Computers & Security (2005) **24**, 322—333

Computers & Security

www.elsevier.com/locate/cose

Information Assurance for security protocols

Giampaolo Bella[a,*], Stefano Bistarelli[b,c]

[a]*Università di Catania, Dipartimento di Matematica e Informatica, Viale A. Doria 6, I-95125 Catania, Italy*
[b]*CNR, Istituto di Informatica e Telematica, Via G. Moruzzi 1, I-56124 Pisa, Italy*
[c]*Università "D'Annunzio", Dipartimento di Scienze, Viale Pindaro 42, I-65127 Pescara, Italy*

Received 17 March 2004; revised 6 October 2004; accepted 25 October 2004

KEYWORDS
Formal analysis;
Constraint solving;
Soft constraints;
Cryptographic protocol;
Realism;
Accuracy;
Formalism;
Retaliation

Abstract Security protocols are used pervasively to protect distributed communications in the third Millennium. This motivates the need for a definition of Information Assurance for security protocols, which, to the best of our knowledge, is still missing. Such a definition is advanced in terms of the requirements that security protocols be analysed at the same time *realistically*, *accurately* and *formally*, notions that the existing literature only favours in separate contexts. The precise meanings of these terms are described by means of general considerations and concrete examples. The main goal of this paper is to draw attention to and raise concern on this novel but significant niche of computer security.

Overview

Computer Security began its course with the implementation of some rudimentary access-control features in Multics (Corbatò and Vyssotsky, 1965), that is in the late 1960s. At that time, security merely meant to regulate access to the sensitive resources of a single computer. Technology progressed fast, bringing a number of new problems forward. With the development of large-scale computer networks, the figure of a new expert became necessary, the *security architect*. We are not going to survey the history of computer security here, but a few observations are necessary to set the ground for our discussion on Information Assurance for security protocols.

Security architects are concerned with security problems that require solutions, and the existing technology where to derive those solutions from. Until the 1980s, their decisions were only taken on the basis of *informal* reasoning. In the best case, teams of architects used to brain-storm together for months or years in the hope to account for *all* possible strengths or weaknesses of a certain security measure. Only if ideas converged on the strengths would that measure be put in place.

* Corresponding author.
E-mail addresses: giamp@dmi.unict.it (G. Bella), stefano.bistarelli@iit.cnr.it, bista@sci.unich.it (S. Bistarelli).

0167-4048/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2004.10.004

But convergence of ideas very rarely derives from unanimous agreement. Indeed, enforcing certain measures in the real world may signify weakening others, which means that what was expected as a solution turns out to be in fact a tradeoff. An enormous number of examples can be made to support the claim that even after the security measures are enforced, acceptance of some "level of risk" is often still required. We only sketch three groups of examples here, which can be skipped for skim-reading: security installations, protocol design principles, and semi-trusted code.

"Security" installations

There exists empirical evidence that protecting a WWW server by allowing only SSH connections has decreased the number of external break-ins. Tolerable drawbacks are the cost of the new software, the time of installing SSH clients on all remote machines, and the efforts to create people's mentality in using the new software. But since the number of attacks has at times been too far from zero yet, many academic sites have decided to even restrict SSH access to principals within the local domain (Cheswick et al., 2004).

Analogous caution must be taken with security protocols. A security protocol is a set of prescribed message exchanges between principals over an insecure network. Each protocol aims at achieving specific security goals, such as confidentiality and authentication. Paulson has proved formally that the BULL recursive protocol meets its goals if encryption is *perfect*, that is if cryptanalysis is impossible (Paulson, 1997). This may have favoured the use of the protocol for some time, but Ryan and Schneider (1998) soon found an attack on the protocol if encryption is bit-wise exclusive-or. At installation time, the security architect would hopefully have the option of choosing a stronger encryption function, which in any case would be far from provably perfect. No boolean statement about the strengths of the installed software would be conceivable.

Protocol design principles

Anderson and Needham (1995) propose a number of influential principles for designing security protocols. Often has it been remarked that those principles are not meant to totally *ensure* the protocol goals, but that it merely is *prudent* to conform to them, as the title of an influential paper confirms (Abadi and Needham, 1996). In other words, no design principle should be taken as biblic. For example, one of the most popular principles states that each protocol message should be explicit about its meaning, that is to say that nothing should be taken for granted. But Syverson (1996) warns us that this principle too has limitations. Protocols such as Bellovin and Meritt's (1993) Encrypted Key Exchange (EKE) do not conform to it, and indeed meet their goals by not conforming to it. The contents of any encrypted message of the EKE protocol can equally be either a key or a nonce, so that not even a legitimate principal gets evidence about which session a cipher-text belongs to until he terminates the protocol on faith.

Semi-trusted code

Code downloaded from the Internet should not be entirely trusted. A number of viruses spread through the world as email attachments for example. A possible strategy to control the behaviour of the code and limit its potential damages is to execute it in a *sandbox*. The principal is granted fine-grained access control for each potentially dangerous call, so he can take informed decisions. But experience shows that to run code in a sandbox may result lengthy and tedious, so that the principal ends up granting too many permissions to the code anyway. Even more crucial is the problem of executing code from different origins, hence with different levels of trust, in the same runtime. Some levels may have to be lowered or decreased. Techniques based on inspection of the execution stack or on execution history are currently being developed (Abadi and Fournet, 2001).

What now

The third Millennium seems to be favouring an increasing awareness about the issues sketched above, which support the claim that "Security is not a simple boolean predicate" (Anderson, 1993). Worse yet, complete security still seems out of reach or perhaps impossible. Modern security architects normally build their arguments on top of such postulates. As from the late 1980s they have also started to take into account insights derived from *formal* analyses, which were just beginning to be conceived, as opposed to the merely *informal* analyses they relied on before. Skepticism has gone up and down throughout the years, but the contribution of formal reasoning in general is nowadays unquestionable. This can be seen throughout nearly all niches of computer security, ranging from policies (Cholvy and Cuppens, 1997; Halpern and Weissman, 2003) to protocols (Fábrega et al., 1998; Gollmann, 2000a).

With this paper, we intend to develop a notion of Information Assurance (''IA'' in the following) for security protocols. These are pre-established cryptographic message exchanges between remote principals wishing to communicate securely over an insecure network. We advance a tentative notion of IA for security protocols, which the literature appears to be lacking. In doing so, our aim is to provide strong assurances about the information that principals derive from executing the protocols, such as confidentiality of messages or authentication of remote principals. On these bases, we advance the following definition.

