Journal of Information Assurance and Security 2 (2008) 89-101

Retaliation Against Protocol Attacks

Giampaolo Bella
Dip. di Matematica e Informatica
Università di Catania, Italy
giamp@dmi.unict.it
Stefano Bistarelli
Dip. di Scienze
Università "G. D'Annunzio" di Pescara, Italy
bista@sci.unich.it
and
Istituto di Informatica e Telematica, C.N.R., Pisa, Italy
stefano.bistarelli@iit.cnr.it
Fabio Massacci
Dip. di Informatica e Telecomunicazioni
Università di Trento, Italy
massacci@ing.unitn.it

Abstract: Security protocols intend to give their parties reasonable assurance that certain security properties will protect their communication session. However, the literature confirms that the protocols may suffer subtle and hidden attacks. Flawed protocols are customarily sent back to the design process, but the costs of reengineering a deployed protocol may be prohibitive. This paper outlines the concept of *retaliation*: who would steal a sum of money today, should this pose significant risks of having twice as much stolen back tomorrow? When ethics is left behind, attacks are always balanced decisions: if an attack can be retaliated, the economics of security may convince the attacker to refrain from attacking, and us to live with a flawed protocol. This new perspective requires a new threat model where any party may decide to subvert the protocol for his own sake, depending on the risks of retaliation. This threat model, which for example is also suitable to studying non-repudiation protocols, seems more appropriate than the Dolev-Yao model to the present technological/social setting. It is demonstrated that machine-assisted protocol verification can can effectively be adapted to the new threat model.

I. Introduction

A security protocol is a social behaviour that principals of a distributed system must follow to obtain some important collective benefits in terms of security. For the good principals, it is sufficient to state some clear, understandable, and acceptable rules describing how to execute the security protocol correctly, namely by the book. Because they are good principals, they will conform to the rules, and behave as the protocol prescribes.

The bad principals, by definition, will not conform to the rules and, rather, will execute the protocol arbitrarily, that is incorrectly.

Classical research in distributed systems and security starts off exactly from the need to counter the disruptive behaviour of the bad principals. Research efforts have focused on designing a protocol so that if the good principals outnumber the bad ones, the collective benefits will be achieved regardless of the bad principals' behaviours. Another perspective aims at limiting the bad principals' profit, regardless of how many or how smart they are [11]. The general line of research seems towards proving that those who conform to the protocol are somewhat safeguarded with their own aims. Our contribution substantially lengthens this line.

The results of verification have normally a significant impact on design. Whenever verification denounces an attack, the protocol ought go back to the design phase. The dominant mode following the discovery of an attack is that the original design is a complete failure. Yet, the attack is often only a little bug. We routinely live with software with bugs, houses with cracks and cars with not so good tyres. Provided the buggy part does not get in the critical path, we can use the system perfectly well. These considerations lead us to wondering what may happen after an attack takes place. Can we still get something useful from the protocol? We expect to obtain deeper insights about the entanglements of a protocol by continuing its analysis after an attack is pinpointed. In other terms, we are crossing a doorstep that usually stops researchers and sends them to publishing their findings.

Received May 29, 2008

1554-1010 $ 03.50 Dynamic Publishers, Inc.

Our analysis helps us understand whether it is at all possible to threaten the bad principals exactly when they execute the protocol incorrectly. In the real world, a virtuous behaviour is imposed on people by taking measures of real security such as hardening the windows against crash. There is a perfect simile with security protocols so far. However, the real world also relies on countermeasures of security so that the vandals who, despite the rules, crash the windows are jailed. Our simile flickers here but must not. People balance the advantages of breaking the law on one side with its consequences on the other side. We observe that this applies to both the real and the digital world. So, if we convince the protocol participants to weigh up the benefits of an incorrect execution with the consequent threats, they would opt to execute the protocol correctly if the threats were heavier.

The essence of *retaliation* for security protocols has come clear. Let us consider Lowe's famous attack to the public-key Needham-Schroeder protocol [13]. The attack entitles the bad principal to ask for a transfer of money. Would he really steal a sum of money if the threats that twice as much would consequently be stolen to him were significant? This kind of analysis opens up the ground to novel, realistic considerations about security protocols. When an attack is discovered, it is worth studying further to verify if it can be retaliated. An affirmative conclusion, perhaps supported by appropriate risk analysis, may let us decide to keep the protocol in use as it stands. If redesign is costly, retaliation may signify that a flawed protocol can still achieve a sufficient and stable level of security.

It is unfortunate that retaliation is normally accepted and legitimised in military contexts. Our findings appear to establish this concept also in commercial systems regulated by security protocols. However, this may turn out useful to all protocol participants. Attackers usually leave ethics behind and adopt a simple cost/benefit conduct. Either the media or their own mischievousness — which makes themselves victims of retaliation attacks — will have informed them that certain attacks can be retaliated. It can be deduced that, at that stage, their simple code of conduct will convince them to retreat. As a consequence, the good principals will be safe enough with a flawed protocol that permits retaliation.

The present paper builds on top of ideas that we informally sketched [4]. The presentation gains a precise formulation of the novel threat model that supports the notion of retaliation. Moreover, all definitions are presented formally here. Finally, the novel concept of *out-of-band challenge* is advanced. Because each principal minds his own business with any legal (if he is good) or also illegal (if he is bad) means, he can issue out-of-band challenge messages to suspect or detect that something dodgy happened.

The organisation of this manuscript is simple. The presentation opens up by triggering the reader's intuition with an example (§II). Only at that stage are the key formal elements introduced (§III), and the novel threat model specified (§IV). The continuation of protocol analysis after an attack is found (§V) is central to our work. This paper extends a previous version [5] with the first findings on machine-assisted formal protocol verification under the new threat model that encompasses retaliation (§VI). These findings have never been published before. Some conclusions terminate the presentation (§VII).

II. Retaliation in the Public-Key Needham-Schroeder Protocol

The popular public-key protocol due to Needham-Schroeder [15] is a good starting point to our presentation. The notation can be easily summarised as follows.

- Cryptographic keys are denoted by letter K in general. Each letter may feature a principal name as a subscript, expressing the principal who knows the key.
- Nonces are denoted by letter N. Each letter may feature a principal name as a subscript, expressing the principal who invented the nonce.
- The message concatenation operator is denoted by a comma.
- The message encryption operator is denoted by external curly braces featuring the encryption key as a subscript. This paper only features asymmetric encryption.

Having seen the basic protocol notation, the actual protocol can be found Figure 1.

1. $A \rightarrow B : \{\!|Na, A|\!\}_{Kb}$
2. $B \rightarrow A : \{\!|Na, Nb|\!\}_{Ka}$
3. $A \rightarrow B : \{\!|Nb|\!\}_{Kb}$

Figure. 1: The public-key Needham-Schroeder protocol

The goal of this protocol is *authentication*: at completion of a session initiated by A with B, principal A should get evidence to have communicated with B and, likewise, principal B should get evidence to have communicated with A. Assuming that encryption is perfect and that the nonces are truly random, authentication is achieved here by exchange of nonces. Upon reception of Na inside message 2, A should be allowed to conclude that she is interacting with B, the only principal who could retrieve Na from message 1. In the same fashion, upon reception of Nb inside message 3, B should be allowed to conclude that he is interacting with A, the only principal who could retrieve Nb from message 2. However, let us consider Lowe's attack reported in Figure 2.

