frontiers in ICT

ORIGINAL RESEARCH
published: 03 August 2015
doi: 10.3389/fict.2015.00011

Defending vulnerable security protocols by means of attack interference in non-collaborative scenarios

Maria-Camilla Fiazza[1], *Michele Peroli*[1] *and Luca Viganò*[2]*

[1] *Dipartimento di Informatica, Università di Verona, Verona, Italy,* [2] *Department of Informatics, King's College London, London, UK*

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 are 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. We formalize a framework for multi-attacker scenarios and show, through a detailed case study, that concurrent competitive attacks can interfere with each other. It is then possible to exploit interference to provide a form of defense to vulnerable protocols.

Keywords: non-collaborative attackers, attack interference, attacker personality, attack mitigation, security protocols

OPEN ACCESS

Edited by:
Nicola Zannone,
Eindhoven University of Technology, Netherlands

Reviewed by:
Santiago Escobar,
Universitat Politècnica de València, Spain
Yannick Chevalier,
Université de Toulouse, France

Correspondence:
Luca Viganò,
Department of Informatics, King's College London, Strand, London WC2R 2LS, UK
luca.vigano@kcl.ac.uk

Specialty section:
This article was submitted to Computer and Network Security, a section of the journal Frontiers in ICT

Received: *10 December 2014*
Accepted: *29 June 2015*
Published: *03 August 2015*

Citation:
Fiazza M-C, Peroli M and Viganò L (2015) Defending vulnerable security protocols by means of attack interference in non-collaborative scenarios. Front. ICT 2:11. doi: 10.3389/fict.2015.00011

1. Introduction

1.1. Context and Motivation

The typical attacker adopted in security protocol analysis is the *Dolev–Yao (DY) attacker* (Dolev and Yao, 1983), who can synthesize, send, and intercept messages at will but cannot break cryptography (i.e., the DY model follows the *perfect cryptography* assumption). He is thus in complete control of the network – he is often formalized as being the network itself – and 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 vulnerabilities introduced in the implementation phase, the protocol can thus be safely employed in real-life networks, at least in principle.

Symbolic approaches for security protocol analysis typically consider a single DY attacker since models with multiple collaborating DY attackers are reducible to models with one DY attacker [see, e.g., Caleiro et al. (2005) for a proof, Syverson et al. (2000) for an equivalent reduction for Machiavelli-type attackers, and Syverson et al. (2000), Comon-Lundh and Cortier (2003), and Basin et al. (2011) for general results on the reduction of the number of agents to be considered]. However, different models have been recently proposed that consider *multiple attackers*. For instance, Basin et al. (2009) and 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 multiple distributed attackers, with restricted, but more realistic, communication capabilities than those of the standard DY attacker.

Multiple attackers are also considered in Bella et al. (2003, 2008) and Arsac et al. (2009, 2011), where each protocol participant may behave as a DY attacker, without colluding nor sharing knowledge with anyone else. The analysis of security protocols under this multi-attacker model considers scenarios of agents competing with each other for personal profit. Agents may also carry out *retaliation attacks*, where an attack is followed by a counter-attack, 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) and Schaller et al. (2009) and of Bella et al. (2003, 2008) and Arsac et al. (2009, 2011) 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. 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. 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 this interference to mitigate protocol vulnerabilities, thus providing a form of protection to flawed protocols. To investigate non-cooperation between attackers, we have devised a (protocol-independent) model in which (i) a protocol is run in the presence of multiple attackers, and (ii) attackers potentially have different capabilities and different knowledge, and can interfere with each other. This, ultimately, allows us to consider the creation of a benign attacker for protocol defense: agents can rely on a *network guardian*, an *ad hoc* agent whose task is hindering dishonest agents (i.e., attackers) from attacking vulnerable protocols. In other words, we look for *defenses* against discovered attacks.

This paper continues our research endeavor on defending vulnerable protocols: we defined a suitable model for non-collaboration (Fiazza et al., 2011a,b), discussed the conceptual implications of treating protocols as environments (Fiazza et al., 2011c), and presented a methodology to systematize the construction of guardians (Fiazza et al., 2012) and their placement in the network topology (Peroli et al., 2014).

In this paper, we refine and slightly extend our models and techniques to consider the fine details of a refined case study. With this case study, we illustrate that non-collaborative mechanisms (and attacker reasoning) can vary tremendously across protocol types. We proceed as follows. In Section 2, we formalize models for the network and the agents, including agent attitude, goals, and disposition. More specifically, we formalize (i) sets for honest agents and attackers that are used in the analysis with the purpose of filtering those messages that the attackers pay attention to; (ii) datasets that model the knowledge of all agents and the contents of the network; (iii) the messages that transit on the network (i.e., are contained in the network dataset) in the form of triplets ⟨sender-ID, message, receiver-ID⟩; and (iv) a network handler that regulates the execution of the actions of honest agents and attackers. Then, in Section 3, we consider the SRA3P protocol, focusing on attack procedures interacting in manners that cannot be observed in classical settings. In Section 4, we discuss the lessons learned, focusing on success criteria for competitive attackers and how these criteria can be prioritized in order to define attack strategies. In Section 5, we explain how interference between attacks leads to a methodology that can be used for defending vulnerable protocols against attacks. In Section 6, we summarize and discuss future work.

2. Models: Network, Agents, Attitude

2.1. Goals of Modeling and Approach

Our approach is general and abstracts away from the specific security properties for which protocols were designed. Still, for concreteness, when considering a particular protocol, the specific properties must be taken into account. For brevity, in this paper, we restrict our attention to *confidentiality* and to *two* non-collaborative attackers (E_1 and E_2), in addition to honest agents A and B. Let $Eves = \{E_1, E_2\}$ be the *set of attackers*, $Agents = \{A, B, E_1, E_2\}$ the *set of all network agents* (honest and dishonest), X, Y, Z, and W variables varying over *Agents* and E a variable over *Eves*.

To focus on the interference between two attackers directing their attacks toward 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 misled by others on purpose. If the knowledge[1] available to an attacker affects his view, 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. Besides reflecting this stance, a network model for non-collaborative scenarios should also support a form of competition for access to messages, especially if attacks rely on erasing messages.

We diversify the activity of our attackers by admitting that they 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.

We use the predicate $ofInterest_E\ (X)$ (defined by *DecisionalProcess* in **Table 1**) to specify that attacker *E decides* to pay attention to the network traffic generated by an agent X, which gives rise to the set $Attend_E = \{X \in Agents \mid ofInterest_E\ (X)\}$ of the agents to which E is attentive. How and why this decision is taken will be based on decisional processes that can be modeled in different ways according to the attacker strategies that we discuss in Section 3. The characteristic feature of the attackers we consider is their *attitude*. Dishonest agents wish to attack a protocol and are ready, should

[1]We do not attach any epistemic interpretation to the knowledge we consider, but rather 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 (spying, injecting, erasing; messages transit on the network dataset in the form of triplets ⟨*sender-ID, message, receiver-ID*⟩) and system configuration (*True-Sender-ID*, *DecisionalProcess*, *NetHandler*).