IA for security protocols

Providing Information Assurance for security protocols is the process of analysing the protocols under the following three provisos:

1. Realism
2. Accuracy
3. Formalism

The three provisos are easy to motivate. The present ratio cost/technology allows every Internet principal to be a potential attacker; each attacker even is in a condition to exploit attacks mounted by other principals for his own sake. Hence, protocol analyses must be *realistic* and account for a distributed model of attacker. The general impossibility of making strong boolean claims in the field of security, which we have exemplified above, convinces that protocol analyses must also be *accurate* and capture some notion of ''level'' of the goals achieved by a protocol. Furthermore, security architects require formal assurance that a protocol meets its goals in the real world, which is intrinsically difficult because formal models somewhat idealise reality. Hence, a *formalism* that embeds the first two provisos is needed.

Our definition is purposely abstract so as to suscitate different views among different researchers, and hopefully give occasion to the international debate on a niche of computer security that at present deserves consideration. Our own views of the three provisos of realism, accuracy, and formalism are given in the following sections. The last section draws some conclusions.

Realism

During the last two decades, Dolev and Yao's (1983) famous paper has substantially influenced security protocol analyses, regardless of whether they were informal or formal. Dolev and Yao's contribution is essential evidence that collusion of a number of principals to subvert a protocol is equivalent to the hostility of a single, powerful attacker. ''Powerful'' means the ability to monitor the entire network traffic, to break down messages by the conventional operations of splitting and decryption, to build up new messages by the conventional operations of concatenation and encryption, and to engage in unlimited protocol sessions. The only constraint is that the attacker cannot mount brute-force cryptanalysis attacks, hence he can only rely on the keys that become available by any interleaving of the four allowed operations. Their model makes no account for the different security conditions that different areas of the network may have.

These simplifications to the threat model characterise essentially all approaches to analysing security protocols formally. What has never been considered in this context is a threat model where each principal is malicious so as to act for his own profit in every way Dolev and Yao's attacker would. In other words, no principal colludes with anyone else, but they are all separate attackers, as we can see in the real world. (The model is inspired to recent fair-exchange protocols; Shmatikov and Mitchell, 2000.) As a first requirement of IA for security protocols, we propose the use of *realistic* threat models in the sense described above.

Here is one example of a previously unknown insight that our threat model allows us to get. Fig. 1 presents the simple protocol due to Needham and Schroeder (1978), a milestone in the field. This is the variant that relies on asymmetric cryptography, that is each principal owns a private key and a corresponding public key. A *nonce* is a ''number that is used only once'' (Needham and Schroeder, 1978). The protocol assumes that principals can invent *truly-random* nonces, so that, given a nonce N invented by a principal P, the probability that principals other than P guess N is negligible.

The first step sees an *initiator* A initiate the protocol with a *responder* B. Principal A invents a nonce Na and encrypts it along with her identity under B's public key. Upon reception of that

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

Figure 1 The asymmetric Needham—Schroeder protocol.

message, B decrypts it and extracts A's nonce. Then, he invents a nonce Nb and encrypts it along with Na under A's public key. When A receives message 2, she extracts Nb and sends it back to B, encrypted under his public key.

The goal of the protocol is authentication: at completion of a protocol session initiated by A with B, A should get evidence to have communicated with B and, likewise, B should get evidence to have communicated with A. We emphasise how confidentiality of the nonces is here used to achieve authentication. Indeed, upon reception of Na inside message 2, A would conclude that she is interacting with B, the only principal who could retrieve Na from message 1, since Na is a truly-random nonce and encryption is perfect. In the same fashion, when B receives Nb inside message 3, he would conclude that A was at the other end of the network because Nb must have been obtained from message 2, and no one but A could perform this operation. However, Lowe (1995) shows that this protocol is flawed within Dolev and Yao's threat model by exhibiting the attack we present in Fig. 2.

Lowe assumes that principal C impersonates Dolev and Yao's attacker. Notice that C could be a registered principal of the network, so no one could suspect his tampering. Since A initiates with C, she encrypts her nonce and her identity under C's public key. Once these data are obtained, C initiates another session (indicated by the primes) with another principal B, quoting A's data rather than his own. From this message, B deduces that A is trying to communicate with him. Therefore, B replies to A, quoting her nonce and his own, Nb. Since the entire network is under C's control, C intercepts this message before it is delivered to A but cannot decrypt it because encryption is perfect. So, C forwards it to A. The message is of the form that A was expecting, hence A extracts Nb and sends it to the principal with whom she had initiated the first session, C. This hinders the confidentiality of Nb, so C can use it to complete the session with B by issuing message 3′, which is of the form that B was expecting.

$$\begin{array}{lll}
1. & A \to C & : \{\!|Na, A|\!\}_{Kc} \\
1'. & C \to B & : \{\!|Na, A|\!\}_{Kb} \\
2'. & B \to A & : \{\!|Na, Nb|\!\}_{Ka} \\
2. & C \to A & : \{\!|Na, Nb|\!\}_{Ka} \\
3. & A \to C & : \{\!|Nb|\!\}_{Kc} \\
3'. & C \to B & : \{\!|Nb|\!\}_{Kb}
\end{array}$$

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

As a result, B believes to have communicated with A, while A was in fact communicating with C. In other words, C impersonates A with B: the protocol has ultimately failed to achieve authentication because it has failed to keep Nb confidential. Lowe observes that this may have drastic consequences, such as the following. If B is a bank, C can steal money from A's account as shown in Fig. 3. The bank B would honour the request believing it came from the account holder A.

This terminates Lowe's study, which is sound within Dolev and Yao's threat model. We studied the protocol within our own threat model and highlighted an *indeliberate attack* (Bella and Bistarelli, 2002) whereby B learns A's nonce Na, which is meant for use with C, not with B. "Indeliberate attack" means that B obtains the nonce without any deliberate action—still, it counts as an attack because the nonce is not meant to be known to B. This indeliberate attack may have another drastic consequence. If A is a bank, B can steal money from C's account as shown in Fig. 4. The bank A would honour the request believing it came from the account holder C.

The threat model given here allows *retaliation* of B against C, a novel concept that has been recently developed Bella et al. (2003). In short, because C initially steals from A's account and frauds B, principal B can then retaliate by stealing from C's account and frauding A. This form of retaliation whereby a principal who is attacked in turn exploits this attack to attack back appears perfectly realistic in the present Internet setting where any principal may have sufficient knowledge and instruments.

We can meet the first requirement, that of realism, by adopting the realistic threat model described here.

Accuracy

We concentrate on the goals of confidentiality and authentication. Confidentiality of a message means that the message remains undisclosed to those not intended to learn it (Dolev and Yao, 1983; Neuman and Ts'o, 1996). Authentication of a principal means that we truly are communicating

$$C \to B : \ \{\!|Na, Nb, \text{“Transfer £1000 from } A\text{'s account to } C\text{'s”}|\!\}_{Kb}$$

Figure 3 Lowe's fraud to bank B.

