TLP **4** (5 & 6): 545–572, 2004.
DOI: 10.1017/S1471068404002121 Printed in the United Kingdom

Soft constraint programming to analysing security protocols

GIAMPAOLO BELLA

Dipartimento di Matematica e Informatica, Università di Catania, Viale A. Doria 6, I-95125 Catania, Italy
(*e-mail:* giamp@dmi.unict.it)

STEFANO BISTARELLI

Istituto di Informatica e Telematica, CNR, Via G. Moruzzi 1, I-56124 Pisa, Italy
(*e-mail:* stefano.bistarelli@iit.cnr.it)

Dipartimento di Scienze, Università "D'Annunzio", Viale Pindaro 42, I-65127 Pescara, Italy
(*e-mail:* bista@sci.unich.it)

Abstract

Security protocols stipulate how the remote principals of a computer network should interact in order to obtain specific security goals. The crucial goals of *confidentiality* and *authentication* may be achieved in various forms, each of different strength. Using soft (rather than crisp) constraints, we develop a uniform formal notion for the two goals. They are no longer formalised as mere yes/no properties as in the existing literature, but gain an extra parameter, the *security level*. For example, different messages can enjoy different levels of confidentiality, or a principal can achieve different levels of authentication with different principals. The goals are formalised within a general framework for protocol analysis that is amenable to mechanisation by model checking. Following the application of the framework to analysing the asymmetric Needham-Schroeder protocol (Bella and Bistarelli 2001; Bella and Bistarelli 2002), we have recently discovered a new attack on that protocol as a form of retaliation by principals who have been attacked previously. Having commented on that attack, we then demonstrate the framework on a bigger, largely deployed protocol consisting of three phases, Kerberos.

KEYWORDS: security, constraints, security protocols, soft constraints

1 Overview

A number of applications ranging from electronic transactions over the Internet to banking transactions over financial networks make use of security protocols. It has been shown that the protocols often fail to meet their claimed goals (Abadi and Needham, 1996; Lowe, 1996), so a number of approaches for analysing them formally have been developed (Lowe, 1995; Bella and Riccobene, 1997, Paulson, 1998; Bodei *et al.*, 2001; Bodei *et al.*, 1996; Focardi *et al.*, 2000b; Focardi *et al.*, 2000a; Abadi, 1997; Abadi and Gordon, 1997). The threats to the protocols come from malicious principals who manage to monitor the network traffic building fake messages at will. A major protocol goal is *confidentiality*, confirming that a message remains undisclosed to malicious principals. Another crucial goal is

authentication, confirming a principal's participation in a protocol session. These goals are formalised in a mere "yes or no" fashion in the existing literature. One can just state whether a key is confidential or not, or whether a principal authenticates himself with another or not.

Security goals are not simple boolean properties. "Security is not a simple boolean predicate; it concerns how well a system performs certain functions" (Anderson, 1993). Indeed, experience shows that system security officers exercise care in applying any firm boolean statements to the real world even if they were formal. In general, formal security proofs are conducted within simplified models. Therefore, security officers attempt to bridge the gap between those models and the real word by adopting the largest possible variety of security measures all together. For example, firewalls accompany SSH connections. Limiting the access to certain ports of a server is both stated on the firewall *and* on the server itself. Biometric technology recently set aside the use of passwords to strengthen authentication levels of principals. Still, principals' credentials can be constrained within a validity time interval. The officer shall balance the cost of an extra security measure with his perception of the unmanaged risks. Any decision will only achieve a certain *security level*.

Security levels also characterise security patches (Foley, 2002). Each patch in fact comes with a recommendation that is proportionate to the relevance of the security hole the patch is meant to fix. Patches may be critical, or recommended, or suggested, or software upgrade, etc. Depending on the cost of the patch and on the relevance of the hole, the security officer can decide whether or not to upgrade the system. It is a security policy what establishes the maximum level up until a patch can be ignored.