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

$$\frac{(m_1, m2) \in D_E^i}{m_j \in D_E^i \text{ for } j \in \{1, 2\}}\ \textit{(Proj)} \qquad \frac{\{m\}_k \in D_E^i \quad k^{-1} \in D_E^i}{m \in D_E^i}\ \textit{(Decr)}$$

$$\frac{<X, m, Y> \in D_{net}^i \quad sender(<X, m, Y>) \in D_E^i \quad Y \in D_E^i \quad \psi}{m \in D_E^{i+1}}\ \textit{(Restricted-Spy)}$$

$$\frac{<X, m, Y> \in D_{net}^i \quad ofInterest_E(X) \quad Y \in D_E^i \quad \psi}{m \in D_E^{i+1} \wedge sender(<X, m, Y>) \in D_E^{i+1}}\ (Inflow - Spy)$$

$$\frac{<X, m, Y> \in D_{net}^i \quad sender(<X, m, Y>) \in D_E^i \quad ofInterest_E(Y) \quad \psi}{m \in D_E^{i+1} \wedge Y \in D_E^{i+1}}\ \textit{(Outflow-Spy)}$$

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

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

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

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

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

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

*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.*

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 who do not cooperate in the classical sense but have the same attack power and differ with respect to the information content of their knowledge. Such differentiation arises out of attentional choices and not out of intrinsic constraints. Strategic and attitude considerations (see Sections 4.2 and 4.3 for further discussion) should not be derivable explicitly from the attacker model; rather, they should configure it. To support this, 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. For concreteness, in Section 2.2, we first discuss how we model honest and dishonest agents statically, i.e., how we model their knowledge and how they synthesize, analyze, and spy messages. Then, in Section 2.3, we discuss the formalization of the dynamics of the network model.

2.2. Agent Model

In this section, we discuss the agent model, focusing in particular on the datasets for honest and dishonest agents and the network. More specifically, the knowledge of each honest or dishonest agent X is characterized by a proprietary dataset D_X, which is monotonically non-decreasing as agents never forget. We write D_X^i when it is important to highlight that the dataset is to be considered at a particular state *i*. The network *net* is also formalized through a dataset D_{net}, also possibly indexed. In Section 2.3, we discuss how datasets evolve and how indexing and evolution are related to actions and message transmission.

We adapt the notion of DY attacker to capture a non-collaborative scenario. **Table 1** shows how one such attacker E may be formalized within our model, with rules with respect to D_E and D_{net}. 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 carry out logical inference to identify defenses against attacks; rather, we recognize in the system's evolution what in our eyes corresponds to a defense.

Like honest agents, attackers can *send* and *receive* messages, where they can derive new messages by:

- synthesizing messages via the rules *Comp* (for concatenation of messages m_1 and m_2) and *Encr* (for encryption of a message m with a key k),
- analysis messages via the rules *Proj* (for projection of a message pair into its components) and *Decr* (for decryption of a message m with a key k^{-1}).

Attackers can also eavesdrop messages transiting on the network (via the *-*Spy* rules, which deserve a more detailed explanation that we give below) and remove them, so that they do not reach their intended receiver (via the rule *Erase*). Attackers can also partially impersonate other agents by injecting messages under a false identity (via the rule *Injection*); we write $E(X)$ to specify that E impersonates X. Further rules can be easily added to those in **Table 1** for other forms of encryption, digital signatures, hashing, creation of nonces, and other fresh data.

Messages transit on the network dataset in the form of triplets ⟨sender-ID, message, receiver-ID⟩. 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 impersonation). The most significant feature concerns spying, represented through three different *-*Spy* rules formalizing 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 is *Restricted-Spy*: only the messages involving known agents in both sender and receiver roles, regardless of their honesty, become part of the attacker's dataset. In our model, what matters is the actual sender and not the declared sender (as formalized by the rule *True-Sender-ID* in **Table 1**). This prevents total impersonation and allows filtering messages on the basis of the agent's attentional choices.

The attentional filter 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 is achieved by the rules *Inflow-Spy* and *Outflow-Spy*. Attackers have the option of accepting or rejecting the newly discovered identifier X, on the basis of the predicate $ofInterest_E\ (X)$, which models the decisional process for attention.

An attacker cannot apply any of the *-*Spy* rules to obtain the message m without knowing the identifier of m's sender or m's intended receiver. By not providing a "generalized spy" rule to waive this requirement, we ensure that $D_E^0\ \cap Agents = \varnothing$ implies $D_E^i \cap Agents = \varnothing$ for all i. Although E can augment D_E indefinitely (through internal message generation and the synthesis rules *Comp* and *Encr*), E's network activity is in fact null.

An attacker's dataset D_E consists of (i) messages derived from the messages received or spied and (ii) identifiers of the agents to whom the attacker is attentive. $Attend_E$ 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. Differently from D_{net} (defined later), agent datasets do not contain triplets ⟨*sender-ID, message, receiver-ID*⟩, but only messages or identifiers.

Once a new identifier X enters D_E, E establishes a belief about the honesty of X and places the identifier in one of H_E, A_E, or U_E. We do not specify how the agents initially build their knowledge base and establish or update their belief about the attitude of other known agents. Agents are on the watch for suspicious messages, which may indicate that an attack is ongoing or 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 consider are *smart*: they always employ the available strategic information, and different strategies might be defined and grafted onto our approach.

Attackers do not have automatic access to triplets relating sender, message, and receiver. They must infer key pieces 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 of known agents small. The difficulty of inference rises with the number of attackers in the set $Attend_E$.

2.3. Network Model

We now consider the evolution of the datasets. All the operations that can change the state of the network dataset D_{net} (sending, receiving, injecting, and erasing message) are termed *actions*, whereas we consider spying simply as an operation: although it requires interacting with the network, it does not change its state. Messages in transit are inserted in D_{net}, where attackers can spy them before they are delivered to their intended receivers. Contextual to delivery, the message is removed from the dataset. 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 D_{net}; the index of D_{net}^i is shared with all the proprietary datasets D_X^i, whose states are synchronized accordingly. D_{net}^i is the state of the network dataset *after the i-th action*.

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

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 concurrency.

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. We introduce a *network handler* [formalized by (*NetHandler*) in **Table 1**], whose task is to regulate the selection of the next action and implement the dependencies between the selected action and the knowledge available to each attacker; the network handler also keeps the system evolution in accordance with additional constraints, modeling, for example, information-sharing within specific subsets of agents and network topology.

When the state of the network changes (e.g., as a result of injection or sending), 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 spy rules, the inject rule and 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 takes a message triplet and the state and returns the identifiers of the attackers who can spy "before" the message is erased from D_{net}. *canSee* returns at least the identifier of the attacker whose erase request was served.

If the network handler does not receive any erase 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*, the prior (simulated) spy is subjected 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 a message m addressed to Y, at least one of which has requested m to be erased. 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.

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 how to derive systematically an attack behavior of this type.

In the following case study, we start from a simple protocol for which a vulnerability is known and describe a possible reasoning for a competitive attacker in the context of the protocol's main features; as a result, we devise for the known ("classical") attack a variant that explicitly considers the possibility of ongoing independent attacks.

The visibility of the last message to the attackers makes a huge difference on the outcome. By erasing messages, through the mechanism normed by *canSee*, attackers can deny information to their competitors and manipulate their ability to succeed. In Section 3.1, we discuss the scenario under the assumption that all attackers see the last message sent by the honest initiator, so as to focus our attention on behavioral choices during the protocol run. We then explore (in Section 3.2) the general case, in which the last message is not trivially visible to all attackers; here we focus on the attacker's reasoning, emphasizing what can be learned during the protocol run.

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 attackers, E_1 and E_2:

- 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 $E_2 \in H_{E_1}$).
- Case 2: E_1 and E_2 know each other as attackers. $E_1 \in D_{E_2}$ and $E_2 \in D_{E_1}$ and they have correctly understood that the other is behaving as a dishonest agent ($E_1 \in A_{E_2}$ and $E_2 \in A_{E_1}$).
- 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 $E_2 \notin D_{E_1}$).
- Case 4: E_2 knows E_1 as honest. Only one of E_1 and E_2 is paying attention to the other, say $E_1 \in H_{E_2}$ and $E_2 \notin D_{E_1}$.
- Case 5: E_2 knows E_1 as dishonest. Only one of E_1 and E_2 is paying attention to the other and knows his identifier, say $E_1 \in A_{E_2}$ and $E_2 \notin D_{E_1}$
- Case 6: E_2 knows E_1, but he is unsure of E_1's honesty. Only one of E_1 and E_2 is paying attention to the other and knows his identifier, say $E_1 \in U_{E_2}$ and $E_2 \notin D_{E_1}$.