The attack consists in a malicious principal C masquerade as a principal A with a principal B, after A initiated a session with C. This scenario, which sees C interleave two sessions,

1. $A \rightarrow C : \{\!|Na, A|\!\}_{Kc}$
   $1'$. $C \rightarrow B : \{\!|Na, A|\!\}_{Kb}$
   $2'$. $B \rightarrow A : \{\!|Na, Nb|\!\}_{Ka}$
2. $C \rightarrow A : \{\!|Na, Nb|\!\}_{Ka}$
3. $A \rightarrow C : \{\!|Nb|\!\}_{Kc}$
   $3'$. $C \rightarrow B : \{\!|Nb|\!\}_{Kb}$

Figure. 2: Lowe's attack to the Needham-Schroeder protocol

indicates failure of authentication of A with B, which follows from failure of confidentiality of Nb. Lowe also reports that, if B is a bank for example, C can steal money from A's account by sending a single message (Figure 3). Upon reception of the two nonces of the session with A, the bank B would honour the request believing it came from the account holder A. The sender label can be changed at will, and notoriously is unreliable.

4. $C \rightarrow B$:
$\{\!|Na, Nb,$ "Transfer £1000 from A's account to C"$|\!\}_{Kb}$

Figure. 3: Completion of Lowe's attack

A more thorough confidentiality analysis with soft-constraints [3] reveals that, as a by-product of Lowe's attack, B has learnt nonce Na, which was invented by A to be shared with C only. Formally speaking, this already is a violation of the protocol, because it is against its underlying policy. On one hand, it may not seem a major observation, as we already know that the protocol *is* flawed and is flawed exactly in terms of confidentiality of the nonces. On the other hand, we wonder what may happen in practice if B later realises the significance of the nonce he indeliberately received, and hence decides to take advantage of it. In terms of security analysis, it is not interesting to study how B could realise that: if a bad principal has a key ring with many keys, he may systematically try them all at the available locks. In most cases, there will be some "proximity" between the key ring and the potential victim locks, such as within the same newsgroup, or the same LAN, or the same institution. However, the very consequences of the most pessimistic case that sees B exploit Na are the focus here: B can also rob the robber by a single message, as described in Figure 4. Upon reception of the two nonces of the session with C, the bank A would honour the request believing it came from the account holder C.
This is a form of indirect retaliation: C robs A through B, hence B robs C through A. It may turn out to be more or less appealing in practice. Nevertheless, what can be learnt is that something significant may follow after an attack happens in the first place, and therefore we should also look beyond protocol attacks. It is something that is made possible exactly because the first attack took place, so it is not just another attack. Also, it is imprecise to see this scenario as a classical cascade of attacks because the victim of the first attack is not the same as that of the retaliation attack. The most appropriate connotation indeed appears to be that of retaliation: because something happens, something else can happen against that.

$4'$. $B \rightarrow A$:
$\{\!|Na, Nb,$ "Transfer £2000 from C's account to B's"$|\!\}_{Ka}$

Figure. 4: Retaliating Lowe's attack

It is interesting to point out that the retaliation attack is possible because not only does Lowe's attack disclose Nb to C, but it also reveals Na to B. The same pattern lies behind classical attacks to other protocols, such as Splice/AS and the Helsinki protocol [7]. Hence, also those attacks can be retaliated. A fundamental prerequisite to study retaliation attacks in detail is to allow the principals to change behaviour from unaware mediator to active attacker, as is the case of B in the example above, or from victimiser to victim, as is the case of C. It seems that the classical Dolev-Yao threat model consisting in a super-potent attacker is inappropriate to the present technological/social setting. Today, each principal may have capacity and competence to decide to act illegally for his own sake. This change to the threat model is defined below (§IV) but some basic terminology must be introduced first.

III. Basic terminology

For simplicity, in the following we do not specify a more or less free algebra of messages, since this is only needed when modelling a specific protocol with a specific formal method. We only assume one exists, so that messages are elements of this algebra and can be suitably identified by a number to avoid ambiguity. Following Backes et al. [1] we uniquely identify each message so that even if a principal takes a message and simply forwards it to another one, it will be denoted by a different identifier. The underlying algebra of messages would then tell us that the messages are indeed "equal in content". Such a notion can then be used when modelling a specific protocol step.

Definition 1 (Events) *An* Event *is one of the following actions:*

- *a principal* sends *a message to another principal; it is denoted by a 4-uple* $\mathsf{s}(A : A' \rightarrow B[\#])$ *mentioning the actual sender A, the alleged sender A', the recipient B, and the message number $\#$;*
- *a principal* receives *a message; it is denoted by a tuple*

$\mathsf{r}(A:\#)$ *mentioning the receiver* A *and the message number* $\#$.

Example 1 *Consider the Needham-Schroeder protocol (Figure 1). Its events and messages can be easily formalised as follows. The event whereby* A *initiates with* B *can be denoted by* $\mathsf{s}(A:A\rightarrow B[1])$*; the event whereby* B *receives the message can be denoted by* $\mathsf{r}(B:1)$*; the event whereby some* C *intercepts the same message can be denoted by* $\mathsf{r}(C:1)$.

Definition 2 (Traces) *A* Trace T *is a list of events formalising a specific network history. It must respect Lamport's causality principle and the unique identification of messages by Backes et al. [1]: each sending event must precede the corresponding receiving event and each sending event must introduce a message with a new formal identifier.*

Example 2 *Consider the network history on which Lowe's attack (Figure 2) takes place. It can be formalised by the trace:*

$$T_{Lowe}=\left[\begin{array}{l}\mathsf{s}(A:A\rightarrow C[1]),\mathsf{r}(C:1),\\ \mathsf{s}(C:A\rightarrow B[1']),\mathsf{r}(B:1'),\\ \mathsf{s}(B:B\rightarrow A[2']),\mathsf{r}(C:2'),\\ \mathsf{s}(C:C\rightarrow A[2]),\mathsf{r}(A:2),\\ \mathsf{s}(A:A\rightarrow C[3]),\mathsf{r}(C:3),\\ \mathsf{s}(C:A\rightarrow B[3'])\end{array}\right]$$

It can be seen that the reception events in T_{Lowe} *confirm that* C *learns nonce* Nb *and* B *learns nonce* Na.

Definition 3 (Trace Projections and Extensions) A Projection $T/\mathcal{A}$ *of a trace* T *over a set of principals* $\mathcal{A}$ *is the sublist of events in* T *that are performed by some principal in* $\mathcal{A}$. *An* Extension T' *of a trace* T *is any trace beginning with* T. *In symbols:* $T\sqsubseteq T'$*; the concatenated trace* $T_1;T_2$ *is such that* $T_1\sqsubseteq T_1;T_2$.