This all confirms that real-world security is based on security levels rather than on categorical, definitive, security assurances. In particular, security levels characterise the protocol goals of confidentiality and authentication. Focusing on the former goal, we remark that different messages require "specific degrees of protection against disclosure" (Gray, 2001). For example, a user password requires higher protection than a *session key*, which is only used for a single protocol session. Intuitively, a password ought to be "more confidential" than a session key. Also, a confidentiality attack due to off-line cryptanalysis should not be imputed to the protocol design. Focusing on authentication, we observe that a certificate stating that K is a principal A's *public key* authenticates A very weakly. The certificate only signifies that A is a registered network principal, but in fact confers no guarantee about A's participation in a specific protocol session. A message signed by A's *private key* authenticates A more strongly, for it signifies that A participated in some protocol session in order to sign the message.

Our original contributions. We have developed enriched formal notions for the two goals. Our definitions of *l-confidentiality* and of *l-authentication* highlight the security level l. One of the advantages of formalising security levels is to capture the real-world non-boolean concepts of confidentiality and authentication.

Each principal assigns his own security level to each message – different levels to different messages – expressing the principal's trust on the confidentiality of

the message. So, we can formalise that different goals are granted to different principals. By a *preliminary analysis*, we can study what goals the protocol achieves in ideal conditions where no principal acts maliciously. An *empirical analysis* may follow, whereby we can study what goals the protocol achieves on a specific network configuration arising from the protocol execution in the real world. Another advantage of formalising security levels is that we can variously compare attacks – formally.

Our security levels belong to a finite linear order. Protocol messages can be combined (by concatenation or encryption) or broken down (by splitting or decryption) into new messages. We must be able to compute the security levels of the newly originated messages out of those of the message components. Therefore, we introduce a semiring whose career set is the set of security levels. Its two functions provide the necessary computational capabilities. Our use of a semiring is loosely inspired to Denning's use of a lattice to characterising secure flows of information through computer systems (Denning, 1976). The idea of using levels to formalise *access rights* is in fact due to her. Denning signals an attack whenever an object is assigned a label worse than that initially specified. We formalise protocol attacks in the same spirit.

Another substantial contribution of the present work is the embedding of a novel *threat model* in a framework for protocol analysis. Our threat model regards *all principals as attackers if they perform, either deliberately or not, any operation that is not admitted by the protocol policy*. Crucially, it allows any number of non-colluding attackers. This overcomes the limits of Dolev and Yao's popular threat model (Dolev and Yao, 1983), which reduces a number of colluding principals to a single attacker. The example that follows shows the deeper adherence of our threat model to the real world, where anyone may attempt to subvert a protocol for his (and only his) own sake.

Let us consider Lowe's popular attack on the asymmetric Needham-Schroeder protocol (Lowe, 1995) within Dolev and Yao's threat model. It sees an attacker C masquerade as A with B, after A initiated a session with C. This scenario clearly contains an authentication attack following the confidentiality attack whereby C learns B's nonce Nb for A. Lowe reports that, if B is a bank for example, C can steal money from A's account as follows (Lowe, 1995):

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

where $\{\!|m|\!\}_K$ stands for the ciphertext obtained encrypting message m with key K (external brackets of concatenated messages are omitted). The bank B would honour the request believing it came from the account holder A.

We argue that the analysis is constrained by the limitations of the threat model. Plunging Lowe's scenario within our threat model highlights that B has mounted an indeliberate confidentiality attack on nonce Na, which was meant to be known to A and C only. As C did previously, B can equally decide to illegally exploit his knowledge of Na. If A is a bank, B can steal money from C's account as follows:

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

The bank A would honour the request believing it came from the account holder C.

The details of our findings on the Needham–Schroeder protocol can be found elsewhere (Bella and Bistarelli, 2002). Our empirical analysis of the protocol uniformly detects *both* attacks in terms of decreased security levels: both C's security level on Nb and B's security level on Na become lower than they would be if C did not act maliciously.