We now consider a detailed case study, pointing to Fiazza et al. (2011a,b) and Peroli et al. (2014) for other case studies.

3.1. The Shamir–Rivest–Adleman Three-Pass Protocol

The *Shamir–Rivest–Adleman Three-Pass protocol* [*SRA3P* (Clark and Jacob, 1997)] consists in three message exchanges, as shown in **Table 3**(A), and it assumes commutative cryptography, i.e., that $\{|\{|M|\}_{K_A}|\}_{K_B} = \{|\{|M|\}_{K_B}|\}_{K_A}$ holds. The goal is confidentiality; if the message transmitted is interpreted as a session key, then the protocol can be considered as a *key transport* protocol.

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 $I \subseteq canSee\ (< X, m, Y >, i+1)$

TABLE 3 | Attacks against SRA3P.

(A) SRA3P			**(B) Classical Attack**		
(1)	$A \rightarrow B$	$: \{\|M\|\}_{K_A}$	(1)	$A \rightarrow E(B)$	$: \{\|M\|\}_{K_A}$
(2)	$B \rightarrow A$	$: \{\|\{\|M\|\}_{K_A}\|\}_{K_B}$	(2)	$E(B) \rightarrow A$	$: \{\|M\|\}_{K_A}$
(3)	$A \rightarrow B$	$: \{\|M\|\}_{K_B}$	(3)	$A \rightarrow E(B)$	$: M^* = M$
(C) Strong attack			**(D) Subtle attack**		
(1)	$A \rightarrow E_{1,2}(B)$	$: \{\|M\|\}_{K_A}$	(1)	$A \rightarrow E_{1,2}(B)$	$: \{\|M\|\}_{K_A}$
(2)	$E_1(B) \rightarrow A$	$: \{\|M\|\}_{K_A}$	(2)	$E_1(B) \rightarrow A$	$: \{\|M\|\}_{K_A}$
(3′)	$E_2(A) \rightarrow E_1$	$: M_{fake}$	(3′)	$E_2(A) \rightarrow E_1(B)$	$: M_{fake}$
(3)	$A \rightarrow E_{1,2}(B)$	$: M^*$	(3)	$A \rightarrow E_{1,2}(B)$	$: M^*$

K_A and K_B are private keys, and cryptography is commutative.
(A) Protocol followed by honest agents. (B) Classical attack on SRA3P, employed by attackers when unaware of active competitors. (C) Strong competitive attack, employed by attackers when the competitor's identifier is known (E_2 knows that his competitor is E_1). (D) Subtle competitive attack, employed by attackers when aware of the existence of an active competitor but unsure of the competitor's identity (E_2 knows that he has a competitor but does not know that it is E_1).

The classical attack to SRA3P exploits A as an oracle for the content of the message [**Table 3**(B)]. The attacker E replaces the intended recipient B in receiving the message and pretends to perform step (2), actually sending back the message $\{|M|\}_{K_A}$ without further encryption. A continues according to the protocol and removes his key from the message, thus sending back the *secret* M without any encryption. We write M^* to emphasize that M transits in the clear without A meaning it.

The classical attack is successful; however, it prevents the intended recipient B from receiving any messages at all. In case the honest agents had prior agreement that an exchange was to take place, B can detect that something has gone wrong. The classical attack is very strong against detection even in this case: after discovering M, the attacker E impersonates A and performs the protocol with B, *de facto* carrying out a complete man-in-the-middle attack (completely undetected).

Provided that some attacker answered A in step (2) by sending $\{|M|\}_{K_A}$, it is sufficient to spy the message in step (3) to acquire the secret. In our set-up, any attacker attempting to erase a message is always successful in preventing honest agents from receiving it, but he is not necessarily successful in hiding it from other attackers [only attackers in canSee ($\langle A, M^*, B\rangle$, i) have access to M]. In this situation, a second attacker E_2 can prevent his competitors from acquiring the secret only by weakening their ability to identify the message M^* as the true response of A in step (3). A competitive attacker will therefore attempt to mislead his competitors by sending on the network fake messages that are not related to the information coming from A.

If the recipient of a fake message is expecting to receive M^*, he may be led into thinking that he has successfully carried out his attack. He may then stop spying the current run of the conversation between A and B and conclude that he has succeeded when in fact he has acquired the wrong "secret" M_{fake}. If, instead, the competitor E_1 is not following the classical attack and chooses to keep listening in on the conversation, he receives more than one message M^* and does not know which one has been sent by the honest agent A.

The competitor faces a degree of uncertainty in identifying M^* that is not present in the classical attack. The degree of uncertainty (i.e., between how many fake messages he has to choose the real one) to which E_1 is subject can be increased arbitrarily by E_2, who can send multiple and unrelated fake messages, both before and after M^* transits on the network.

The success of such a non-collaborative behavior in securing sole ownership of the secret depends critically on the listening behavior of the competitor: if E_1 stops spying network traffic as soon as a response is received, then it is critical for E_2 to send a fake message *before* A's reply; in case of success, the competitor fails to acquire the secret. If the competitor is actively listening past the reception of the first response, then M^* is eventually acquired – but not by itself: a situation of uncertainty arises.

In classical settings, uncertainty does little more than affect the probability that an attack will be successful; however, if honest agents are immersed in a retaliatory framework, when an attacker guesses the wrong M^* and uses it as a session key to communicate with A, he may incur in significant consequences. Therefore, attackers in non-collaborative scenarios should be careful to evaluate the probability of correctly guessing M^* against the added costs of failure – either in terms of retaliation or of the strategic risks of being detected or identified by honest agents.

As a result of this discussion, for competitive scenarios involving SRA3P, we propose two competitive variants of the classical attack, employed by attackers who are aware of the presence of active competitors. We term the variants *strong* and *subtle* attack, differing with respect to attacker knowledge. If the attacker is aware of the identity of the competitor, he will employ the strong attack, whereas he will resort to the subtle attack when only the competitor's presence is known. These new attack behaviors are also oracle-type [transmission step, see Carlsen (1994) for a taxonomy of flaws and attacks] and are shown in **Table 3**(C,D).

The main difference between the two non-collaborative attack behaviors lies in the method of delivery of fake messages to E_1. If the competitor's identity is known, E_2 can ensure that the fake message is seen even if E_1 is not paying attention to E_2's traffic: E_2 sends the fake message directly, using the network primitive *send*. If, on the other hand, E_1's identity is unknown, E_2 is forced to rely on a reasonable prediction of E_1's behavior and thus injects the fake message, impersonating A. The misleading message M_{fake} is successfully delivered if E_2 is present in E_1's dataset and E_1 spies it. If E_1 does not gain M_{fake}, E_2 fails to pollute the competitor's knowledge but does not compromise his own ability to observe M^*.

SRA3P is such that all attentive attackers can potentially acquire the secret if an attack on the initiator A is carried out. Exclusive knowledge of the secret can only occur through two mechanisms: through the outcome of erase requests (which is not under the control of network agents) or by misleading other attackers into interpreting a fake message as M^*.

An attack is successful if it goes undetected by the initiator A, who then transmits M in the clear as M^*. Our agents are intelligent and they make use of all information available to perform in-protocol detection of attacks. With respect to SRA3P, a clear indication for A consists in receiving a duplicate response from agents posing as B; under this circumstance, A concludes that there has been a security violation and halts the execution of the protocol to protect the secret M.

From the attackers' perspective, an ongoing attack can be detected by observing that the message transiting on the network in step (2) is equal to the message $\{|M|\}_{K_A}$ transiting on step (1). The attack trace is unambiguous to spying attackers. SRA3P is very unfriendly for attacker labeling: identifiers do not transit on the network, neither in the clear nor encrypted (the attackers do not see the *IDs*). Decisional processes cannot rely on any conclusive information concerning the identity of the agents involved in a given protocol run and must resort to inference on the basis of their current knowledge.