A remark is necessary about trace projection. Let us suppose that a trace features the event whereby A sends a message to B. This event certainly belongs to the projection of the trace over set $\{A\}$, but not over the set $\{B\}$ because reception is not guaranteed in general. Likewise, if the original trace features the event whereby A receives a message, this event belongs to projection of the trace over $\{A\}$. There is no strong relation between the projection and extension operators, so that in general $T/\{A\}\not\sqsubseteq T$.

Example 3 *Consider the trace representing Lowe's attack. It can be easily projected over the attacker* C *as:*

$$T_{Lowe}/\{C\}=\left[\begin{array}{l}\mathsf{s}(C:A\rightarrow B[1']),\mathsf{r}(C:2'),\\ \mathsf{s}(C:C\rightarrow A[2]),\mathsf{r}(C:3),\\ \mathsf{s}(C:A\rightarrow B[3'])\end{array}\right]$$

.

Example 4 *Consider the example trace:*

$$T'=[\mathsf{s}(A:A\rightarrow C[1]),\mathsf{r}(C:1),\mathsf{s}(C:A\rightarrow B[1'])]$$

It follows that $T'\sqsubseteq T_{Lowe}$*, but* $T'\not\sqsubseteq T_{Lowe}/\{C\}$ *because* A*'s sending the first message does not appear in* $T_{Lowe}/\{C\}$*. Also,* $T_{Lowe}/\{C\}\not\sqsubseteq T_{Lowe}$.

Classical security terms such as spoofing and sniffing can be easily defined formally using the notion of trace. A principal *spoofs* a message on a trace if the trace features an event in which the actual sender is different from the alleged sender. A principal *intercepts* a message meant for someone else on a trace if the trace features an event whereby the principal receives the message but no event whereby the intended recipient of the message receives it. If a trace on which an interception event takes place is extended with the event whereby the intended recipient of the intercepted message actually receives it, then the interception event should be more correctly addressed as a *sniffing* event. It means that these notions only make sense exactly with respect to a trace and precisely to the very trace under consideration. By contrast, they are pointless on their own.

A *formal protocol model* generically is the set of all possible traces induced by the protocol. It can be defined in the formal model of choice (CSP [20], Inductive Method [17], Strand Spaces [22], etc). It is denoted by (variants of) the Greek letter Π.

IV. $\mathcal{BUG}$: A New Threat Model

A subtler classification of principals than the classical spy/non-spy one is needed. Our interest is in a social taxonomy reflecting whether the principals behave legally or not, rather than in notions such as initiator or responder. The taxonomy is taken as a threat model for the security considerations that follow.

Definition 4 ($\mathcal{BUG}$ Threat Model) *The* $\mathcal{BUG}$ *threat model partitions the principals according to three, disjoint, social behaviours: the* bad*, the* good *and the* ugly *principals. These are defined as follows:*

Bad *principals are attempting to break the protocol for their own illegal benefits. They may or may not collude with each other. They are denoted by (variants of) the calligraphic letter* $\mathcal{B}$.

Ugly *principals are acting with no precise social/legal commitment: they may follow the protocol and may, deliberately or not, let the bad principals exploit them. They are denoted by (variants of) the calligraphic letter* $\mathcal{U}$.

Good *principals follow the protocol rules, and are exactly those who should enjoy the protocol goals exactly by conforming to its rules. They are denoted by (variants of) the calligraphic letter* $\mathcal{G}$.

Our taxonomy is both similar to and different from the Dolev-Yao [10] simple classification of principals. It is similar in the admission that someone can act illegally. We are however accounting for a *set* of bad principals rather than for a single spy, signifying that more than one principal may

want to subvert the protocol. Crucially, each bad principal may want to act by himself, as it is realistic nowadays. By contrast, the Dolev-Yao spy is the logical product of any set of colluding principals, as it was more realistic decades ago when computer networks were rare.

A distinction is necessary between good/bad and ugly participants because we want to discuss what happens after an attack. It is important to identify the participants who should have benefited from the protocol goals (the good), the participants who actually benefited from the flaw (the bad) and finally those who took part in the session and indeliberately contributed to the flaw (the ugly). Because the principals can change role, for example from good to bad, by performing some event, the taxonomy depends on the specific trace under consideration. This relation requires further specification, but for simplicity it is sufficient to clarify that if a specific partition (of the principals into the roles) underlies a trace, another partition can underly an extension of the given trace.

In the original Dolev-Yao model, and in some later more complicated incarnation such as the Bellare-Rogaway [6] model, the ugly and the bad were grouped together: the intruder can use as oracle any stage of the protocol. However this does not distinguish who gained from the protocol failure. But such a clear distinction is always present in the informal description of an attack in a research paper: sentences such as "and thus A can impersonate B", "C can learn M" etc. mark exactly the notion of who is gaining. However, to impersonate Bob, it might be the case that Alice needs to exploit Ive's participation in the protocol, in which case Ive would be playing, deliberately or not, the role of an ugly principal.

Before moving on to a formal example, we assume the existence of a predicate over a protocol trace that evaluates to true if the trace contains an attack according to some suitable definition. The predicate takes as parameters also the specific social behaviours of the principals on that trace: $\mathsf{A}(T, \mathcal{B}, \mathcal{U}, \mathcal{G})$. Clearly, additional predicates can be introduced formalising specific attacks, but we can do with one for the sake of presentation.

Example 5 *Consider Lowe's attack (Figure 2) to the Needham-Schroeder protocol and the trace T_{Lowe} (Example 2) formalising it. On this trace it can be observed that: C is the subject of the attack, the attacker; A is just playing by the rules with no deliberate commitment; B is the object of the attack, the victim. So, we define:*

- $\mathcal{B} = \{C\}$
- $\mathcal{U} = \{A\}$
- $\mathcal{G} = \{B\}$

It follows that $\mathsf{A}(T_{Lowe}, \mathcal{B}, \mathcal{U}, \mathcal{G})$ holds.

Example 6 *Consider the completion of Lowe's attack (Figure 3). It can be formalised as an extension of the trace T_{Lowe} (Example 2) as:*

$$T_1 = T_{Lowe}; [\mathsf{s}(C : A \rightarrow B[4]), \mathsf{r}(B : 4)]$$

On this trace it can be observed that: C is the subject of the attack, the attacker; B is just playing by the rules with no deliberate commitment; A is the object of the attack, the victim. So, we define:

- $\mathcal{B}_1 = \{C\}$
- $\mathcal{U}_1 = \{B\}$
- $\mathcal{G}_1 = \{A\}$

It follows that $\mathsf{A}(T_1, \mathcal{B}_1, \mathcal{U}_1, \mathcal{G}_1)$ holds.

The two previous examples show that the social roles that the agents play vary from the trace T_{Lowe} formalising Lowe's attack, to the trace T_1 formalising its completion with the illegal money transfer. It is clear that, while Lowe's attack directly impacts B, the consequent theft impacts A.