The framework presented throughout this paper supersedes an existing kernel (Bella and Bistarelli, 2001, pear) by extending it with five substantial features: (1) The principles of the new threat model that allows all principals to behave maliciously; (2) the combination of preliminary and empirical analyses; (3) the study of the authentication goal; (4) the formalisation of an additional event whereby a principal discovers a secret by cryptanalysis – this allows a larger number of network configurations to be studied through an empirical analysis; and (5) a comprehensive study of how message manipulation and exposure to the network lowers the security level of the message – this is implemented by a new algorithm called RiskAssessment.

Since we only deal with bounded protocols and finite number of principals, our framework is amenable to mechanisation by model checking, although this exceeds the purposes of the present paper.

Findings on the running example – Kerberos. We demonstrate our framework on a largely deployed protocol, Kerberos. Our preliminary analysis of the protocol formally highlights that the loss of an *authorisation key* would be more serious than the loss of a *service key* by showing that the former has a higher security level than the latter. By similar means, the preliminary analysis also allows us to compare the protocol goals in the forms they are granted to initiator and responder. It shows that authentication of the responder with the initiator is weaker than that of the initiator with the responder. To the best of our knowledge, developing such detailed observations formally is novel to the field of protocol analysis.

The empirical analysis that follows studies an example scenario in which a form of cryptanalysis was performed. The analysis highlights how that event lowers a number of security levels, and so lowers confidentiality and authentication for a number of principals.

Paper outline. After an outline on semiring-based Soft Constraints Satisfaction Problems (SCSPs) (section 2), our framework for protocol analysis is described in section 3. Then, the Kerberos protocol is introduced in section 4 and analysed in section 5. Some conclusions (section 6) terminate the presentation.

2 Soft constraints

Several formalisations of the concept of *soft constraints* are currently available (Schiex *et al.*, 1995; Dubois *et al.*, 1993; Freuder and Wallace, 1992; Fargier and Lang, 1993). In the following, we refer to one that is based on c-semirings (Bistarelli, 2001; Bistarelli *et al.*, 1997, 2001), which can be shown to generalise and express many of the others.

A soft constraint may be seen as a constraint where each instantiation of its variables has an associated value from a partially ordered set. Combining constraints will then have to take into account such additional values, and thus the formalism has also to provide suitable operations for combination ($\times$) and comparison ($+$) of tuples of values and constraints. This is why this formalisation is based on the concept of semiring, which is just a set plus two operations.

A semiring is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ such that:

- A is a set and $\mathbf{0}, \mathbf{1} \in A$;
- $+$ is commutative, associative and $\mathbf{0}$ is its unit element;
- $\times$ is associative, distributes over $+$, $\mathbf{1}$ is its unit element and $\mathbf{0}$ is its absorbing element.

A *c-semiring* is a semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ such that: $+$ is idempotent, $\mathbf{1}$ is its absorbing element and $\times$ is commutative.

Let us consider the relation $\leqslant_S$ over A such that $a \leqslant_S b$ iff $a + b = b$. Then it is possible to prove that (see Bistarelli *et al.*, 1997):

- $\leqslant_S$ is a partial order;
- $+$ and $\times$ are monotone on $\leqslant_S$;
- $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum;
- $\langle A, \leqslant_S\rangle$ is a complete lattice and, for all $a, b \in A$, $a + b = lub(a, b)$.

Moreover, if $\times$ is idempotent, then: $+$ distributes over $\times$; $\langle A, \leqslant_S\rangle$ is a complete distributive lattice and $\times$ its glb.

Informally, the relation $\leqslant_S$ gives us a way to compare (some of the) tuples of values and constraints. In fact, when we have $a \leqslant_S b$, we will say that b *is better than* a. Below, $a \leqslant_S b$ will be often indicated by $a \leqslant b$.

A *constraint system* is a tuple $CS = \langle \mathscr{S}, \mathscr{D}, \mathscr{V}\rangle$ where $\mathscr{S}$ is a c-semiring, $\mathscr{D}$ is a bounded set (the domain of the variables) and $\mathscr{V}$ is an ordered set of variables.