3.1.1. Attacker Configuration and Outcomes of Interaction

We examine the outcome of attacks for the six cases above. Refer to **Table 4** for a synthetic view of the message exchanges in each configuration.

TABLE 4 | Traces for non-collaborative attacks against SRA3P.

T1: cases 1, 4, 5, 6			**T2: case 2**		
(1)	$A \rightarrow E_{1,2}(B)$	$: \{\vert M\vert\}_{K_A}$	(1)	$A \rightarrow E_{1,2}(B)$	$: \{\vert M\vert\}_{K_A}$
(2_1)	$E_1(B) \rightarrow A$	$: \{\vert M\vert\}_{K_A}$	(2)	$E_2(B) \rightarrow A$	$: \{\vert M\vert\}_{K_A}$
(3_2)	$E_2(A) \rightarrow E_1$	$: M_{fake}$	(3_1)	$E_1(A) \rightarrow E_2$	$: M'$
(3)	$A \rightarrow E_{[1],2}(B)$	$: M^{*}$	(3_2)	$E_2(A) \rightarrow E_1$	$: M''$
			(3)	$A \rightarrow E_{1,2}(B)$	$: M^{*}$

T3: case 3			**T4: cases 4, 5 and 6**		
(1)	$A \rightarrow E_{1,2}(B)$	$: \{\vert M\vert\}_{k_A}$	(1)	$A \rightarrow E_{1,2}(B)$	$: \{\vert M\vert\}_{k_A}$
(2_1)	$E_1(B) \rightarrow A$	$: \{\vert M\vert\}_{k_A}$	(2_2)	$E_2(B) \rightarrow A$	$: \{\vert M\vert\}_{k_A}$
(2_2)	$E_2(B) \rightarrow A$	$: \{\vert M\vert\}_{k_A}$	(2_1)	$E_1(B) \rightarrow E_2(A)$	$: \{\vert M\vert\}_{k_A}$
			(3_2)	$E_2(A) \rightarrow E_1$	$: M_{fake}$
			(3)	$A \rightarrow E_{1,[2]}(B)$	$: M^{*}$

Traces are exhaustive aside for order of attackers.
Case 1: E_1 and E_2 know each other as honest. Case 2: E_1 and E_2 know each other as dishonest. Case 3: E_1 and E_2 are unaware of each other. Case 4: E_2 knows E_1 as honest. Case 5: E_2 knows E_1 as dishonest. Case 6: E_2 knows E_1 but has not yet established a belief on E_1's honesty.

In order to completely specify agent behavior, we state the following:

1. An attacker who, before starting his own attack, spies $\{|M|\}_{K_A}$ (part of an attack trace) transiting on the network moves on to step (3) of his chosen competitive attack (strong or subtle). If the attacker spies the attack trace after sending $\{|M|\}_{K_A}$ himself, then he requests that the message be erased. In our set-up, erase requests always prevent messages from reaching honest recipient, although attackers cannot deterministically be prevented from spying it. This behavioral rule accounts for attackers being aware that duplicate messages can be exploited to perform attack detection.
2. An attacker who is employing a competitive attack (as he is aware of the presence of active competitors) continues spying on the network even after receiving the first message.
3. An attacker may learn that he has incorrectly classified an agent as honest. We do not focus here on decisional processes for agent classification and therefore we posit that these processes serve as oracles for the identifiers of honest agents involved in the protocol run and also for the identity of the mislabeled agent. Whenever evidence that an agent has been mislabeled arises, the decisional processes of the agents allow relabeling in A the dishonest agent who has triggered the anomalous situation detected. For completeness, we explicitly mention in case 1.T1-B which choices would be available to the agent, should the decisional process yield incorrect answers. The reliability of deductive processes related to identifiers determines to what degree an attacker can be confident in choosing the strong attack over the subtle attack in **Table 3**.
4. We posit that *canSee* for A's opening message comprises both E_1 and E_2; otherwise, only one attacker would be active in the run of the protocol examined.
5. We initially posit that *canSee* yields the entire attacker set for the message sent by the honest agent A in step (3); otherwise only some (one) of the intruders could acquire M^{*}. Section 3.2 discusses how outcomes are affected by *canSee*.

Case 1: E_1 and E_2 know each other as honest: Initially, E_1 and E_2 are unaware of active competitors and mount the classical attack. The first between E_1 and E_2 to send the message at step (2) reveals to the other that he has incorrectly classified an agent. Let E_1 attack first. E_2 employs his decisional process to identify the mislabeled attacker.

- (1.T1-A) E_1 *is identified as an attacker by* E_2. E_2 switches to the strong attack, with the goal of gaining exclusive access to M. In step (3_2), E_2 sends a fake message to the unsuspecting competitor E_1, who is expecting a message from A containing M^{*} on the clear. E_1 may now think that he has successfully completed the attack, but in fact he did not acquire the secret M. After receiving M_{fake}, E_1 stops monitoring the network, according to the classical attack behavior.

  If E_1 continues to spy, he will also acquire M^{*}. However, E_1 finds himself in a situation of uncertainty, as he is not able to determine if it is M_{fake} or M^{*} (or neither) that comes from A. E_1 can at most determine that there is an unlabeled active competitor, one that he has not previously identified in A_{E_1}.
- (1.T1-B) E_2 *fails to identify* E_1 *as a dishonest agent.* E_2 has two strategies available: risk revealing himself as an attacker and employ the strong attack against all agents he is attentive to (except the protocol initiator), or employ the subtle competitive attack with partial impersonation.

Case 2: E_1 and E_2 know each other as attackers: Both E_1 and E_2 know the competitor's identity and thus both follow the strong attack. The attack trace prescribes waiting for a competitor to start the attack procedure, by sending $\{|M|\}_{K_A}$ to A. Both attackers are waiting for the other to take action. The situation could result in a deadlock, but the attackers know that a message has been erased and that A is waiting for an answer.

The attackers wait for a reasonable amount of time (depending on the real set-up where the protocol is utilized) and then one takes the initiative, say E_2 first answers A. The strong attack consists in polluting the knowledge of the competitor with a fake message. Both attackers send their fake messages (M' and M''), thereby recreating the uncertainty of the previous case. This time the uncertainty spreads over both attackers and none dominates the other.

Case 3: E_1 and E_2 are unaware of each other: hence, both employ the classical attack. The attackers, not paying attention to the other's communications, do not realize that an attack trace is transiting on the network. A receives a duplicate message that he correctly interprets in terms of an ongoing attack. The attackers are detected, even if not explicitly identified. A abandons the protocol to keep M secret.

Case 4: E_2 knows E_1 as honest: E_2 is not aware of other attackers and can choose to attack right away or wait for a reasonable time to try detecting a mislabeled attacker.

- (4.T1) E_2 *waits and* E_1 *starts the classical attack.* E_2 has the chance of detecting E_1 as an attacker and starts the strong attack. The situation is reduced to case 1. If E_1 continues to listen on the network after the end of his (unsuccessful) attack, he realizes that he is in a situation of uncertainty, not knowing which between M^{*} and M_{fake} is A's secret. E_1 is now certain that an attacker is present but he does not know who, because the identifier E_2 is not in E_1's proprietary dataset. E_1 can thus switch to an exploratory strategy, using the inflow-spy rule for the subsequent runs of the protocol.

- (4.T4) E_2 *starts the classical attack.* Not having E_2's identifier in his dataset, E_1 does not pay attention to the message and does not notice the attack. E_1 continues his attack and sends $\{|M|\}_{k_A}$. In step (2_1), E_2 detects the dishonesty of E_1 and switches to the strong attack. There is an important difference with respect to case 4.T1: E_2 erases the message sent by E_1 to A, thereby preventing A from detecting a duplicate message and protecting his own attack.