Example 7 *Consider our continuation of Lowe's complete attack (Figure 4). It can be formalised as an extension of the trace T_1 (Example 6) as:*

$$T_2 = T_1; [\mathsf{s}(B : C \rightarrow A[4']); \mathsf{r}(A : 4')]$$

On this trace it can be observed that: B is the subject of the attack, the attacker; A is just playing by the rules with no deliberate commitment; C is the object of the attack, the victim. So, we define:

- $\mathcal{B}_2 = \{B\}$
- $\mathcal{U}_2 = \{A\}$
- $\mathcal{G}_2 = \{C\}$

It follows that $\mathsf{A}(T_2, \mathcal{B}_2, \mathcal{U}_2, \mathcal{G}_2)$ holds.

V. Beyond Protocol Attacks

Before going beyond protocol attacks, we provide a classical formal definition of protocol vulnerability.

Definition 5 (Vulnerability) *A protocol Π is vulnerable to an attack A that is mounted by the principals in $\mathcal{B}$ exploiting those in $\mathcal{U}$ against those in $\mathcal{G}$ if there exists a protocol trace T that features A mounted by $\mathcal{B}$ exploiting $\mathcal{U}$ against $\mathcal{G}$ (Figure 5).*

Definition 5 is formalised in Figure 5, where a suitable predicate representing vulnerability is introduced as a function of the protocol, the attack and the principals' behaviours. Building on top of this definition we will characterise the subtler notion of retaliation.

$$\mathsf{Vulnerability}(\Pi, \mathsf{A}, \mathcal{B}, \mathcal{U}, \mathcal{G}) \equiv \exists\ T.\ \ T \in \Pi\ \wedge\ \mathsf{A}(T, \mathcal{B}, \mathcal{U}, \mathcal{G})$$

Figure. 5: Defining protocol vulnerability formally

A. Retaliation

What is the essence of retaliation? Should a principal cheat, he can be cheated back. It is therefore not obvious whether the principal will choose to cheat. A positive decision requires the absence of unbearable hazards. Clearly, retaliation is meaningful if hitting back is a meaningful property in the context of the given protocol. As the bad principals are protocol participants, namely insiders, we can assume that they want to reap the benefits of the protocol (such as authentication), plus any additional benefits they may obtain by misbehaving. These latter benefits should be balanced with the threats of being hit back. Designing a protocol so as to increase those threats will simply produce a stronger protocol.

Definition 6 (Retaliation) *A protocol* Π *allows* retaliation of an attack A that is mounted by the principals in $\mathcal{B}$ exploiting those in $\mathcal{U}$ against those in $\mathcal{G}$ *if, for every protocol trace that features* A *mounted by* $\mathcal{B}$ *exploiting* $\mathcal{U}$ *against* $\mathcal{G}$*, there exists an extension of the trace featuring* A *mounted by some* $\mathcal{B}'$ *exploiting some* $\mathcal{U}'$ *against some* $\mathcal{G}'$*. The principals in* $\mathcal{B}$ *change their role in the extended trace; vice versa, those in* $\mathcal{B}'$ *did not play the same role in the original trace. If* $\mathcal{B}' = \mathcal{G}$ *and* $\mathcal{B} = \mathcal{G}'$*, then* Π *allows* direct retaliation*, else* Π *allows* indirect retaliation.

Clearly, direct retaliation is the most intuitive form of retaliation, which sees the good and the bad principals exactly switch their roles. However, our examples have shown that more articulated forms of the property, such as indirect retaliation, are possible. Definition 6 is formalised in Figure 6, where a suitable predicate representing retaliation is introduced as function of the protocol, the attack and the principals' behaviours. The intuition is that each time there is an attack, some additional event may take place to retaliate, that is to attack the initial attackers. This typically involves some principals' changing their social behaviour. The formal definition in the figure confirms the change of roles: those who are now bad, the $\mathcal{B}'$, are a subset of those who were either ugly or good; those who were bad, the $\mathcal{B}$, are a subset of those who are currently either ugly or good.

$$
\begin{array}{l}
\mathsf{Retaliation}(\Pi, \mathsf{A}, \mathcal{B}, \mathcal{U}, \mathcal{G}) \equiv \\
\quad \forall\ T.\ \ T \in \Pi\ \wedge\ \mathsf{A}(T, \mathcal{B}, \mathcal{U}, \mathcal{G})\ \rightarrow \\
\qquad (\exists\ T', \mathcal{B}', \mathcal{U}', \mathcal{G}'.\ \ T' \in \Pi\ \wedge\ T \sqsubseteq T'\ \wedge \\
\qquad\qquad \mathcal{B}' \subseteq \mathcal{U} \cup \mathcal{G}\ \wedge\ \mathcal{B} \subseteq \mathcal{U}' \cup \mathcal{G}'\ \wedge \\
\qquad\qquad \mathsf{A}(T', \mathcal{B}', \mathcal{U}', \mathcal{G}'))
\end{array}
$$

Figure. 6: Defining retaliation formally

B. Suspicion and Detection

In the previous section we introduced the definitions of protocol vulnerability and retaliation. These were given in terms of a global view of the traces of events, a god-centric perspective. Equivalent principal-centric versions are of little significance because an attack is by its definition undetectable by its target principal.

However, a principal-centric perspective is possible if we envisage some empirical control event that principals can perform outside the protocol, which we call *out-of-band challenge*. The principals can easily use this method to check whether something fishy happened during the protocol.

The protocol responder can use the out-of-band challenge to raise his *suspicion* that something went wrong. Precisely, suspicion means that a good principal suspects that an attack was attempted, but has no clue on the possible attacker. In our example protocol, this can be achieved by a suitable message, as in Figure 7. Principal B is attempting a dull money

$$B \rightarrow A\text{: } \{\!|Na, Nb, \text{“Transfer £1 from } B\text{'s account to } B\text{'s”}|\!\}_{Ka}$$

Figure. 7: B's challenge for suspicion

transfer either within his own account or between two of his accounts. Notice that the amount is meaningless here — it may be 0 or another irrelevant value. Principal B can verify from his bank statement if the transfer went through. If this is affirmative, B gets a confirmation that A acknowledges the pair Na, Nb with him. Otherwise, B learns that his session with A was somewhat compromised by someone, exactly because A does not acknowledge the pair of nonces.

The challenge for suspicion can be made stronger, indeed becoming a challenge for *detection*. In our example protocol, this can be achieved by a suitable set of messages, as in Figure 8. Principal B is again attempting a dull money

$$\forall X.\quad B \rightarrow A :$$
$$\{\!|Na, Nb, \text{“Transfer £1 from } X\text{'s account to } B\text{'s”}|\!\}_{Ka}$$

Figure. 8: B's challenge for detection

transfer from any account holder onto his own. Principal B can verify from his bank statement for which principal X his attempt went through. This means that A associated the pair Na, Nb to X rather than to B. In consequence, B detects that X acted as a bad principal between A and B: the protocol admits a trace modelling this social behaviour.