Given a semiring $\mathscr{S} = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ and a constraint system $CS = \langle \mathscr{S}, \mathscr{D}, \mathscr{V}\rangle$, a *constraint* is a pair $\langle def, con\rangle$ where $con \subseteq \mathscr{V}$ and $def : \mathscr{D}^{|con|} \to A$. 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 $con \subseteq \mathscr{V}$ 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*.

Notice that a classical constraint satisfaction problem (CSP) is an SCSP where the chosen c-semiring is:

$$S_{CSP} = \langle \{false, true\}, \vee, \wedge, false, true\rangle.$$

Fuzzy CSPs (Dubois *et al.*, 1993; Ruttkay, 1994; Schiex, 1992) can instead be modelled as SCSPs by choosing the c-semiring:

$$S_{FCSP} = \langle [0, 1], max, min, 0, 1\rangle.$$

Figure 1 shows the graph representation of a fuzzy CSP. Variables and constraints are represented respectively by nodes and by undirected (unary for c_1 and c_3 and

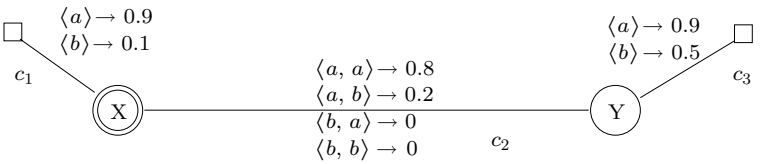

Fig. 1. A fuzzy CSP.

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. Here we assume that the domain D of the variables contains only elements a and b.

Combining and projecting 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 $def(t) = def_1(t \downarrow_{con_1}^{con}) \times def_2(t \downarrow_{con_2}^{con})$, where $t \downarrow_Y^X$ denotes the tuple of values over the variables in Y, obtained by projecting tuple t from X to Y. In 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 which is obtained by multiplying the elements associated by the original constraints to the appropriate subtuples.

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

In short, combination is performed via the multiplicative operation of the semiring, and projection via the additive operation.

Solutions. The *solution* of an SCSP problem $\mathsf{P} = \langle C, con \rangle$ is the constraint $Sol(\mathsf{P}) = (\bigotimes C) \Downarrow_{con}$. That is, we combine all constraints, and then project over the variables in *con*. In this way we get the constraint over *con* which is "induced" by the entire SCSP.

For example, each solution of the fuzzy CSP of Figure 1 consists of a pair 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.

Partial Information and Entailment. A constraint is a relation among a specified set of variables. It gives some information on the set of possible values that those variables may assume. Such information is usually not complete since a constraint may be satisfied by several assignments of values of the variables (in contrast to the situation that we have when we consider a valuation, which tells us the only possible

assignment for a variable). Therefore, it is natural to describe constraint systems as systems of *partial* information (Saraswat, 1993).

The basic ingredients of a constraint system (defined following the information systems idea) are a set $D*$ of *primitive constraints* or *tokens*, each expressing some partial information, and an entailment relation $\vdash$ defined on $\wp(D*) \times D*$ (or its extension defined on $\wp(D*) \times \wp(D*)$, such that $u \vdash v$ iff $u \vdash P$ for all $P \in v$) satisfying:

- $u \vdash P$ for all $P \in u$ (reflexivity) and
- if $u \vdash v$ and $v \vdash z$, then $u \vdash z$ (transitivity).

As an example of entailment relation, consider $D*$ as the set of equations over the integers; then $\vdash$ could include the pair $\langle \{x = 3, x = y\}, y = 3 \rangle$, which means that the constraint $y = 3$ is entailed by the constraints $x = 3$ and $x = y$. Given $X \in \wp(D*)$, let $\overline{X}$ be the set X closed under entailment. Then, a constraint in an information system $\langle \wp(D*), \vdash \rangle$ is simply an element of $\overline{\wp(D)}$.

In the SCSP framework a token is simply a soft constraint (that is domain assignment and some associated semiring values); the entailment rule will compute/change new soft constraint (and new levels) (Bistarelli *et al*., 2002).