$B \to A: \quad \{\!|Na, Nb, \text{“Transfer } \pounds 1000 \text{ from } C\text{'s account to } B\text{'s”}|\!\}_{Ka}$

Figure 4 Our fraud to bank *A*.

with that principal (Gollmann, 2000b, 2001). We observe that neither of these two properties is boolean in the real world: only certain *levels* of confidentiality or authentication are achieved in practice. Levels are known as a means to conduct reasoning, as is the case of Abadi's *types* (Abadi, 1999), which means they are in fact meta-levels. In contrast, we are introducing object-levels.

We now tackle the second requirement of our notion of IA for security protocols, the requirement of *accuracy*. (The treatment is informal here, its formal version is deferred— see section 'Formalism'.)

Confidentiality

To set about this goal, let us consider *session keys* for example. One or two of them are invented per each session (that is, execution) of a protocol, the reason being that each key is only meant to be used for a short lifetime. Clearly one such key is less sensitive information than a principal's password or PIN number, whose lifetimes usually are considerably long. That is to say that the acceptable level of confidentiality on a session key is lower than the acceptable level of confidentiality on a password. Most protocols follow this proviso. While session keys may be sent within message bodies on repeated occasions, passwords are only used as encryption keys typically once or twice. Indeed, sending a secret over the network exposes it to risks. The more the secret is used to form messages, the higher the risks that an attacker tampers with it—the longer the secret is on the network, the higher the risks it runs. Also, we may have chains of session keys, each encrypting the next one. The confidentiality levels of the keys decrease along the chain, as confidentiality of each key rests on confidentiality of all preceding keys. In practice, one would prefer to rely on cipher-texts sealed under the first key in the chain rather than under the last one.

Such a dependency chain is a didactic example, but we remark that *all* protocols exchange components some of which are more sensitive than others. Kerberos (Bella and Paulson, 1998) and the Yahalom protocol (Paulson, 2001), for example, have dependency chains of length two. An example of a lengthy dependency chain derives from the Cardholder Registration phase of the SET protocol, a large e-commerce protocol by Visa and Mastercard (Bella et al., 2003). Fig. 5 shows a simplified version of the last two messages.

The peers are a cardholder *C* and a certification authority *CA*. In message 5, *C* sends *CA* a digital envelope formed by the concatenation of two cipher-texts.

The first is sealed under *CA*'s public encryption key, pubEK *CA*, which is known to all, and contains the following items: a session key KC_3, the *PAN* (Personal Account Number) that is written on *C*'s credit card, and *Csecret*, which is a secret number invented by *C*. The second component is sealed under session key KC_3 and contains the cardholder's identity, another session key KC_2 and many other components, which we are hiding behind *rest* as they are irrelevant to our discussion.

Upon receipt of message 5, the *CA* decrypts the first component by its private encryption key priEK *CA*, extracts KC_3 and uses it to decrypt the second component. So, the *CA* gets KC_2 and uses it to encrypt a secret number it has just invented, *CASecret*. The so formed cipher-text is sent back to *C* in message 6. When the protocol session terminates, the cardholder *C* uses *CA*'s secret number *CASecret* to compute another secret number *PANSecret* to be used in the subsequent phases of the protocol.

The protocol presents the following dependency chain. Confidentiality of *PANSecret* depends on confidentiality of *CASecret*, which depends on confidentiality of KC_2, which depends on confidentiality of KC_3, which depends on confidentiality of priEK *CA*. The level of confidentiality the protocol achieves on *PANSecret* is certainly lower than that achieved on priEK *CA*. This example demonstrates that a confidentiality argument based on merely boolean statements of the form "KC_2 remains confidential and so does *PANSecret*" would not be sufficiently accurate. Clearly, proving confidentiality of the message at the end of the chain implies that confidentiality of the message at the beginning of the chain is achieved too. But the opposite implication does not hold. For example, we could imagine that confidentiality levels were linearly ordered as I, II, III, IV, ..., IX, X, with I being the maximum level. More appropriate claims should have the form "priEK *CA* is I-confidential, and *PANSecret* is V-confidential" for example, in case the security level only decreases by one at each node along the dependency chain.

$5.\, C \to CA: \{\!|KC_3, PAN, CSecret|\!\}_{\text{pubEK } CA}, \{\!|C, KC_2, rest|\!\}_{KC_3}$

$6.\, CA \to C: \{\!|C, CA, CASecret|\!\}_{KC_2}$

Figure 5 A simplified fragment of the SET protocol.

Once fixed criteria are established to manipulate the security levels, the protocol analyser could even compare the security level achieved by different protocols on sensitive message components. For example, the analyser could consider two different key-distribution protocols, such as Otway-Rees and Kerberos as they are presented by Burrows et al. (1989), and study what confidentiality levels they achieve on the session key. This would give security architects deeper insights than current analyses do. In our threat model formalised above, where each principal sees the entire network traffic, a secret runs risks of leaking that are proportional to the time the secret has been on the network. Therefore, one appropriate criterion to manipulate the security levels is to decrease the level associated to a message component every time that component is manipulated by any of the operations on messages (concatenation, splitting, encryption and decryption)—see section 'Formalism'.

A message that has been disclosed to the attacker can be seen as having the minimum confidentiality level. This inspires a strategy to compare confidentiality attacks. For example, leaking a message that was I-confidential reports a more significant attack than leaking a message that was V-confidential. To the best of our knowledge, such accuracy is missing to existing protocol analyses, but we find it an indispensable prerequisite of appropriate IA for security protocols.

Authentication

Analogous considerations apply to the authentication goal. Classifications of the different kinds of authentication exist (Gollmann, 1996; Lowe, 1997) but each of them should be studied in terms of the levels we introduce here.

Authentication (of whatever kind) of a principal is normally established by means of a message that expresses the principal's presence. The more the message is confidential, the stronger the achieved authentication. For example, sending Alice's identity on the network to Bob is the weakest form of authentication of Alice to Bob because message "Alice" is in fact public (all principals know each other's identities). Clearly, a much stronger authentication would Alice achieve by sending Bob a cipher-text sealed under Alice's private key, which has a high confidentiality level. And some intermediate level of authentication Alice would get by sealing a cipher-text under some session key. The existence of a dependency chain generalises these observations. If we consider the fragment of the SET protocol as shown in Fig. 5, principal C achieves a higher authentication level with CA by using KC_3 rather than by using $CASecret$.