After detection, B has sufficient evidence against the attacker, so he can draw a balance between two alternatives: either sue the attacker or retaliate against him.

VI. Towards Formal Verification under the $\mathcal{BUG}$ Threat Model

Classical properties such as authentication have been vastly analysed. Can we formally analyse properties such as retaliation? From a theoretical standpoint there is not a big differ-

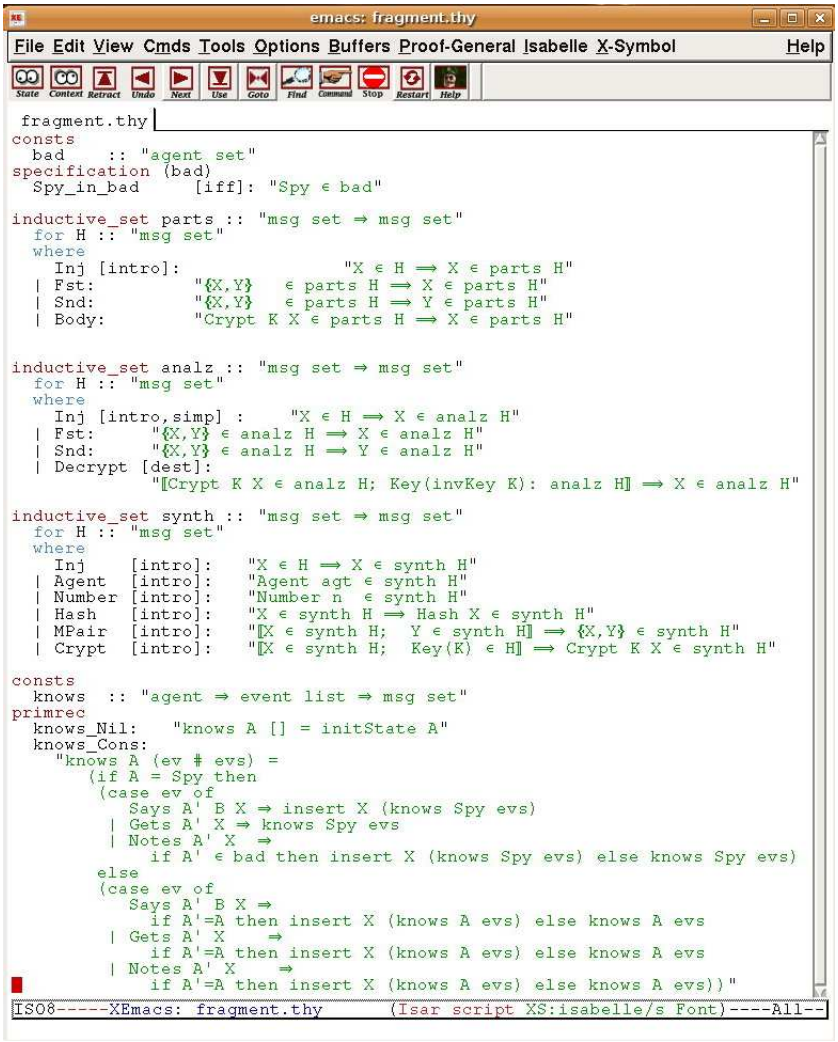

Figure. 9: Defining the main functions for the Inductive Method in Isabelle

ence. We have casted our properties as properties of traces because almost all research in tool-supported security verification is based on defining the protocol goals as properties of traces [9, 12–14, 17, 19, 20] or fragments thereof [8, 21].

The key observation is that the emphasis in the traditional work on security verification was on finding attacks or showing that no attack existed. This was reflected on formal models by the nature of the checked properties, which were essentially of existential nature: is there a trace T in the protocol Π such that A holds on T? Here, T, Π, and A can be complicated at will. Indeed, A as a formally defined property can be extremely complicated, for instance including arithmetical constraints on the number of events and arbitrarily many quantifiers. Theorem-proving fellows wished to prove that no such trace existed, while model-checking fans longed for a witness of its existence.

Our properties are much more complex, as they feature at least two quantifiers over a single trace, and we may also expect quantifier alternation. Lifting a theory of authentication to our properties appears to be reasonably simple: a formal account is already available using the method of soft constraint programming [3] with pen and paper. Lifting the automatic tool support remains a real challenge. We have already got to grasps with this challenge by experimenting with Paulson's Inductive Method [18] of protocol verification. Our initial findings are published here for the first time. The Inductive Method is developed for classical analyses in the Dolev-Yao threat model, but our experiments support the claim that it is realistically scalable to the $\mathcal{BUG}$ threat model. The Inductive Method is thoroughly supported by the generic proof assistant Isabelle [16], which can be obtained from the Internet [23] under the Open Source Software BSD licence.

A. *Outline of the Inductive Method*

There is only room here for a brief introduction to the method — the complete presentation is elsewhere [2, 18]. Figure 9 shows file fragment.thy opened by the graphical interface to

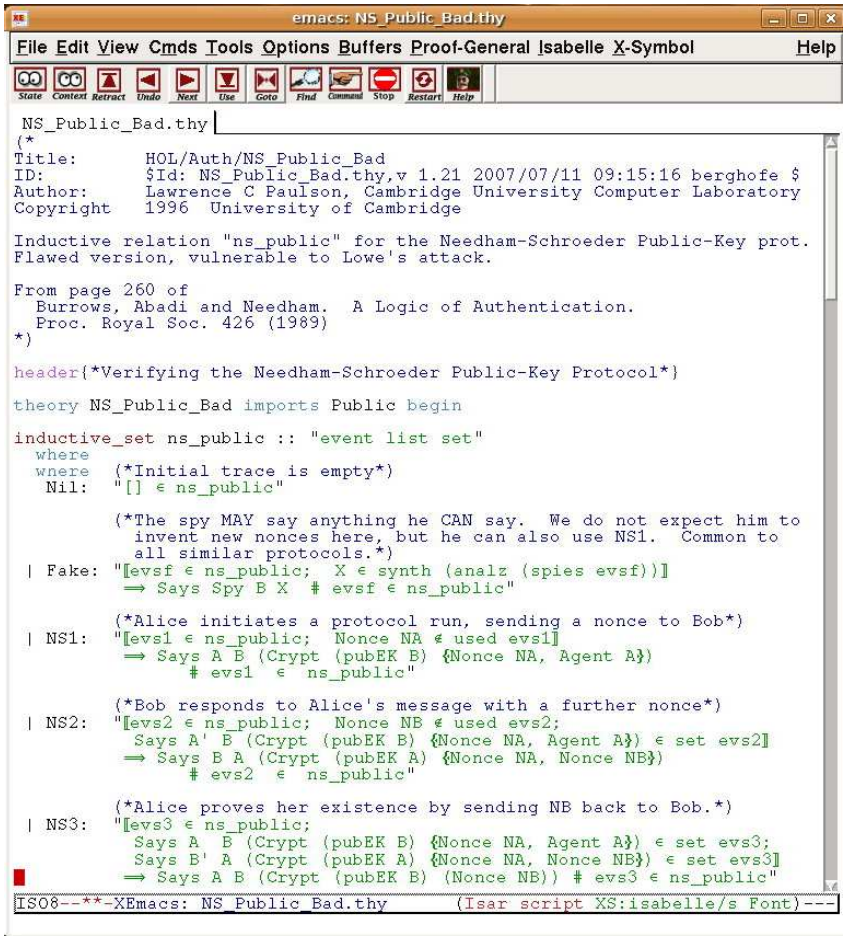

Figure. 10: Inductive model of the public-key Needham-Schroeder protocol

Isabelle [24]. The file sums up the definitions of a few main functions.