3 Constraint programming for protocol analysis

This section presents our framework for analysing security protocols. Using soft constraints requires the definition of a c-semiring.

Our *security semiring* (section 3.1) is used to specify each principal's trust on the security of each message, that is each principal's *security level* on each message. The security levels range from the most secure (highest, greatest) level *unknown* to the least secure (lowest, smallest) level *public*. 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.

Using the security semiring, we define the *network constraint system* (section 3.2), which represents the computer network on which the security protocols can be executed. The development of the principals' security levels from manipulation of the messages seen during the protocol sessions can be formalised as a *security entailment* (section 3.3), that is an entailment relation between constraints. Then, given a specific protocol to analyse, we represent its assumptions in the *initial SCSP* (section 3.4). All admissible network configurations arising from the protocol execution as prescribed by the protocol designers can in turn be represented in the *policy SCSP* (section 3.5). We also explain how to represent any network configuration arising from the protocol execution in the real world as an *imputable SCSP* (section 3.7).

Given a security level l, establishing whether our definitions of *l-confidentiality* (section 3.8) or *l-authentication* (section 3.9) hold in an SCSP requires calculating

the solution of the imputable SCSP and projecting it on certain principals of interest. The higher l, the stronger the goal. For example, *unknown-confidentiality* is stronger than *public-confidentiality*, or, A's security level on B's public key (learnt via a certification authority) being *public* enforces *public-authentication* of B with A, which is the weakest form of authentication. We can also formalise confidentiality or authentication attacks. The definitions are given within specific methodologies of analysis.

By a *preliminary analysis*, we can study what goals the protocol achieves in ideal conditions where no principal acts maliciously, namely the very best the protocol can guarantee. We concentrate on the policy SCSP, calculate its solution, and project it on a principal of interest. The process yields the principal's security levels, which allow us to study what goals the protocol grants to that principal in ideal conditions, and which potential attacks would be more serious than others for the principal. For example, the most serious confidentiality attacks would be against those messages on which the principal has the highest security level.

An *empirical analysis* may follow, whereby we can study what goals the protocol achieves on a specific network configuration arising from the protocol execution in the real world. We concentrate on the corresponding imputable SCSP, calculate its solution and project it on a principal of interest: we obtain the principal's security levels on all messages. Having done the same operations on the the policy SCSP, we can compare the outcomes. If some level in the imputable is lower than the corresponding level in the policy, then there is an attack in the imputable. In fact, some malicious activity contributing to the network configuration modelled by the imputable SCSP has taken place so as to lower some of the security levels stated by the policy SCSP.

The following, general treatment is demonstrated in section 4.

3.1 The security semiring

Let n be a natural number. We define the set L of *security levels* as follows.

$$L = \{unknown,\ private,\ traded_1,\ traded_2, \ldots, traded_n,\ public\}$$

where *unknown* is the maximum element of L and *public* is the minimum one.

Although our security levels may appear to resemble Abadi's *types* (Abadi, 1999), there is in fact little similarity. Abadi associates each message to either type *public*, or *secret*, or *any*, whereas we define n security levels with no bound on n, and *each principal associates a level of his own to each message* as explained in the following. Also, while Abadi's *public* and *private* cannot be compared, our levels are linearly ordered.

The security levels express each principal's trust on the security of each message. Clearly, *unknown* is the highest security level. We will show how, under a given protocol, a principal assigns *unknown* to all messages that do not pertain to the protocol, and to all messages that the principal does not know. A principal will assign *private* to all messages that, according to himself, are known to him alone, such as his own long-term keys, the nonces invented during the protocol execution,

or any secrets discovered by cryptanalysis. In turn, a principal will assign $traded_i$ to the messages that are exchanged during the protocol: the higher the index i, the more the messages have been handled by the principals, and therefore the more principals have potentially learnt those messages. So, *public* is the lowest security level. These security levels generalise, by the $traded_i$ levels, the four levels that we have discussed elsewhere (Bella and Bistarelli, 2001).