It is desirable to also have a strategy to compare authentication attacks. A possible strategy is based on the following observation. If an attacker impersonates Alice with Bob, the significance of the attack is proportional to the confidentiality level of the message used by the attacker to impersonate Alice. For example, if the attacker gets hold of Alice's private signature key, he can mount a more significant authentication attack on Bob (and on any other principal) than if he gets hold of Alice's session key shared with Bob (in which case he could only cheat on that session).

Formalism

Formal analyses of security protocols facilitate a deep understanding of the protocols. A variety of approaches have been taken, ranging from *state enumeration* (Ryan and Schneider, 2000), to *provable security* (Bellare and Rogaway, 1995), to *induction* (Paulson, 1998). These efforts have led to the discovery of a number of protocol attacks (and of the corresponding patches), or to the formal establishment that certain goals are achieved.

However, we observe that the exhibition of an attack raises more interest among the security architects than the exhibition of a proof that a goal is met. While it is easy to verify that the former can take place, it is less easy to believe that a formal proof would still hold in the real world. Such skepticism is motivated by the idealised models within which any formal proof is conducted. Although it is virtually impossible to bridge the gap between a formal model and the real world, following our first requirement (see section 'Realism') certainly is an improvement. Reluctance towards formal proofs that goals are met also comes from the nature of the offered insights. These are firm boolean claims of the form "session key K is confidential" or "principal A authenticates principal B", while security architects rely on levels of those goals, whose importance was stated above by our second requirement (see section 'Accuracy'). We meet the two requirements by adopting the realistic threat model and the confidentiality/authentication levels described in the previous two sections.

Our third requirement towards IA for security protocols is therefore *formal* analysis, which, as remarked above, is severely limited on its own, but can be powered with the other two requirements. It was not obvious in the beginning what approach to protocol analysis could embed all three requirements, but Constraint Solving soon seemed to be an appropriate candidate (Bistarelli, 2004). We present here only the basics of our approach, which is at the same time realistic, accurate and formal. The complete description can be found elsewhere (Bella and Bistarelli, 2004). It should be remarked that ours is *one* possible approach that embeds all three features, but certainly others may be taken. We expect that some existing formal approaches can be extended to accommodate our first two requirements (see sections 'Realism' and 'Accuracy').

Basics of soft constraint programming

Informally speaking, given a set of variables $\mathcal{V}$ and a set of domain values $\mathcal{D}$, a *constraint* is a law that associates *n*-tuples of domain elements to *n*-tuples of variables. A *soft constraint* is a constraint where each association of its variables has an associated value from a partially ordered set $\mathcal{A}$. On this set, two operations are defined that allow for combination, $\times$, and comparison, $+$. If $\mathbf{0}$ is the unit element of $+$, and $\mathbf{1}$ is the unit element of $\times$, we can require appropriate properties on the two operations so that the tuple $\langle \mathcal{A}, +, \times, \mathbf{0}, \mathbf{1}\rangle$ is a c-semiring (Bistarelli et al., 1997).

Let us consider the relation $\leq_S$ over A such that $a \leq_S b$ iff $a + b = b$. It can be proved that $\leq_S$ is a partial order, $+$ and $\times$ are monotone on $\leq_S$, $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum. The relation $\leq_S$ gives us a way to compare (some of the) tuples of values and constraints. In fact, when we have $a \leq_S b$, we will say that *b is better than a*. Below, $\leq_S$ will be often indicated by $\leq$.

A *constraint system* is a tuple $CS=\langle \mathcal{S}, \mathcal{D}, \mathcal{V}\rangle$ where $\mathcal{S}$ is a c-semiring, $\mathcal{D}$ is a finite set (the domain of the variables) and $\mathcal{V}$ is an ordered set of variables. Given a semiring $\mathcal{S}=\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ and a constraint system $CS=\langle \mathcal{S}, \mathcal{D}, \mathcal{V}\rangle$, a *constraint* is a pair $\langle def, con\rangle$ where $con \subseteq \mathcal{V}$ and $def : \mathcal{D}^{|con|} \to A$. Therefore, a constraint specifies a set of variables (the ones in *con*), and assigns to each tuple of values of these variables an element of the semiring.

A *soft constraint satisfaction problem* (SCSP) is a pair $\langle C, con\rangle$ where $con \subseteq \mathcal{V}$ and C is a set of constraints: *con* is the set of variables of interest for the constraint set C, which however may concern also variables not in *con*.

We use the semiring $S_{SCSP} = \langle [0,1], max, min, 0, 1\rangle$, for the sake of demonstration, to build the example SCSP as shown in Fig. 6. Variables and constraints are represented, respectively, by nodes and by undirected (unary for c_1 and c_3 and binary for c_2) arcs, and semiring values are written to the right of the corresponding tuples. The variables of interest (that is the set *con*) are represented with a double circle. It is assumed that the domain D of the variables contains only elements a and b.

Combining soft constraints

Given two constraints $c_1 = \langle def_1, con_1\rangle$ and $c_2 = \langle def_2, con_2\rangle$, their *combination* $c_1 \otimes c_2$ is the constraint $\langle def, con\rangle$ defined by $con = con_1 \cup con_2$ and $def(t) = def_1(t\downarrow^{con}_{con_1}) \times def_2(t\downarrow^{con}_{con_2})$, where $t\downarrow^{X}_{Y}$ denotes the tuple of values over the variables in Y, obtained by projecting tuple t from X to Y. In other words, combining two constraints means building a new constraint involving all the variables of the original ones, and which associates to each tuple of domain values for such variables a semiring element that is obtained by multiplying the elements associated by the original constraints to the appropriate subtuples. In short, combination is performed via the multiplicative operation of the semiring.

Projecting soft constraints

Given a constraint $c = \langle def, con\rangle$ and a subset I of $\mathcal{V}$, the *projection* of c over I, written $c \Downarrow_I$ is the constraint $\langle def', con'\rangle$ where $con' = con \cap I$ and $def'(t') = \sum_{t/t\downarrow^{con}_{I\cap con} = t'} def(t)$. Informally, projecting means eliminating some variables. This is done by associating to each tuple over the remaining variables a semiring element that is the sum of the elements associated by the original constraint to all the extensions of this tuple over the eliminated variables. In short, projection is performed via the additive operation of the semiring.

Solution of an SCSP

The *solution* of an SCSP $P = \langle C, con\rangle$ is the constraint $Sol(P) = (\otimes C) \Downarrow_{con}$. That is, we combine all constraints, and then project over the

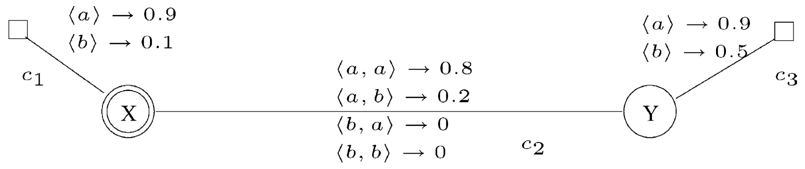


Figure 6 An example SCSP.

variables in *con*. In this way we get the constraint over *con* that is "induced" by the entire SCSP.