There exists an unlimited population of principals who are entitled to initiate at will an unlimited number of sessions of the given security protocol. Among the principals is the *spy*, who monitors the entire network traffic and in consequence knows who sends and who receives which messages. This feature indicates that the method was conceived under Dolev-Yao's threat model. An unspecified set of *bad* principals have colluded with the spy by revealing their long-term secrets. The spy is herself bad, as it can be seen in Figure 9. However, she is the only network principal who can send arbitrary messages built from components intercepted from the network traffic. Interception is modelled by the function knows, and creation of fake messages by a conjunct use of the functions analz and synth. All are described below.

The network traffic develops according to the events performed by the principals while they are executing the given protocol. Typical events are to send or to receive a message. A history of the network is represented by a *trace*, the list of events occurred throughout that history. The set of all possible traces is the formal model for the given protocol, and is defined inductively by specific rules drawn from the protocol. For example, if the protocol prescribes that B sends A a message m' upon reception of m, then the model features a rule that may extend a generic trace by the event Says $B\ A\ m'$ each time the trace contains the event Gets $B\ m$. In other words, that reception event is a precondition and that sending event is a postcondition of the rule. Therefore, the events occur via the firing of the inductive rules. But, as induction prescribes, no rule is forced to fire, so no event is forced to occur.

The binary function knows, defined by primitive recursion in Figure 9, formalises the knowledge that principals derive from observing a trace [2]. So, knows $A\ evs$ is the set of messages that principal A either sends or receives on trace evs. Should A be the spy, the set would include all messages that anyone ever sends or receives on the trace. Figure 9 also shows that the unary function parts extracts all components (portions of clear-text messages and bodies of cipher-texts) from a set of messages; analz is the same but only opens those cipher-texts whose encrypting key is available. This means that it is assumed that no cryptanalysis is possible, namely that encryption is totally reliable. In consequence,

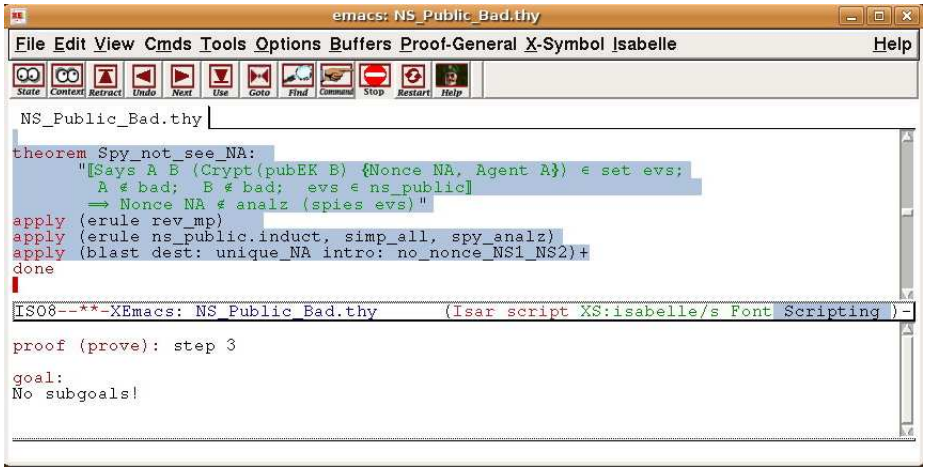

Figure. 11: Guarantee of confidentiality of the initiator's nonce

confidentiality of a message component m in a trace evs can be expressed as

$$m \notin \mathsf{analz}(\mathsf{knows}\,\mathsf{Spy}\ evs).$$

The function synth, also defined by induction in Figure 9, is crucial. It expresses the spy's illegal activity in building up messages at will. It can be seen that the spy can synthesise any agent name or number (timestamp), and hash available messages. She can also concatenate messages into longer ones, and build ciphertexts using available keys. Therefore, the set

$$\mathsf{synth}(\mathsf{analz}(\mathsf{knows}\,\mathsf{Spy}\ evs))$$

expresses all messages that the spy can synthesise from the analysis of the network traffic over trace evs.
Each kind of cryptographic key has its own syntax. For our examples below, it must be mentioned that the public keys are denoted by function pubK. Moving on to the actual formal guarantees, they come in the form of theorems that hold of the protocol model. Precisely, each theorem is expressed over a generic trace and hence holds in general. A proof is conducted by structural induction on the length of the trace, resulting in a number of long subgoals that can span several pages. It is here, where the long and often tedious proof entanglements contrast human scrupulousness, that the proof assistant Isabelle comes into help by solving the simple cases automatically.

B. Analysis of Needham-Schroeder under Dolev-Yao

Paulson analysed the public-key Needham-Schroeder protocol under Dolev-Yao's threat model years ago using the Inductive Method. The protocol model is quoted in Figure 10. It can be seen that the full specification comes with file NS_Public_Bad.thy and is defined as the Isabelle theory NS_Public_Bad. This is built up by extending theory Public, which defines all functions for public-key protocols. The actual protocol model, constant ns_public, is declared as a set of traces and defined inductively by five rules. Rule Nil sets the base of the induction stating that the empty trace belongs to the model. Rule Fake models the spy's activity: given a trace *evsf* in the model, its extension (# is the list append operator) with the event whereby the spy sends some agent B one of her fake messages X is still a trace of the protocol model. The inductive layout is clear. Notice that the message X is derived from the set of fakes described in the previous section. The remaining rules model the steps of the protocol, one by one.
Rule NS1 has the only premise that A uses a fresh nonce. Freshness on a trace is modelled via the function used, whose intuitive definition is omitted from this presentation. Rule NS2 also relies on the assumption that B received an instance of the first message on the given trace *evs2*. Notice that Says $A'\,B\,X$ is an old syntax for Gets $B\,X$. Rule NS3 admits that A concludes the protocol with B only if A initiated it with B and got a matching instance of the second message.
Paulson's analysis of this protocol confirmed Lowe's attack. Here, we are interested in a guarantee expressing confidentiality of A's nonce, not of B's — the reasons being clarified in the next section. It is given in Figure 11 in the same form as it is released with Isabelle [23]. The blue shade indicates what Isabelle has processed until the moment when the picture is taken. The human analyser can interact with the proof assistant using the window buttons, for example executing an extra step by pressing the "Next" button. This theorem insists that A initiates the protocol with B using a nonce *NA*. It requires that both of them are not bad, and concludes that the nonce remains confidential (spies evs is the old syntax for knows Spy evs).
The proof of this theorem is simpler to execute interactively than to describe on paper. Each proof command is introduced by the keyword apply. The first command brings the first premise, namely the Says event, into the inductive formula. The second one applies induction, then simplifies all