Case 5: E_2 knows E_1 as dishonest: E_2 knows his competitor's identifier. When A initiates the protocol, E_2 waits for E_1 to start the attack and prepares to send a fake message in step (3_2), employing the strong attack (case 5.T1). If E_1 does not send $\{|M|\}_{k_A}$ within a reasonable time (case 5.T4), E_2 performs the attack in step (2_2). This message goes undetected by E_1, who will send his message later. E_2 is aware that another attacker is present and is on the watch for a replicate attack message, which he erases. If E_1 acts first, the sequence of messages is the same as in case 1; otherwise, the sequence is the same as in case 4.T4.

If E_1 continues to spy after receiving M_{fake}, he can realize that he is uncertain with respect to M and can therefore deduce the presence of an unknown attacker. E_1 employs exploratory versions of the *-*Spy* rules to try gaining information about the competitor's identity.

Case 6: E_2 knows E_1 but he is unsure of E_1's honesty: this reduces to cases 5.T1 and 5.T4, according to who first initiates the attack by sending $\{|M|\}_{k_A}$, be it E_1 (case 6.T1) or E_2 (case 6.T4). In all cases, E_2 has a clear advantage because he is paying attention to E_1's messages but his own messages are not being attended to. In addition to what happens in case 5, E_2 has the opportunity to correctly label E_1: E_2 moves E_1's identifier from U_{E_2} into A_{E_2}.

3.2. The General Case for SRA3P

We now move on to a more thorough view, examining the full range of outcomes emerging from two simultaneous independent attacks against SRA3P. Attack traces are the same as those reported in **Table 4**.

With reference to **Tables 5** and **6**, for each of the six cases, we identify the following subcases:

- E_1 is using the classical attack and stops spying on the network after receiving the first message that can play the role of M ("E_1 stops"). In case 2, this situation does not occur as attackers who are aware of competitors keep spying after the first message received.
- E_1 continues to spy even after receiving the first message that can be interpreted as M; we treat separately the different values of *canSee* for A's response in step (3). Note that if an attacker is not in *canSee*, he fails regardless of the number of fake messages dispatched. In case 3, the honest initiator detects an ongoing attack and withdraws from the protocol; as a result, step (3) is not carried out and we do not distinguish subcases in **Table 5**.

For each attacker role, we describe:

- "Attack": which attack has been used ("classical" or "strong") or if there has been a switch from the classical to the strong attack during the protocol run ("Cl $\rightarrow$ Str").
- "Detection": the ability to acquire further information on competitors. Possible values are: none performed ("none"); none possible, because the agent already has a correct understanding of the situation ["none (c)"]; in-protocol detection, by spying the attack trace when no competitor is known ["(in) trace"]; post-protocol detection, by realizing that more than one candidate M has been spied and an unknown competitor is responsible for the uncertainty ["(post) uncertainty"] – with the variant ("post $\exists$") uncertainty to also signal that the identifier of the previously unknown competitor is not in *Attend*.
- "Messages": the set of messages that can be interpreted as M. M! indicates that only M has been spied; M^+ indicates that more than one message, including M, has been spied; M_{fake} that only fake messages have been spied; "none," that no message has been spied during the protocol run.
- "Result": the result of the protocol run. Possible results are: "full failure" (the attacker does not acquire M and takes a fake message for the secret), "failure" (the attacker does not acquire M and realizes it), "uncertainty" (the attacker acquires the secret M along with other fake messages), "success" (the attacker knows M without uncertainty), "dominance" (the attacker succeeds and all his competitors fully fail).

For honest agents, we show only the result: either security failure or attack detection through duplicate messages. For ease of reference, the last two rows of each subtable pertain to the outcome of interaction when a guardian G is introduced in the network, along with a single (competitive) attacker E. Refer to Section 5 for an introduction to guardians.

G_{E_1} plays the role of E_1 against E playing E_2, and G_{E_2} plays the role of E_2 against E_1. Similarly to attackers, we show for G the possible conclusions that can be drawn on the actual security of the protocol run and what can be deduced about attacker identity. Security can be: *compromised*, if E knows M with certainty; *uncertain E*, if E knows M but cannot identify it with certainty; *restored*, if E fails to acquire M; *enforced*, if thanks to G being present, flags were raised for A that allow A to detect an ongoing attack and abort the protocol to protect M.

4. Lessons Learned

4.1. Some General Comments

First of all, let us observe that classical attacks rely on rather simple mechanisms, such as replacing part of a message (to force a wrong binding between a public key and the agent it is believed to belong to) or replaying part of a message (e.g., an old session key, which is wrongly accepted as new). Aside for a general interest in flaw taxonomy, the classical view on protocol security typically pays little attention to the *basic* mechanism upon which all security rests; it is taken for granted that it is there and that, since it is so basic, it is trivially possible to attack it. On the other hand, moving from single to multiple attackers brings to the fore that

TABLE 5 | Overall SRA3P results, detailed view of cases 1, 2 and 3.

Case 1		E_1 stops	E_1 continues and canSee(M*)=		
Agent	Feature		{E_1, E_2}	{E_1}	{E_2}
E_1	Attack	Classical	Classical	Classical	Classical
	Detection	None	(Post) uncertainty	(Post) uncertainty	None
	Messages	M_{fake}	M^+	M^+	M_{fake}
	Result	Full failure	Uncertainty	Uncertainty	Full failure
E_2	Attack	Cl → Str	Cl → Str	Cl → Str	Cl → Str
	Detection	(In) trace	(In) trace	(In) trace	(In) trace
	Messages	M!	M!	None	M!
	Result	Dominance	Success	Failure	Dominance
A	Result	Failure	Failure	Failure	Failure
G_{E_1}	Detection	None	(Post) label	(Post) label	None
	Security	Compromised	Compromised	Restored	Compromised
G_{E_2}	Detection	(In) label	(In) label	(In) label	(In) label
	Security	Restored	Uncertain E	Uncertain E	Restored
Case 2		**E_1 stops**	**canSee(M*)=**		
Agent	**Feature**		**{E_1, E_2}**	**{E_1}**	**{E_2}**
E_1	Attack	–	Strong	Strong	Strong
	Detection	–	None (c)	None (c)	None (c)
	Messages	–	M^+	M^+	M_{fake}
	Result	–	Uncertainty	Uncertainty	Full failure
E_2	Attack	–	Strong	Strong	Strong
	Detection	–	None (c)	None (c)	None (c)
	Messages	–	M^+	M_{fake}	M^+
	Result	–	Uncertainty	Full failure	Uncertainty
A	Result	–	Failure	Failure	Failure
G_{E_1}	Detection	–	None (c)	None (c)	None (c)
	Security	–	Uncertain E	Restored	Uncertain E
G_{E_2}	Detection	–	None (c)	None (c)	None (c)
	Security	–	Uncertain E	Uncertain E	Restored
Case 3		**–**	**–**		
Agent	**Feature**				
E_1	Attack	–	Classical		
	Detection	–	(Post, ∃) failure		
	Messages	–	None		
	Result	–	Failure		
E_2	Attack	–	Classical		
	Detection	–	(Post, ∃) failure		
	Messages	–	None		
	Result	–	Failure		
A	Result	–	Detection (duplicates)		
G_{E_1}	Detection	–	(Post) ∃		
	Security	–	Enforced		
G_{E_2}	Detection	–	(Post) ∃		
	Security	–	Enforced		

Case 1: E_1 and E_2 know each other as honest. If E_1 is no longer listening on the network, only E_2 can place an erase request in step (3) and thus can acquire the message M with certainty. If the competitor E_1 continues to eavesdrop, the dominant intruder can fail to acquire M* whenever $E_2 \notin$ canSee. If, on the other hand, it is the attacker at disadvantage (E_1) that is not in canSee, then E_1 fails regardless of the number of fake messages.*
Case 2: E_1 and E_2 know each other as dishonest.
Case 3: E_1 and E_2 are unaware of each other.

accessibility to the basic building block of the security mechanism is not granted but rather fought for. Competitive attackers deal with their competitors precisely by attempting to secure a privileged position with respect to the basic mechanism.