For example, the solution of the example SCSP in Fig. 6 consists of all possible pairs of domain values (that is, a domain value for each of the two variables) and an associated semiring element. Such an element is obtained by looking at the smallest value for all the subtuples (as many as the constraints) forming the pair. For example, for tuple $\langle a, a\rangle$ (that is, $x=y=a$), we have to compute the minimum between 0.9 (which is the value for $x=a$), 0.8 (which is the value for $\langle x=a, y=a\rangle$) and 0.9 (which is the value for $y=a$). Hence, the resulting value for this tuple is 0.8.

Soft constraint programming to analysing security protocols

We define the *security semiring* to specify each principal's trust on the confidentiality of each message, that is each principal's *security level* on each message. The security levels form the career set L of the security semiring.

$$\begin{aligned} L = \{ & unknown \equiv traded_{-1},\\ & private \equiv traded_0,\\ & traded_1,\\ & traded_2,\\ & \ldots,\\ & traded_n,\\ & public \equiv traded_{n+1}\} \end{aligned}$$

They range from the most secure (highest) level *unknown* (double named as $traded_{-1}$) to the least secure (lowest) level *public* (double named as $traded_{n+1}$). Intuitively, if A's security level on m is *unknown*, then no principal (included A) knows m according to A, and, if A's security level on m is *public*, then all principals potentially know m according to A. The lower A's security level on m, the higher the number of principals knowing m according to A. For simplicity, we state no relation between the granularity of the security levels and the number of principals. We define $+_{sec}$ and $\times_{sec}$ by the following axioms.

Axiom 1. $traded_i +_{sec} traded_j = traded_{min(i,j)}$

Axiom 2. $traded_i \times_{sec} traded_j = traded_{max(i,j)}$

The structure $\mathcal{S}_{\text{sec}} = \langle L, +_{\text{sec}}, \times_{\text{sec}}, public, unknown\rangle$ can be easily proved to be a c-semiring.

Using the security semiring, we define the *network constraint system*, which represents the computer network on which the security protocols can be executed. It does not depend on any specific protocol. It is expressed as $\text{CS}_n = \langle \mathcal{S}_{\text{sec}}, \mathcal{D}, \mathcal{V}\rangle$ where:

- $\mathcal{S}_{\text{sec}}$ is the security semiring just mentioned;
- $\mathcal{V}$ is a bounded set of variables, each standing for a principal;
- $\mathcal{D}$ is a bounded set of values including the empty message $\{|\,|\}$ and all atomic messages, as well as all messages recursively obtained by concatenation and encryption.

The development of the principals' security levels from manipulation of the messages seen during the protocol sessions can be formalised as a *security entailment*, which is an entailment relation between constraints (Bella and Bistarelli, 2002; Bistarelli et al., 2002). The relation is defined by four rules, one for each operation on the messages (splitting, decryption, concatenation and encryption). In brief, every time a principal invents a secret message, the principal's security level on the message decreases from *unknown* to *private*; every time the message is sent on the network the secret level of the message is decreased (for example from *private* to $traded_1$, from $traded_1$ to $traded_2$, etc.) to represent exposure to the network risks. This influences the principal's security levels on all messages that feature that secret, whose new (decreased) levels are computed by entailment. For example, encryption and concatenation build up new messages from known ones. The new messages must not get a worse security level than the known ones have. So, the corresponding rules choose the better of the given levels (Bella and Bistarelli, 2004). Precisely, if messages m_1 and m_2 have security levels v_1 and v_2, respectively, then the encrypted message $\{|m_1|\}_{m_2}$ and the compound message $\{|m_1, m_2|\}$ get a new level that is the better of v_1 and v_2, namely $v_1 +_{sec} v_2$.

At this stage, given a specific protocol, we represent the policy that accompanies the protocol as an SCSP called the *policy SCSP*. It formalises all admissible network configurations arising from the protocol execution as prescribed by the protocol designers. Therefore, any interleaving of protocol sessions in which no principal has acted maliciously is represented in the policy SCSP for the given protocol. The exact construction is done algorithmically, but is irrelevant to our discussion. Then, a particular network configuration arising from the protocol execution in the real world can be represented as another SCSP, an *imputable SCSP*. We have designed another algorithm for this task. There exists one such SCSP per each possible network configuration under the given protocol, while there exists one policy SCSP per each protocol (Bella and Bistarelli, 2004).

Given a security level *l*, we use *l-confidentiality* and *l-authentication* to formally capture the level of achievement of the goal. In case of confidentiality of a message for a principal in an SCSP, that level is the principal's security level on the message. It is computed by calculating the solution of the SCSP, projecting it on the principal and evaluating it on the message.

Definition 1 (*l-Confidentiality*). Given an *SCSP* p, let $Sol(\mathrm{p}) \Downarrow_{\{A\}} = \langle def_A, \{A\}\rangle$; *l*-confidentiality of *m* for *A* in p holds iff $def_A(m) = l$.

By comparing the solutions of the policy and the given imputable SCSPs we can formally define a confidentiality attack.

Definition 2 (*Confidentiality attack*). Given the policy SCSP P for a given protocol, and an imputable SCSP p for the same protocol, there is a confidentiality attack by *A* on *m* in p iff *l*-confidentiality of *m* in p for *A* holds, l'-confidentiality of *m* in p for *A* holds, and $l' < \mathrm{l}$.