We introduce an additive operator, $+_{sec}$, and a multiplicative operator, $\times_{sec}$. To allow for a compact definition of the two operators, and to simplify the following treatment, let us define a convenient double naming:

$$\begin{aligned} &-unknown \equiv traded_{-1} \\ &-private \equiv traded_0 \\ &-public \equiv traded_{n+1} \end{aligned}$$

Let us consider an index i and an index j both belonging to the closed interval $[-1,\ n+1]$ of integers. We define $+_{sec}$ and $\times_{sec}$ by the following axioms.

Ax. 1: $traded_i \ +_{sec} \ traded_j \ = \ traded_{min(i,j)}$
Ax. 2: $traded_i \ \times_{sec} \ traded_j \ = \ traded_{max(i,j)}$

Theorem 1 (*Security Semiring*)
The structure $\mathscr{S}_{sec} = \langle L, +_{sec}, \times_{sec}, public, unknown \rangle$ is a c-semiring.

Proof hint
Clearly, $\mathscr{S}_{sec}$ enjoys the same properties as the structure $S_{finite-fuzzy} = \langle \{-1, \ldots, n+1\}, max, min, -1, n+1 \rangle$. Indeed, the security levels can be mapped into the values in the range $-1, \ldots, n+1$ (*unknown* being mapped into 0, *public* being mapped into $n+1$); $+_{sec}$ can be mapped into function *max*; $\times_{sec}$ can be mapped into function *min*. Moreover, $S_{finite-fuzzy}$ can be proved a *c*-semiring as done with the fuzzy semiring (Bistarelli *et al*., 1997). □

Our security semiring is in fact a linear order, but the general treatment provided here complies with the general case where $+_{sec}$ and $\times_{sec}$ must be mapped into more complex functions than *max* and *min*.

3.2 The network constraint system

We define a constraint system $CS_n = \langle \mathscr{S}_{sec}, \mathscr{D}, \mathscr{V} \rangle$ where:

- $\mathscr{S}_{sec}$ is the security semiring (§3.1);
- $\mathscr{V}$ is a bounded set of variables.
- $\mathscr{D}$ 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.

We name CS_n as *network constraint system*. The elements of $\mathscr{V}$ stand for the network principals, and the elements of $\mathscr{D}$ represent all possible messages. Atomic messages typically are principal names, timestamps, nonces and cryptographic keys. Concatenation and encryption operations can be applied a bounded number of times.

Encryption

$$\frac{def(m_1) = v_1;\quad def(m_2) = v_2;\quad def(\{\!|m_1|\!\}_{m_2}) = v_3}{def(\{\!|m_1|\!\}_{m_2}) = (v_1 +_{sec} v_2) \times_{sec} v_3}$$

Concatenation

$$\frac{def(m_1) = v_1;\quad def(m_2) = v_2;\quad def(\{\!|m_1, m_2|\!\}) = v_3;}{def(\{\!|m_1, m_2|\!\}) = (v_1 +_{sec} v_2) \times_{sec} v_3}$$

Decryption

$$\frac{def(m_1) = v_1;\quad def(m_2^{-1}) = v_2;\quad def(\{\!|m_1|\!\}_{m_2}) = v_3;\quad v_2, v_3 < unknown}{def(m_1) = v_1 \times_{sec} v_2 \times_{sec} v_3}$$

Splitting

$$\frac{def(m_1) = v_1;\quad def(m_2) = v_2;\quad def(\{\!|m_1, m_2|\!\}) = v_3}{def(m_1) = v_1 \times_{sec} v_3;\quad def(m_2) = v_2 \times_{sec} v_3}$$

Fig. 2. Computation rules for security levels.

Notice that CS_n does not depend on any protocols, for it merely portrays a computer network on which any protocol can be implemented. Members of $\mathscr{V}$ will be indicated by capital letters, while members of $\mathscr{D}$ will be in small letters.