In Fiazza et al. (2011a,b), we considered the Boyd–Mathuria Example (BME), a deliberately vulnerable protocol introduced in Boyd and Mathuria (2003). For completeness, **Table 7** shows the definition of the protocol and a classical attack against it, and

TABLE 6 | Overall SRA3P results, detailed view of cases 4, 5 and 6.

Case 4A: E_1 starts the attack

Case 4B: E_2 starts the attack

Agent	**Feature**	**E_1 stops**	**canSee(M^*)=**		
			{E_1, E_2}	**{E_1}**	**{E_2}**
E_1	Attack	Classical	Classical	Classical	Classical
	Detection	None	(Post ∃) uncertainty	(Post ∃) uncertainty	None
	Messages	M_{fake}	M^+	M^+	M_{fake}
	Result	Full failure	Uncertainty	Uncertainty	Full failure
E_2	Attack	Cl → Str	Cl → Str	Cl → Str	Cl → Str
	Detection	(In) trace	(In) trace	(In) trace	(In) trace
	Messages	$M!$	$M!$	none	$M!$
	Result	Dominance	Success	Failure	Dominance
A	Result	Failure	Failure	Failure	Failure
G_{E_1}	Detection	None	Post (∃)	Post (∃)	None
	Security	Compromised	Compromised	Restored	Compromised
G_{E_2}	Detection	(In) label	(In) label	(In) label	(In) label
	Security	Restored	Uncertain E	Uncertain E	Restored

Case 5A: E_1 starts the attack

Case 5B: E_2 starts the attack

Agent	**Feature**	**E_1 stops**	**canSee(M^*) =**		
			{E_1, E_2}	**{E_1}**	**{E_2}**
E_1	Attack	Classical	Classical	Classical	Classical
	Detection	None	(Post ∃) uncertainty	(post ∃) Uncertainty	None
	Messages	M_{fake}	M^+	M^+	M_{fake}
	Result	Full failure	Uncertainty	Uncertainty	Full failure
E_2	Attack	Strong	Strong	Strong	Strong
	Detection	None (c)	None (c)	None (c)	None (c)
	Messages	$M!$	$M!$	None	$M!$
	Result	Dominance	Success	Failure	Dominance
A	Result	Failure	Failure	Failure	Failure
G_{E_1}	Detection	None	Post (∃)	Post (∃)	None
	Security	Compromised	Compromised	Restored	Compromised
G_{E_2}	Detection	None (c)	None (c)	None (c)	None (c)
	Security	Restored	Uncertain E	Uncertain E	Restored

Case 6A: E_1 starts the attack

Case 6B: E_2 starts the attack

Agent	**Feature**	**E_1 stops**	**canSee(M^*) =**		
			{E_1, E_2}	**{E_1}**	**{E_2}**
E_1	Attack	Classical	Classical	Classical	Classical
	Detection	None	(Post ∃) uncertainty	(Post ∃) uncertainty	None
	Messages	M_{fake}	M^+	M^+	M_{fake}
	Result	Full failure	Uncertainty	Uncertainty	Full failure
E_2	Attack	Strong	Strong	Strong	Strong
	Detection	(In) label	(In) label	(In) label	(In) label
	Messages	$M!$	$M!$	None	$M!$
	Result	Dominance	Success	Failure	Dominance
A	Result	Failure	Failure	Failure	Failure
G_{E_1}	Detection	None	Post (∃)	Post (∃)	None
	Security	Compromised	Compromised	Restored	Compromised
G_{E_2}	Detection	(In) label	(In) label	(In) label	(In) label
	Security	Restored	Uncertain E	Uncertain E	Restored

Case 4: E_2 knows E_1 as honest.
Case 5: E_2 knows E_1 as dishonest.
Case 6: E_2 knows E_1 but has not yet established a belief on E_1's honesty.

TABLE 7 | The Boyd–Mathuria example protocol and a masquerading attack against it.

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

TABLE 8 | Traces for non-collaborative attacks against BME. Traces are exhaustive: E_1 and E_2 have priority over honest agents and S is honest.

T1: cases 1, 3, 4, 6	T2 and [T3]: cases 1, 3, 4, 6
(1) $A \to E_{1,2}(S) \; : A, B$ $\downarrow$ (1_1) $E_1(A) \to S \; : A, E_1$ $\uparrow$ (1_2) $E_2(A) \to S \; : A, E_2$ (2_1) $S \to A \; : M_1$ (2_2) $S \to A \; : M_2$	(1) $A \to E_{1,2}(S) \; : A, B$ $\downarrow$ (1_1) $E_1(A) \to S \; : A, E_1$ $\uparrow$ (1_2) $E_2(A) \to S \; : A, E_2$ (2_1) $S \to A \; : M_1$ (3) $A \to E_{1,2}(B) \; : \{\!|k_{AE_1}|\!\}_{k_{E_1S}}$ [(2_2) $S \to A \; : M_2$]
T4: case 2	**T5: case 5**
(1) $A \to E_{1,2}(S) \; : A, B$ $\downarrow$ (1_1)$^+$ $E_1(A) \to E_2(S) \; : A, E_1$ $\uparrow$ (1_2) $E_2(A) \to E_1(S) \; : A, E_2$	(1) $A \to E_{1,2}(S) \; : A, B$ $\downarrow$ (1_1) $E_1(A) \to E_2(S) \; : A, E_1$ $\uparrow$ (1_2) $E_2(A) \to S \; : A, E_2$ (2) $S \to A \; : M_2$ (3) $A \to E_{1,2}(B) \; : \{\!|k_{AE_2}|\!\}_{k_{E_2S}}$

Where: $M_1 = \{\!|k_{AE_1}|\!\}_{k_{AS}}, \{\!|k_{AE_1}|\!\}_{k_{E_1S}}$, $M_2 = \{\!|k_{AE_2}|\!\}_{k_{AS}}, \{\!|k_{AE_2}|\!\}_{k_{E_2S}}$

Arrows: relative order between (1_1) and (1_2) is irrelevant in determining the outcome.

Table 8 shows the non-collaborative attacks against it (the complete outcome of the interaction between two non-collaborative attackers can be found in Fiazza et al. (2011a,b). We can actually see the BME case study and the SRA3P case study [presented in Fiazza et al. (2012) and extended in this paper] as key-exchange protocols: BME by design and SRA3P by considering M to be used as a key. Still, there are fundamental differences between the results of our analyses of the case studies. This is kind of natural as, depending on the type of the protocol, there is a different type of competitive attack situation that we have to consider; in the case of BME, the result is basically yes or no, meaning that the attack succeeded or not, whereas in the case of SRA3P, an attacker must actually reason on the outcome of the attack that he carried out, i.e., whether he was really able to succeed or not. In fact, SRA3P differs significantly from BME in that success is not necessarily exclusive and in that it requires interacting with a second honest agent (S) to carry out a masquerading attack.

Moreover, in BME, attackers attack three things simultaneously: A as the target of the attack, B as the agent to masquerade, the key as the means to acquire information. All the pieces of meta-information in the opening message (meaning of the two fields and expected answer) are attacked. In SRA3P, the only known meta-information is that the message is encrypted with commutative cryptography. With less meta-information, it is far harder to move effectively in the SRA3P environment.

We learn from SRA3P that attackers can face, according to the protocol in use, a multiplicity of factors and that BME only gave a window over a rather simple case. Attackers who are successful in the SRA3P scenario have a success criterion that is far more complex than what could be suitable for BME: the panorama looks a bit more involved.