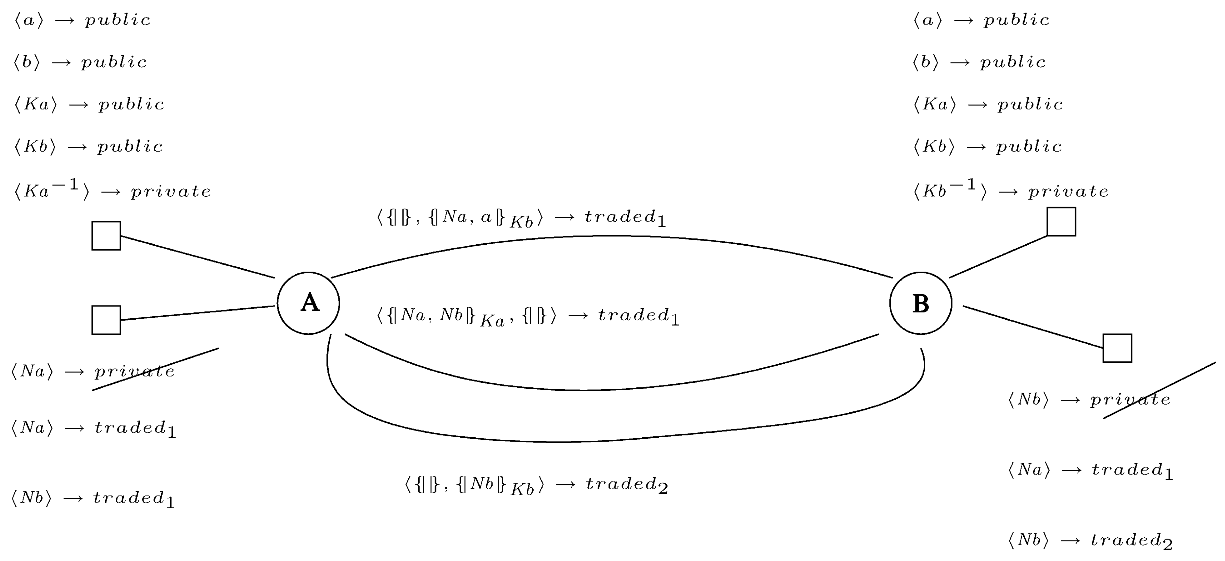
Therefore, if $Sol\ (\mathrm{P}) \Downarrow_{\{A\}} = \langle Def_A, \{A\}\rangle$, there is a confidentiality attack by *A* on *m* in p iff $def_A(m) < Def_A(m)$. Attacks can be realistically compared: the more an attack lowers a security level allowed by the policy SCSP, the worse that attack.

We exemplify this treatment on the Needham—Schroeder protocol seen above (section 'Realism'). Fig. 7 presents the fragment of policy SCSP for the protocol pertaining to a single session between principals *A* and *B*. Notice the unary constraints formalising each principal's security levels prior to the beginning of any protocol session, and the binary constraints each formalising a session step. If the reader is familiar with Roscoe's intensional/extensional specifications of protocols (Roscoe, 1996), the binary constraints may be seen as intensional specification as they assert a property in terms of communications within the protocol. Conversely, the unary constraints may be seen as extensional specification because they convey the principals' security level independently of the details of the protocol.

The figure shows that *A*'s security level on her nonce *Na* was initially *private* prior to the beginning of any protocol session, but it is lowered to *traded*$_1$ by entailment as soon as *A* invents it and sends it off in step 2 of the protocol. Likewise, *B*'s security level on *Nb* is *traded*$_2$ though it was originally *private*. All details that are irrelevant to the session are omitted from the figure. For example, all other principals' security levels on *Na* and *Nb* are *unknown* because the policy prescribes that no one acts maliciously.

Fig. 8 formalises the network configuration defined by Lowe's attack. The solution of this SCSP projected on variable *C* is a constraint that associates security level *traded*$_4$ to the nonce *Nb*. Following Definition 1, *Nb* is *traded*$_4$-confidential for *C* in this SCSP. Hence, by Definition 2, there is a confidentiality attack by *C* on *Nb* in this problem, because *Nb* got level *unknown* in the policy SCSP.

The problem solution projected on variable *B* gives security level *traded*$_2$ to the nonce *Na*, which



$\langle a\rangle \rightarrow public$
$\langle b\rangle \rightarrow public$
$\langle Ka\rangle \rightarrow public$
$\langle Kb\rangle \rightarrow public$
$\langle Ka^{-1}\rangle \rightarrow private$
$\langle Na\rangle \rightarrow$ ~~$private$~~
$\langle Na\rangle \rightarrow traded_1$
$\langle Nb\rangle \rightarrow traded_1$
A
B
$\langle\{\!|\,|\!\},\{\!|Na,a|\!\}_{Kb}\rangle \rightarrow traded_1$
$\langle\{\!|Na,Nb|\!\}_{Ka},\{\!|\,|\!\}\rangle \rightarrow traded_1$
$\langle\{\!|\,|\!\},\{\!|Nb|\!\}_{Kb}\rangle \rightarrow traded_2$
$\langle a\rangle \rightarrow public$
$\langle b\rangle \rightarrow public$
$\langle Ka\rangle \rightarrow public$
$\langle Kb\rangle \rightarrow public$
$\langle Kb^{-1}\rangle \rightarrow private$
$\langle Nb\rangle \rightarrow$ ~~$private$~~
$\langle Na\rangle \rightarrow traded_1$
$\langle Nb\rangle \rightarrow traded_2$

Figure 7 Fragment of the policy SCSP for the Needham—Schroeder protocol.