3.3 Computing the security levels by entailment

Recall that each principal associates his own security levels to the messages. Those levels evolve while the principal participates in the protocol and performs off-line operations such as encryption, concatenation, decryption, and splitting. We define four rules to compute the security levels that each principal gives to the newly generated messages. The rules are presented in Figure 2, where function *def* is associated to a generic constraint projected on a generic principal A.

A message is known to a principal when the principal's security level on that message is lower than *unknown*. 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. 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 $\{\!|m_1, m_2|\!\}$, whose current level be some v_3, get a new level that is the better of v_1 and v_2, "normalised" by v_3. This normalisation, which is done in terms of the $\times_{sec}$ operator, influences the result only if the new level is better than the current level.

Decryption and splitting break down known messages into new ones. The new messages must not get a better security level than the known ones have. So, the corresponding rules choose the worse of the given levels by suitable applications of $\times_{sec}$, and assign it to the new messages. Recall that, in case of asymmetric cryptography, the decryption key for a ciphertext is the inverse of the key that was used to create the ciphertext. So the rule for decryption considers the inverse of message m_2 and indicates it as m_2^{-1}. Conversely, in case of symmetric cryptography,

we have $m_2^{-1} = m_2$. The rule for splitting presupposes that concatenation is transparent in the sense that, for any index n, an n-component message can be seen as a 2-component message, namely $\{\!|m_1, m_2, \ldots, m_n|\!\} = \{\!|m_1, \{\!|m_2, \ldots, m_n|\!\}\!|\}$. We now define a binary relation between constraints.

Definition 2 (*Relation* $\vdash$)
Consider two constraints $c_1, c_2 \in C$ such that $c_1 = \langle def_1, con\rangle$ and $c_2 = \langle def_2, con\rangle$. The binary relation $\vdash$ is such that $c_1 \vdash c_2$ iff def_2 can be obtained from def_1 by a number (possibly zero) of applications of the rules in Figure 2 .

Theorem 3 (*Relation* $\vdash$ *as entailment relation*)
The binary relation $\vdash$ is an entailment relation.

Proof hint.
Relation $\vdash$ enjoys the reflexivity and transitivity properties that are needed to be an entailment relation. □

In the following, $c^{\vdash}$ represents the reflexive, transitive closure of the entailment relation $\vdash$ applied to the constraint c. While other entailment relations (Bistarelli *et al*., 2002) involve all constraints that are related by the partial order $\leqslant_S$, the security entailment only concerns the subset of those constraints obtainable by application of the four rules in Figure 2.

3.4 The Initial SCSP

The designer of a protocol must also develop a *policy* to accompany the protocol. The policy for a protocol $\mathscr{P}$ is a set of rules stating, among other things, the preconditions necessary for the protocol execution, such as which messages are public, and which messages are private for which principals.

It is intuitive to capture these policy rules by our security levels (section 3.1). Precisely, these rules can be translated into unary constraints. For each principal $A \in \mathscr{V}$, we define a unary constraint that states A's security levels as follows. It associates security level *public* to those messages that are known to all, typically principal names and timestamps; level *private* to A's initial secrets, such as keys (e.g. A's long-term key if $\mathscr{P}$ uses symmetric cryptography, or A's private key if $\mathscr{P}$ uses asymmetric cryptography, or A's pin if $\mathscr{P}$ uses smart cards) or nonces; level *unknown* to all remaining domain values (including, for example, the secrets that A will invent during the protocol execution, or other principals' initial secrets).

This procedure defines what we name *initial SCSP for* $\mathscr{P}$, which specifies the principals' security levels when no session of $\mathscr{P}$ has yet started. Notice that the constraint store representing each principal's security levels is computed using the reflexive, transitive, closure of the entailment relation (section 3.3). So, when a new message is invented, the corresponding constraint is added to the store along with all constraints that can be extracted by entailment.

Considerations on how official protocol specifications often fail to provide a satisfactory policy (Bella *et al*., 2003) exceed the scope of this paper. Nevertheless, having to define the initial SCSP for a protocol may help pinpoint unknown deficiencies or ambiguities in the policy.