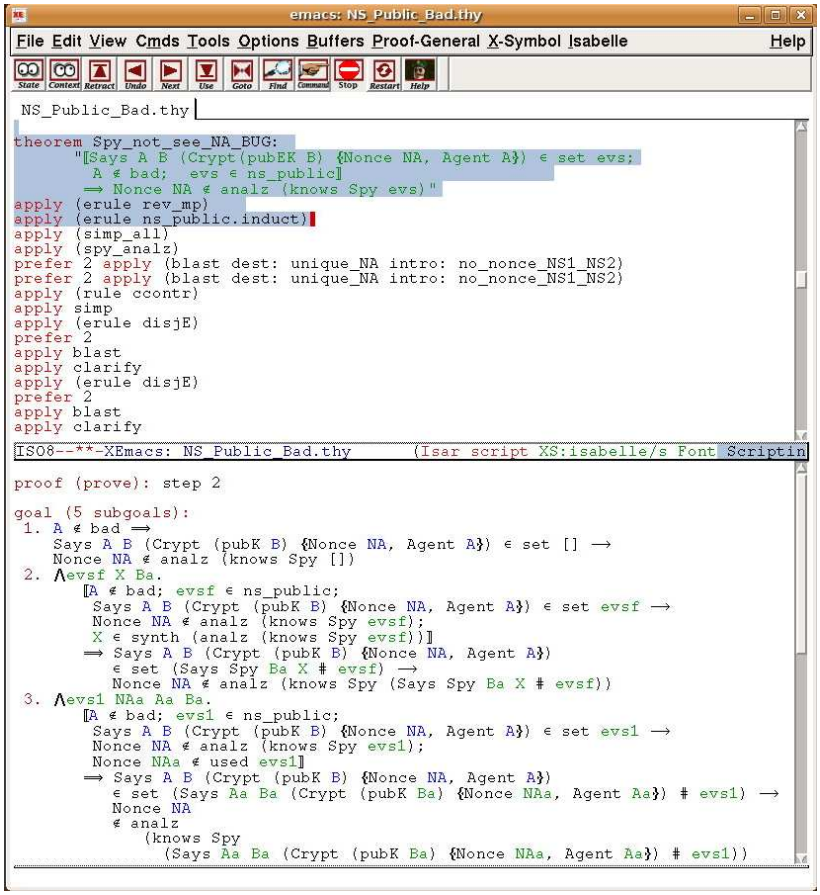

Figure. 12: Attempt to prove confidentiality of the initiator's nonce in $\mathcal{BUG}$: level 2

cases, and solves the subgoal corresponding to rule Fake by a standard method spy_analz. The final command applies the classical reasoner to all remaining subgoals with the only aid of two lemmas proved before.

C. Analysis of Needham-Schroeder under $\mathcal{BUG}$

Can we use the Inductive Method and its Isabelle support under the $\mathcal{BUG}$ threat model? Can we tailor the strict formalisation of the spy to a broader and more realistic threat? No, it appears that this cannot be done straightforwardly, that is, without a complete redefinition of the agents' datatype and related functions. However, we had a number of insights in the attempt to keep the standard formalisation of the spy. One that seems the most balanced between simplicity and expressiveness merely concentrates on the theorem statements.

As we observed above (§II), an attack can be retaliated when it causes more than one violation of the protocol policy: one that is directly useful to the attacker to mount the attack, and the others that are usually neglected. We noticed that with the public-key Needham-Schroeder protocol, the attacker's activity indeliberately revealed nonce *NA* to B, when in fact A only intended to share that nonce with the attacker. Having observed that it is feasible that B sooner or later intends to exploit his knowledge of *NA* against other principals, the direction for tomorrow's formal protocol verification becomes clear: it is necessary to investigate confidentiality of all potentially sensitive components in the $\mathcal{BUG}$ threat model and for all protocols.

With our example protocol, this direction requires establishing formally whether *NA* remains unknown to B, who can be the attacker himself. Strictly speaking, this is daunting due to the formalisation of the spy, who cannot hide behind two agents at the same time. However, here we advance a heuristic that investigates what happens to the confidentiality theorems if we relax the assumptions that the involved agents are not bad: it can reveal novel vulnerabilities of sensitive message components. An attempt to prove the theorem presented in the previous section omitting the assumption of a bad B is in Figure 12. It can be seen that subgoal 1 corresponds to case Nil, subgoal 2 to case Fake, subgoal 3 to case NS1 and so on — there are five subgoals although the window only covers three. Simplification has not been done at the moment: the light blue shading confirms that it is the next step that can be taken.

Let us concentrate on subgoal 3. The inductive formula is available as the third premise in the preconditions. The postcondition expresses the thesis for the trace *evs1* extended

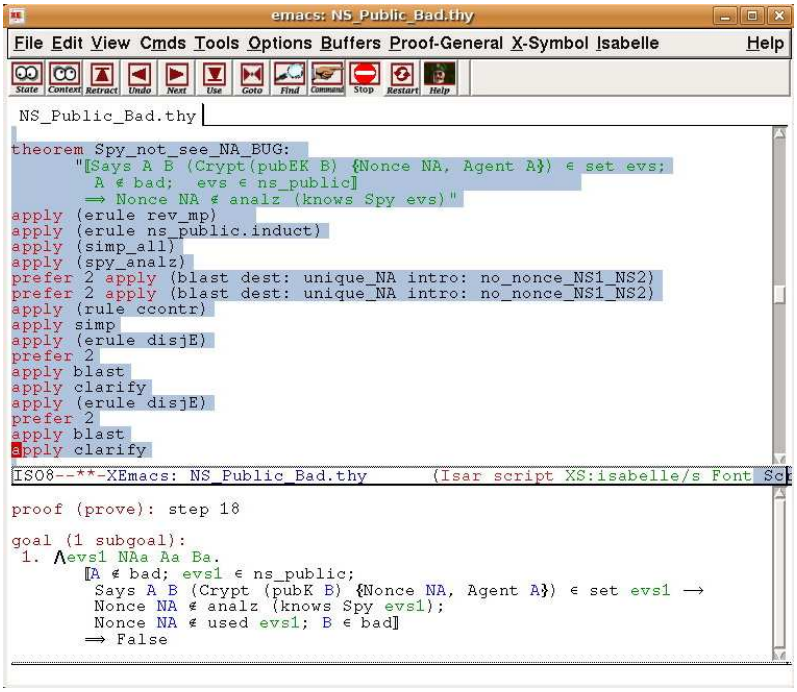

Figure. 13: Attempt to prove confidentiality of the initiator's nonce in $\mathcal{BUG}$: level 18

with the new event introduced by rule NS1. If the event Says $A\ B$ ... was already in trace *evs1*, then the subgoal terminates via an appeal to the inductive formula. Otherwise, that event is exactly the one that rule NS1 introduces, and so it must be the case that A equals *Aa*, B equals *Ba* and so on. If we continue the proof attempt interactively as in Figure 13, Isabelle routinely solves most subgoals. For example, command prefer moves the numbered subgoal to the first slot, to which each standard command applies by default. Suitable applications of this command leave us at level 6 with only the subgoal corresponding to rule NS1 to be solved. We solve its simple subcases interactively by a few intelligible commands. For example, if B is not bad, then he certainly is not the spy (because the set bad includes the spy), and the thesis can be reached. What happens if B were bad, that is, potentially the spy? Figure 13, confirms that Isabelle leaves us with the very subgoal denouncing that B is bad. It is clear that it cannot be terminated because no premises lead to contradiction. This interactive and long proof was described here to facilitate the reader's intuition. However, one appeal to the auto method at level 6 would have led us to the same state in just one step.