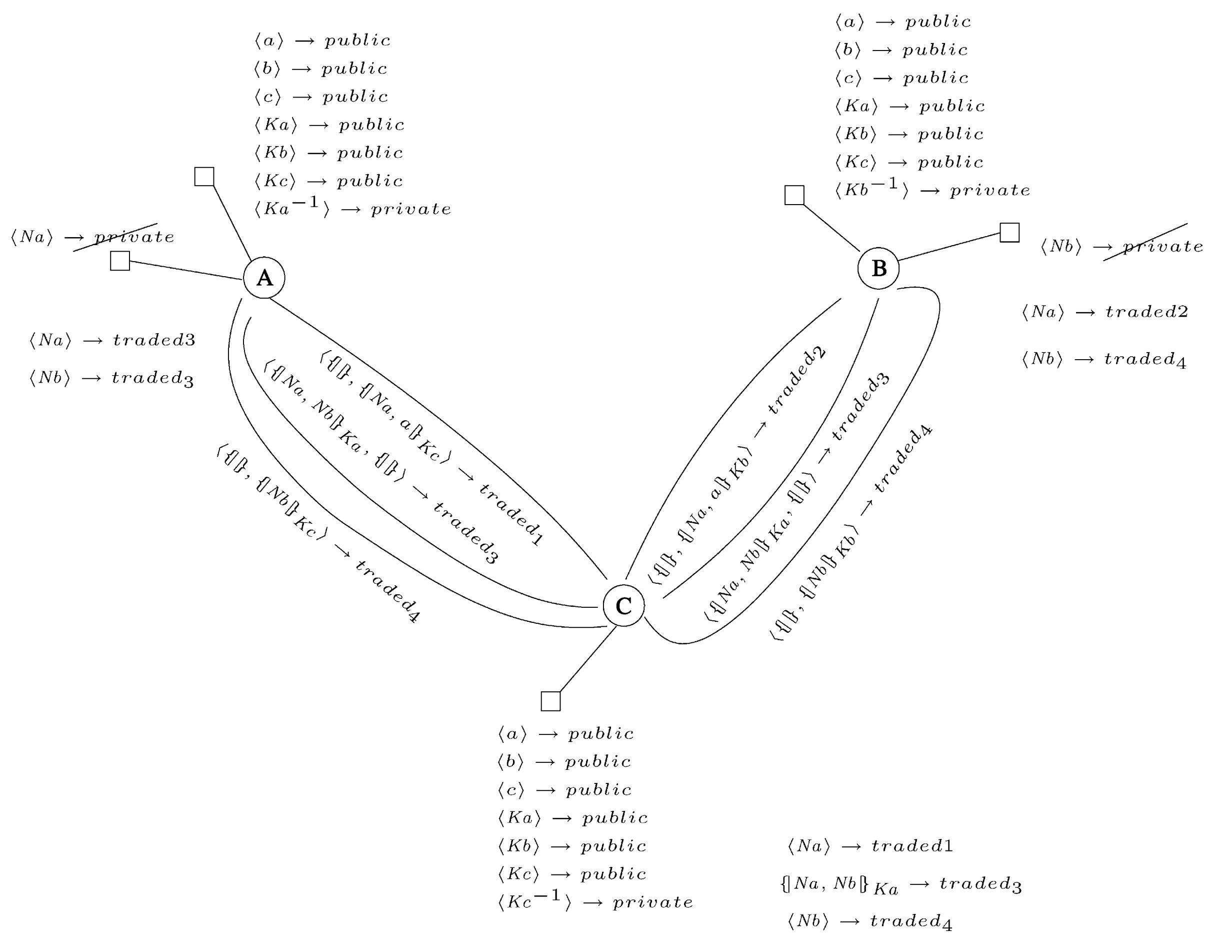

$\langle a\rangle \rightarrow public$
$\langle b\rangle \rightarrow public$
$\langle c\rangle \rightarrow public$
$\langle Ka\rangle \rightarrow public$
$\langle Kb\rangle \rightarrow public$
$\langle Kc\rangle \rightarrow public$
$\langle Ka^{-1}\rangle \rightarrow private$
$\langle Na\rangle \rightarrow$ ~~$private$~~
A
$\langle Na\rangle \rightarrow traded3$
$\langle Nb\rangle \rightarrow traded_3$
$\langle \{\!|\,|\!\}, \{\!|Na, a|\!\}_{Kc}\rangle \rightarrow traded_1$
$\langle \{\!|Na, Nb|\!\}_{Ka}, \{\!|\,|\!\}\rangle \rightarrow traded_3$
$\langle \{\!|\,|\!\}, \{\!|Nb|\!\}_{Kc}\rangle \rightarrow traded_4$
$\langle a\rangle \rightarrow public$
$\langle b\rangle \rightarrow public$
$\langle c\rangle \rightarrow public$
$\langle Ka\rangle \rightarrow public$
$\langle Kb\rangle \rightarrow public$
$\langle Kc\rangle \rightarrow public$
$\langle Kb^{-1}\rangle \rightarrow private$
B
$\langle Nb\rangle \rightarrow$ ~~$private$~~
$\langle Na\rangle \rightarrow traded2$
$\langle Nb\rangle \rightarrow traded_4$
$\langle \{\!|\,|\!\}, \{\!|Na, a|\!\}_{Kb}\rangle \rightarrow traded2$
$\langle \{\!|Na, Nb|\!\}_{Ka}, \{\!|\,|\!\}\rangle \rightarrow traded_3$
$\langle \{\!|\,|\!\}, \{\!|Nb|\!\}_{Kb}\rangle \rightarrow traded_4$
C
$\langle a\rangle \rightarrow public$
$\langle b\rangle \rightarrow public$
$\langle c\rangle \rightarrow public$
$\langle Ka\rangle \rightarrow public$
$\langle Kb\rangle \rightarrow public$
$\langle Kc\rangle \rightarrow public$
$\langle Kc^{-1}\rangle \rightarrow private$
$\langle Na\rangle \rightarrow traded1$
$\{\!|Na, Nb|\!\}_{Ka} \rightarrow traded_3$
$\langle Nb\rangle \rightarrow traded_4$

Figure 8 Fragment of the imputable SCSP corresponding to Lowe's attack.

instead got level *unknown* in the policy SCSP. This signifies that *B* has learnt a nonce that he was not allowed to learn by policy, hence there is an indeliberate confidentiality attack by *B* on *Na*. Notice that the two attacks—the deliberate (Lowe, 1995) and the indeliberate (Bella and Bistarelli, 2002)—are uniformly formalised.

Our approach is realistic, accurate and formal. It is amenable to mechanization by model checking if we appropriately bond all quantities, hence the possible network configurations are finite (Durgin et al., 1999).

Conclusions

We have laid the ground towards the development of a definition of Information Assurance for security protocols. We require that security protocols be analysed realistically, accurately and formally.

"Realistically" means that the model underlying the analysis should exceed the limits of the classical Dolev and Yao's model. We showed how this highlights another consequence of Lowe's attack on the popular asymmetric Needham—Schroeder protocol.

"Accurately" means that the protocol goals should not be considered mere boolean properties because security never is a boolean feature. In contrast, we advocate reasoning about levels of confidentiality or authentication.

"Formally" means that the analysis should be conducted within a formal framework. The literature seems to be missing an approach to protocol analysis that embeds all the three features, so we have sketched a new approach based on Constraint Solving. This confirms that the three requirements we set towards Information Assurance for security protocols can coexist together within a single approach to analysing protocols.

Acknowledgements

We are grateful to Simon Foley, Michael Leuschel and Fabio Massacci for useful discussions on the topic of this paper.

References

Abadi M. Secrecy by typing in security protocols. Journal of the ACM 1999;46(5):749—86.

Abadi M, Fournet C. Mobile values, new names, and secure communication. In: Proceedings of the 28th ACM SIGACT-SIGPLAN symposium on principles of programming languages (POPL'01). ACM Press and Addison Wesley; 2001. p. 104—15.

Abadi M, Needham RM. Prudent engineering practice for cryptographic protocols. IEEE Transactions on Software Engineering January 1996;22(1):6—15.

Anderson R. Why cryptosystems fail, Proceedings of the 1st ACM conference on communications and computer security (CCS'93). ACM Press and Addison Wesley; 1993. p. 217—27.

Anderson R, Needham RM. Robustness principles for public key protocols. In: Coppersmith D, editor. Proceedings of advances in cryptography—CRYPTO'95, LNCS 963. Springer-Verlag; 1995. p. 236—47.

Bella G, Bistarelli S. Confidentiality levels and deliberate/indeliberate protocol attacks. In: Christianson B, Crispo B, Harbison WS, Roe M, editors. Proceedings of the 10th international workshop on security protocols Cambridge, UK, LNCS 2845. Springer; 2002. p. 104—19.