4.2. Success Criteria for Competitive Attackers and Honest Agents

The best result for a competitive attacker consists in violating security without the honest agent realizing it, and making it such that the other attackers conclude their attacks with false information (e.g., M_{fake} taken for M^* in SRA3P) without realizing it. However, as shown with SRA3P, in competitive scenarios with equal-opportunity attackers, it is not possible, in general, to ensure a complete victory under all circumstances. Hence, the result of an attack depends on the strategy and on the knowledge conditions of all the active agents. A competitive agent will try to secure the best result (compatibly with his knowledge) and he will strategically evaluate if it is preferable, for example, to risk being identified as an attacker by other agents or to increase the degree of uncertainty of the competitors.

For competitive agents (in particular, for SRA3P) any of the following factors can form part of the success criterion:

- F1 success in gaining the secret M protected by the security system (or, more generally, in invalidating the target properties of the protocol). For vulnerable protocols, a single attacker without competition always succeeds. The first priority of our competitive attackers is preserving the success of their own attacks, even in the presence of competitors.
- F2 absence of uncertainty on the secret message M.
- F3 effects on competitors: denying competitors access to the secret so that the attacker gains exclusive access to M, ideally also inducing the competitors into thinking they have succeeded.
- F4 effects on competitors: denying competitors certainty or recognizability of the secret.
- F5 post-protocol detection of the attacker's identity: in-protocol detection (of attacks) by honest agents thwarts the attack and corresponds to failing the goal at point 1.
- F6 possible identification as an attacker by other attackers. Knowing the dishonesty of an agent is an advantage, therefore attackers seek to limit the situations in which they can be detected or identified through an explorative spy rule. A good example of this strategy is the difference between the two non-collaborative attacks against SRA3P: employing a direct *send* to the competitor or relying on the prediction that the competitor will spy.
- F7 possible identification of competitors, thereby acquiring a strategic advantage for later protocol runs.

For honest agents, we can distinguish five relevant conditions, each associated to a different level of alarm:

1. No attacker has gained the secret, which has correctly reached the intended recipient (security). This does not hold for SRA3P as it is attackable.

2. No attacker has succeeded in gaining the secret, but the secret has not reached its intended recipient (stalemate, deadlock). For SRA3P, this occurs whenever the initiator detects duplicate messages before step (3), e.g., in case 3.
3. One or more attackers have gained the secret but the honest agent has detected the attack (restart).
4. One or more attackers have gained the secret, the honest agent has detected the attack, and acquired knowledge on the attacker's identity (retaliate and restart).
5. One or more attackers have gained the secret but the attack has not been detected (security failure).

4.3. Priorities and Attacker Strategies

The strategy he employs determines to what extent the goals of a competitive attacker can be accomplished. The foundation of an attacker's strategy is the way he weighs the different success factors to configure his own success criterion: the priorities with which an attacker is configured make up the "attacker's personality." Let us examine two concrete cases as examples.

4.3.1. Contact Information for Social Engineering

Consider a network environment in which each data are sensitive and can yield a profit. Here, we consider privacy violations in which attackers can acquire contact information of individuals with particular characteristics (e.g., influential, rich, or informed); attackers wish to make use of the contacts either to engage in social engineering or in telemarketing.

The bonus associated to exclusive access to the secret is expressed in terms of a secondary resource, such as the waiting time before the line is free or the time necessary to gain the victim's trust in social engineering. If others are attempting to do the same, the attacker has to invest more effort. The secondary resource can be more or less precious; in telemarketing, it is worth very little, whereas for social engineering practices, such as identity theft, it is important that the victim not be alarmed.

In this scenario, the cost of uncertain or wrong information can range from extremely high to near-zero. If a telemarketer is not certain of which of the n messages contains a candidate's phone number, he can simply try them all, at the cost of a phone call's time each. However, social engineering may demand a large personal investment in terms of time and effort for each victim. When an attacker in this scenario successfully acquires a secret without uncertainty, he has realized a gain; clearly, he will try to do so as many times as possible. What we wish to emphasize is that succeeding in any one given run is worth something *per se* in the eyes of the attacker.

The prioritization of success factors typical of an *aggressive attacker* (fit for the situation we have described) is:

$$F1 \text{ or } F2 \quad > \quad F5 \quad > \quad F6 \text{ or } F7 \quad > \quad F3 \text{ or } F4$$

where on the left, we find the highest priority; for as long as the goals in the lower priorities are reachable, attackers will pursue them – disregarding all actions that could lead to pursuing lesser goals at the expense of higher-priority ones. Although priorities may be changed dynamically within a sequence of protocol runs, in the scenario, we have described that attackers have no convenience in switching to other strategies.

4.3.2. Access to Project Documentation in Industrial Espionage

For a more *cautious attacker*, gaining the credentials to access the network environment of the target industry required a considerable prior effort. It is thus a priority for attackers to avoid needing new credentials: attackers have a vested interest in avoiding detection, most notably by honest agents, as they could report to the system administrator.

Consider the case in which keys protected by the protocol are used to encrypt individual digital documents containing the design documentation of a large project, e.g., the plans for a car, a helicopter, a system made of multiple *ad hoc* components. It is acceptable to succeed in acquiring the secret key only sometimes, for as long as the possibility is kept open to continue attacking the protocol.

In industrial or military espionage, even scraps of information are worth it. Breaking a single protocol run is only partial success, in that it gives access to a small part of the documentation. The goals are fully met when the number of stolen pieces rises: the attacker can then progressively build the bigger picture.

A second factor to consider is that monitoring the progress of the project can lead to identifying the moment in which one is willing to risk his cover to attempt gaining conclusive or key information, switching to the strategy of an *aggressive attacker*. Cautiousness is meant to preserve the possibility of acquiring information aggressively in the future.

The cost of uncertainty on information is extremely limited: if in doubt on which one is the decryption password for a given document, try them all. The cost is so low that factors F2 and F4 are not even a priority. The success criterion of a cautious attacker is:

$$F5 \quad > \quad F1 \quad > \quad F6 \quad > \quad F3 \text{ or } F7$$

5. Defending Vulnerable Protocols Against Attacks

As shown in the case studies, we considered here and in our other papers, in competitive scenarios with equal-opportunity, it is not possible for an attacker to ensure that an attack is successful under all circumstances. The outcome depends on the strategy and knowledge conditions of all the active agents, on the visibility of erased messages to other attackers ($canSee \neq \{E_1, E_2\}$) and on protocol-specific features.

The presence of an independent active attacker constrains the success of otherwise sure-fire attacks, so, to make use of the emergent interference between concurrent attacks, it is necessary to ensure that attacks mounted by an attacker are immersed in a competitive environment. To this end, we can construct an additional non-malicious attacker, who carries out attacks against the protocol, discards the data acquired during the protocol run (whenever he acquires any), and reasons on the basis of what he can observe, to assist honest agents in detection tasks.

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 should be on the basis of flags that are independent of the specific attack trace and, in general, 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. Such a *guardian agent* is formally an attacker and can therefore be configured with knowledge of the attack trace(s). 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. 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 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. Then guess-and-test strategies can be used to acquire an understanding of the second attacker's identity.

Guardians are introduced as means to implement active defenses. Although their presence is beneficial to honest agents, guardian agents are formally attackers and are configurable as such. As we have seen in Section 4.3, attackers can have different concrete takes on which features of the information are important; because of the specific ways attackers intend to use the information, they will treat differently the success factors. The two tasks entrusted to a guardian are defending data and discovering malicious agents, so as to be better equipped to defend data in later protocol runs. Any attacker who is unwilling (or incapable) to exploit the data protected by the protocol (e.g., M in SRA3P) can serve as a guardian, because by its presence, he induces interference and mitigates the success of malicious attackers. Just like attackers come in different flavors, guardians can also have different "personalities."

