International Conference on Collaboration Technologies and Systems (CTS), May 2012, pp. 427–438 (2012)
16. ISO: ISO-IEC JTC1.27.02.2(20.03.1.2) entity authentication using symmetric techniques. International Organization for Standardization (ISO) (1990)
17. Mödersheim, S., Viganò, L.: Secure pseudonymous channels. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 337–354. Springer, Heidelberg (2009)
18. Peroli, M., Viganò, L., Zavatteri, M.: Non-collaborative Attackers and How and Where to Defend Flawed Security Protocols (Extended Version) (2014). http://arxiv.org/abs/1405.6912
19. Ryan, P., Schneider, S., Goldsmith, M., Lowe, G., Roscoe, B.: Modelling and Analysis of Security Protocols. Addison Wesley, Reading (2000)
20. Viganò, L.: Automated security protocol analysis with the AVISPA tool. Electron. Notes Theor. Comput. Sci. **155**, 61–86 (2006)