Bella G, Bistarelli S. Soft constraint programming to analysing security protocols. Theory and Practice of Logic Programming 2004;4(5):1—28.

Bella G, Paulson LC. Kerberos version IV: inductive analysis of the secrecy goals. In: Quisquater J-J, Deswarte Y, Meadows C, Gollmann D, editors. Proceedings of the 5th European symposium on research in computer security (ESORICS'98), LNCS 1485. Springer-Verlag; 1998. p. 361—75.

Bella G, Massacci F, Paulson LC. Verifying the SET registration protocols. IEEE Journal of Selected Areas in Communications 2003;21(1):77—87.

Bella G, Bistarelli S, Massacci F. A protocol's life after attacks. In: Proceedings of the 11th international workshop on security protocols Cambridge, UK, LNCS series. Springer-Verlag; 2003, in press.

Bellare M, Rogaway P. Provably secure session key distribution—the three party case. In: Proceedings of the 27th ACM SIGACT symposium on theory of computing (STOC'95). ACM Press and Addison Wesley; 1995. p. 57—66.

Bellovin SM, Merritt M. Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise. In: ACM conference on computer and communications security; 1993. p. 244—50.

Bistarelli S. Semirings for soft constraint solving and programmingIn: LNCS 2962. Springer; 2004.

Bistarelli S, Montanari U, Rossi F. Semiring-based constraint solving and optimization. Journal of the ACM 1997;201—36.

Bistarelli S, Montanari U, Rossi F. Soft concurrent constraint programming. In: Proceedings of the 11th European symposium on programming (ESOP'02). LNCS 2305. Springer-Verlag; 2002. p. 53—67.

Burrows M, Abadi M, Needham RM. A logic of authentication. Proceedings of the Royal Society of London 1989;426:233—71.

Cheswick W, Bellovin SM, Rubin AD. Firewalls and Internet security: repelling the wily hacker. Addison-Wesley 2004.

Cholvy L, Cuppens F. Analyzing consistency of security policies. Proceedings of the 16th IEEE symposium on security and privacy 1997.

Corbatò FJ, Vyssotsky VA. Introduction and overview of the multics system. In: Proceedings of the American Federation of information processing societies conference (AFIPS'65); 1965. p. 185—96.

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

Durgin N, Lincoln P, Mitchell J, Scedrov A. Undecidability of bounded security protocols. In: Proceedings of the FMSP'99.

Fábrega FJT, Herzog JC, Guttman JD. Strand spaces: why is a security protocol correct? In: Proceedings of the 17th IEEE symposium on security and privacy. IEEE Press; 1998.

Gollmann D. What do we mean by entity authentication? Proceedings of the 15th IEEE symposium on security and privacy. IEEE Press; 1996. p. 46—54.

Gollmann D. On the verification of cryptographic protocols—a tale of two committees. Proceedings of the workshop on secure architectures and information flow. In: ENTCS 32. Elsevier Science; 2000a.

Gollmann D. What is authentication? In: Christianson B, Crispo B, Malcolm J, Roe M, editors. Proceedings of the 7th international workshop on security protocols, LNCS 1796. Springer-Verlag; 2000b. p. 65—72.

Gollmann D. Authentication—myths and misconceptions. In: Lam K-Y, Shparlinski IE, Wang H, Xing C, editors. Proceedings of the workshop on cryptography and computational number theory (CCNT'99). Birkhäuser; 2001. p. 203—26.

Halpern JY, Weissman V. Using first-order logic to reason about policies. Proceedings of the 16th IEEE computer security foundations workshop. IEEE Press; 2003.

Lowe G. An attack on the Needham—Schroeder public-key authentication protocol. Information Processing Letters 1995;56(3):131—3.

Lowe G. A hierarchy of authentication specifications. Proceedings of the 10th IEEE computer security foundations workshop. IEEE Press; 1997. p. 31—43.

Needham RM, Schroeder MD. Using encryption for authentication in large networks of computers. Communications of the ACM 1978;21(12):993—9.

Neuman BC, Ts'o T. Kerberos: an authentication service for computer networks, from IEEE communications magazine. September, 1994, William Stallings, Practical cryptography for data Internetworks. IEEE Press; 1996.

Paulson LC. Mechanized proofs for a recursive authentication protocol. In: Proceedings of the 10th IEEE computer security foundations workshop. IEEE Press; 1997. p. 84—95.

Paulson LC. The inductive approach to verifying cryptographic protocols. Journal of Computer Security 1998;6:85—128.

Paulson LC. Relations between secrets: two formal analyses of the Yahalom protocol. Journal of Computer Security 2001; 9(3):197—216.

Roscoe AW. Intensional specifications of security protocols. In: Proceedings of the 9th IEEE computer security foundations workshop. IEEE Press; 1996. p. 28—38.

Ryan PYA, Schneider SA. An attack on a recursive authentication protocol: a cautionary tale. Information Processing Letters 1998;65. Elsevier Science Publishers (North-Holland) Amsterdam.

Ryan PYA, Schneider SA. The modelling and analysis of security protocols: the CSP approach. Addison-Wesley; 2000.

Syverson PF. Limitations on design principles for public key protocols. In: Proceedings of the 15th IEEE symposium on security and privacy. IEEE Press; 1996. p. 62—72.

Shmatikov V, Mitchell JC. Analysis of a fair exchange protocol, Network and distributed system security symposium (NDSS-00); 2000. p. 119—28.

Giampaolo Bella is an Assistant Professor at the University of Catania (Italy), teaching the Computer Security course and classes in Computer Architectures. His main research interests are in the use of formal methods for the verification of crucial security properties, but he is also interested in hardware

verification and in constraint programming. He was a Research Associate at the Technical University of Munich (Germany) in the year 2000, after he received his Ph.D. from the Cambridge University Computer Laboratory. His Ph.D. dissertation, entitled ''Inductive Verification of Cryptographic Protocols'' (CUCL Technical Report 493), focuses on how to use inductive techniques with a mechanized proof assistant to verify real-world security protocols.

Stefano Bistarelli is an Assistant Professor of Computer Science at the Department of Science of the University of Chieti-Pescara and External Researcher of the Institute of Informatics and Telematics of C.N.R. Pisa. He obtained his Ph.D. in Computer Science in 2001 at the Computer Science Department of the University of Pisa. His thesis was awarded by the Italian Association of Artificial Intelligence and by the Italian Chapter of the European Association of Theoretical Computer Science. His research interests range from Artificial Intelligence to Programming Languages, with particular attention to constraint programming, constraint solution algorithms, and soft constraints. He is also interested in Security. Recently, he published his work as book LNCS 2962 by Springer.