For concreteness, let us discuss how a guardian may be effective with respect to the factors presented in Section 4.2. The highest priority is F3: exclusive access to the data protected by the protocol means that no competitors can succeed in violating A's communications. Detection of competitors (F7) is also important, because knowledge of their presence or identity helps in establishing the dominance that is needed for exclusivity, and thus contributes to better performance in subsequent runs. For F4: if it is not possible to dominate the competitors, the guardian will at least strive to make competitors pay the consequences of uncertainty. Finally, F6: avoiding detection so that competitors cannot attempt to circumvent the effectiveness of the guardian. All the other factors are irrelevant in the decisional processes of a guardian.

Table 9 shows the effects of introducing a guardian G for SRA3P, configured as one of the competitive attackers described in the case study. Compared to the guardian for BME

TABLE 9 | Effects of introducing a guardian *G* for SRA3P when attacker *E* is active.

canSee	**Case 2**	**Case 3**	**Cases 1 + 4,5,6** $E \in Attend_G$	**Cases 4,5,6** $G \in Attend_E$
{*E*, *G*}	$\sim^{+}$	$\surd$	$\sim$	
{*G*}	$\surd$	$\surd$	$\surd$	$\surd$
{*E*}	$\sim^{+}$	$\surd$	$\sim$	

G's active interference results in E failing to acquire the secret ($\surd$), in being sometimes uncertain ($\sim$) or in being always uncertain ($\sim^{+}$).
G operates according to the same strategy as the attackers in the corresponding case study. G 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.

TABLE 10 | Effects of introducing a guardian *G* for BME when attacker *E* is active.

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

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.

(**Table 10**) shows the results of Fiazza et al. (2011a,b), a guardian for SRA3P appears to be less effective, in that it prevents E from successfully carrying out his attack in fewer cases. However, it must be noted that SRA3P is a much harder protocol to defend because it does not entail that attacker success is mutually exclusive. Remarkably, G can be effective even when he is not aware of E's presence. The effectiveness of a guardian for SRA3P is comparable to the case of BME, if honest agents can detect and mount retaliatory attacks whenever attackers guess the wrong secret and use it to communicate with honest agents.

We refer to our other publications for discussions on how to construct guardians for other protocols and where to place guardians in network configurations.

6. Conclusion and Future Work

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 vulnerable protocols still being in use.

Along the line of work presented in this paper, we have extended the investigation of the SRA3P protocol, which differs significantly from BME in that success is not necessarily exclusive. The goal of this additional investigation is to bring into focus how the salient features of each protocol are reflected in the possible mechanisms of interference and thus in the identification of potential attacks and in the development of defenses against such attacks.

Our case studies show that in a non-collaborating two-attacker scenario, it is possible to build a defense against an attack that was not possible in the standard one-attacker scenario. This statement is the counterpart of the classical result on *n*-to-1 reducibility for collaborating attackers, and the counterexamples show that exhaustive searches for (guardian-based) defenses cannot be carried out in reduced-complexity settings, as they require at least two attackers.

In our publications so far, we have formalized a framework for non-collaboration, described the notion of competitive attacker, shown a number of results on concurrent attacks giving rise to interference, delineated a strategy for defending protocols, and presented results on the effectiveness of a network guardian configured as a competitive attacker.

We are currently working at applying our approach for the analysis of more complex security protocols and properties than those we considered here. We are also working at implementing our approach. 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. For a successful implementation, we plan, as we proposed in Fiazza et al. (2011c, 2012), to recruit techniques from AI and robotics (fields that traditionally have a complex notion of agent) and to identify network topologies that allow for effective guardians as we started doing in Peroli et al. (2014).

Acknowledgments

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 by the PRIN 2010-2011 Project "Security Horizons".

References

Arsac, W., Bella, G., Chantry, X., and Compagna, L. (2009). *Validating Security Protocols Under the General Attacker, in ARSPA-WITS*, Vol. 5511. York, UK: Springer, LNCS, 34–51.

Arsac, W., Bella, G., Chantry, X., and Compagna, L. (2011). Multi-attacker protocol validation. *J. Automat. Reas.* 46, 353–388. doi:10.1007/s10817-010-9185-y

Basin, D., Caleiro, C., Ramos, J., and Viganò, L. (2011). Distributed temporal logic for the analysis of security protocol models. *Theor. Comp. Sci.* 412, 4007–4043. doi:10.1016/j.tcs.2011.04.006

Basin, D., Capkun, S., Schaller, P., and Schmidt, B. (2009). *Let's get Physical: Models and Methods for Real-World Security Protocols, in TPHOLs*, Vol. 5674. Munich: Springer, LNCS, 1–22.

Bella, G., Bistarelli, S., and Massacci, F. (2003). *A Protocol's Life After Attacks, in Security Protocols XI*, Vol. 3364. Cambridge, UK: Springer, LNCS, 3–18.

Bella, G., Bistarelli, S., and Massacci, F. (2008). Retaliation against protocol attacks. *J. Inform. Assur. Secur.* 3, 313–325.

Boyd, C., and Mathuria, A. (2003). *Protocols for Authentication and Key Establishment*. Springer. doi:10.1007/978-3-662-09527-0

Caleiro, C., Viganò, L., and Basin, D. (2005). Metareasoning about security protocols using distributed temporal logic. *Electron. Notes Theor. Comput. Sci.* 125, 67–89. doi:10.1016/j.entcs.2004.05.020

Caleiro, C., Viganò, L., and Basin, D. (2006). On the semantics of Alice & Bob specifications of security protocols. *Theor. Comp. Sci.* 367, 88–122. doi:10.1016/j.tcs.2006.08.041

Carlsen, U. (1994). *Cryptographic Protocol Flaws, in CSFW-7*. Franconia, NH: IEEE CS, 192–200.

Clark, J., and Jacob, J. (1997). *A Survey of Authentication Protocol Literature: Version 1.0*. Citeseer.

Comon-Lundh, H., and Cortier, V. (2003). *Security Properties: Two Agents are Sufficient, in ESOP*, Vol. 2618. Warsaw: Springer, LNCS, 99–113. doi:10.1016/j.scico.2003.12.002

Dolev, D., and Yao, A. C. (1983). On the security of public key protocols. *IEEE Trans. Inform. Theory* 29, 198–208. doi:10.1109/TIT.1983.1056650

Fiazza, M.-C., Peroli, M., and Viganò, L. (2011a). *Attack Interference: A Path to Defending Security Protocols, in E-Business and Telecommunications*, Vol. 314. Seville: Springer, CCIS, 296–314. doi:10.1007/978-3-642-35755-8_21

Fiazza, M.-C., Peroli, M., and Viganò, L. (2011b). *Attack Interference in Non-Collaborative Scenarios for Security Protocol Analysis, in SECRYPT*. Seville: SciTePress, 144–156.

Fiazza, M.-C., Peroli, M., and Viganò, L. (2011c). *Security Protocols as Environments: A Lesson from Non-collaboration, in TrustCol*. Orlando: IEEE CS Press.

Fiazza, M.-C., Peroli, M., and Viganò, L. (2012). *An Environmental Paradigm for Defending Security Protocols, in CTS*. Denver, CO: IEEE, 427–438. doi:10.1109/CTS.2012.6261087

Peroli, M., Viganò, L., and Zavatteri, M. (2014). *Non-Collaborative Attackers and How and Where to Defend Flawed Security Protocols (Extended Version), in Security Protocols XXII*. Cambridge, UK: Springer, 69–90.

Schaller, P., Schmidt, B., Basin, D., and Capkun, S. (2009). *Modeling and Verifying Physical Properties of Security Protocols for Wireless Networks, in CSF*. Port Jefferson, NY: IEEE CS.

Syverson, P., Meadows, C., and Cervesato, I. (2000). *Dolev-Yao is No Better than Machiavelli, in WITS*. Geneva. Available at: http://www.dsi.unive.it/IFIPWG1_7/wits2000.html

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.