It is no secret that the experiments reported here were conducted only after we had the insights described in the early sections of this paper. Their significance remains unaltered: if we analyse all protocols under the $\mathcal{BUG}$ threat model, we can find violations that can lead to retaliation attacks. In particular, the Inductive Method used on the public-key Needham-Schroeder protocol as we suggested above denounces mechanically B's indeliberate discovery of A's nonce.

This is the current state of the art. However, our last guarantee cannot be considered the most expressive formalisation of a retaliation attack: the spy has bad agents' private keys and hence can access anything that is encrypted with the corresponding public halves. So, there is vast potential for additional research here. In particular, it remains to be formalised that, should the spy hide behind B, she might need to forward A's nonce to another agent. This was the case with Lowe's attack.

VII. Conclusions

Our research is motivated by the novel settings in which security protocols are executed nowadays, significantly different from settings dating back to nearly three decades ago. Security protocols, whose use was typically appanage of 007s to protect their communications from the rest of the world during espionage missions, have now become accessible to a huge international community. The threat model has indeed changed. It is now perfectly realistic to even conceive that each principal may want to attack (whatever this means in a context) everyone else — on-line auctions in particular and e-commerce in general come as examples. Also non-repudiation protocols assume that everyone trusts no-one else.

The good principals were expected in the taxonomy, but the ugly principals perhaps not. The identification of this social behaviour brings forward another new concept: principals cannot and should not be constrained to be playing a single

social behaviour forever. Imposing such a constraint would limit formal analysis significantly in scope. More precisely, given a trace of events representing participation in a protocol, the social behaviours played by each principal can be easily identified, but they may vary in a different trace, such as an extension of the original trace. More simplistically, we could even see all principals as ugly, who turn out to behave as good or as bad according to specific circumstances.

This paper has formalised the notion of retaliation in the context of security protocols. If an attack is discovered, it is worth investigating whether it can be retaliated. If yes, risk analysis may lean towards keeping the protocol in use. This perspective advances on the long-established practice of going back to redesign soon after one attack. An attack signifies a flaw, not necessarily a complete failure. Also the notions of suspicion and detection appear to have never been spelled out explicitly. They are adequately supported by the new threat model. It seems fair to conclude that the path to a new, important niche of protocol verification has just been drawn.

A pen-and-paper formal analysis of retaliation is already available [3]. However, it is widely accepted that mechanical tool support is necessary to deal with proofs about security protocols, as their major difficulty often is their sheer length. This paper has extended a prior version [5] with the first experiments of tool-supported analysis of retaliation attacks. We have discussed the current state of the art, which balances expressiveness with simplicity. But a complete mechanisation would require the full implementation of the $\mathcal{BUG}$ threat model. That is where future research is targeted.

Acknowledgements

Stefano Bistarelli was partially supported by the Italian PRIN project "Vincoli e preferenze come formalismo unificante per l'analisi di sistemi informatici e la soluzione di problemi reali". Fabio Massacci was partially supported by the FIRB "Security" and IST-FET-IP "Sensoria" projects.

References

[1] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations (extended abstract). In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS)*, pages 220–230. ACM Press, 2003.

[2] G. Bella. *Formal Correctness of Security Protocols*. Information Security and Cryptography. Springer, 2007.

[3] G. Bella and S. Bistarelli. Soft constraint programming to analysing security protocols. *Journal of Theory and Practice of Logic Programming*, 4(5):1–28, 2004.

[4] G. Bella, S. Bistarelli, and F. Massacci. A protocol's life after attacks. In *Proc. of the 11th Security Protocols Workshop (SPW'03)*, LNCS 3364, pages 3–18. Springer, 2005.

[5] G. Bella, S. Bistarelli, and F. Massacci. Retaliation: Can we live with flaws? In M. Essaidi and J. Thomas, editors, *Proc. of the Nato Advanced Research Workshop on Information Security Assurance and Security*. IOS Press, 2005.

[6] M. Bellare and P. Rogaway. Provably Secure Session Key Distribution — the Three Party Case. In *Proc. of the 27th ACM SIGACT Symposium on Theory of Computing (STOC'95)*, pages 57–66. ACM Press, 1995.

[7] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Information Security and Cryptography. Springer, 2003.

[8] L. Carlucci Aiello and F. Massacci. Verifying security protocols as planning in logic programming. *Transactions on Computational Logic*, 2(4):542–580, 2001.

[9] E. M. Clarke, S. Jha, and W. Marrero. Verifying security protocols with brutus. *ACM Trans. Softw. Eng. Methodol.*, 9(4):443–487, 2000.

[10] D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.

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

[12] R. Kemmerer, C. Meadows, and J. Millen. Three system for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, 1994.

[13] G. Lowe. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Information Processing Letters*, 56(3):131–133, 1995.

[14] J. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murphi. In *Proc. of the 16th IEEE Symposium on Security and Privacy (SSP'97)*, pages 141–151. IEEE Press, 1997.

[15] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[16] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer, 2002. LNCS Tutorial 2283.

[17] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[18] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[19] F. R. and R. Gorrieri. The compositional security checker: A tool for the verification of information flow security properties. *IEEE Transactions on Software Engineering*, 23(9):550–571, 1997.

[20] S. Schneider. Security properties and CSP. In *Proc. of the 15th IEEE Symposium on Security and Privacy (SSP'96)*, pages 174–187. IEEE Press, 1996.

[21] D. Song. Athena: An automatic checker for security protocol analysis. In *Proc. of the 12th IEEE Computer Security Foundations Workshop (CSFW'99)*. IEEE Press, 1999.

[22] F. Thayer Fabrega, J. Herzog, and J. Guttman. Honest ideals on strand spaces. In *Proc. of the 11th IEEE Computer Security Foundations Workshop (CSFW'98)*. IEEE Press, 1998.

[23] URL. Isabelle download page. `http://www.cl.cam.ac.uk/Research/HVG/Isabelle/download.html`

[24] URL. Proof General: a generic interface for proof assistants. `http://proofgeneral.inf.ed.ac.uk`