BUILDPOLICYSCSP($\mathscr{P}$)

1. $\mathsf{p} \leftarrow$ *initial SCSP for* $\mathscr{P}$;
2. **for** each event *ev* allowed by the policy for $\mathscr{P}$ **do**
3. **if** *ev* = (*A invents n*, for some *A* and *n*) **then**
4. $\mathsf{p} \leftarrow$ p extended with unary constraint on *A* that assigns *private* to *n* and *unknown* to all other messages;
5. **if** *ev* = (*A sends m to B not intercepted*, for some *A*, *m* and *B*) **then**
6. $c \leftarrow Sol(\mathsf{p}) \Downarrow_{\{A\}}$;
7. **let** $\langle def, con \rangle = c^{\vdash}$ **in** *newlevel* $\leftarrow$ RISKASSESSMENT($def(m)$);
8. $\mathsf{p} \leftarrow$ p extended with binary constraint between *A* and *B* that assigns *newlevel* to $\langle \{\!|\}, m \rangle$ and *unknown* to all other tuples;
9. return p;

Fig. 3. Algorithm to construct the policy SCSP for a protocol $\mathscr{P}$.

3.5 The Policy SCSP

The policy for a protocol $\mathscr{P}$ also establishes which messages must be exchanged during a session between a pair of principals while no-one performs malicious activity. The protocol designer typically writes a single step as $A \rightarrow B : m$, meaning that principal *A* sends message *m* to principal *B*. The policy typically allows each principal to participate in a number of protocol sessions inventing a number of fresh messages. Assuming both these numbers to be bounded, a bounded number of *events* may take place (Durgin *et al*., 1999). Because no principal is assumed to be acting maliciously, no message is intercepted, so a message that is sent is certain to reach its intended recipient. Therefore, we only formalise the two following events:

1. A principal invents a fresh message (typically a new nonce).
2. A principal sends a message (constructed by some sequence of applications of encryption, concatenation, decryption, and splitting) to another principal, and the message is delivered correctly.

Clearly, additional events can be formalised to capture protocol-specific details, such as principal's annotation of sensitive messages, message broadcast, SSL-secure trasmission, and so on.

We read from the protocol policy each allowed step of the form $A \rightarrow B : m$ and its informal description, which explains whether *A* invents *m* or part of it. Then, we build the *policy SCSP for* $\mathscr{P}$ by the algorithm in Figure 3.

The algorithm considers the initial SCSP (line 1) and extends it with new constraints induced by each of the events occurring during the protocol execution (line 2). If the current event is a principal *A*'s inventing a message *n* (line 3), then a unary constraint is added on variable *A* assigning security level *private* to the domain value *n*, and *unknown* to all other values (line 4). If that event is a principal *A*'s sending a message *m* to a principal *B* (line 5), then the solution of the current SCSP p is computed and projected on the sender variable *A* (line 6), and extended by entailment (line 7). The last two steps yield *A*'s view of the network traffic. In particular, also *A*'s security level on *m* is updated by entailment. For example, if *m* is built as $\{\!|Na, Nb|\!\}$, the security levels of *Na* and *Nb* derive from the computed solution, and then the level of *m* is obtained by the concatenation rule of the entailment relation.

At this stage, A's security level on m is updated again by algorithm RiskAssessment (line 7). As explained in the next section, this shall assess the risks that m runs following A's manipulation and the exposure to the network. The current SCSP can be now extended with a binary constraint on the pair of variables A and B (line 8). It assigns the newly computed security level *newlevel* to the tuple $\langle\{\}, m\rangle$ and *unknown* to all other tuples. This reasoning is repeated for each of the bounded number of events allowed by the policy. When there are no more events to process, the current SCSP is returned as policy SCSP for $\mathscr{P}$ (step 9), which is our formal model for the idealised protocol. Termination of the algorithm is guaranteed by finiteness of the number of allowed events. Its complexity is clearly linear in the number of allowed events, which is in turn exponential in the length of the exchanged messages (Durgin *et al.*, 1999).

We note that a binary constraint between (a pair of variables formalising respectively) sender and receiver of m, which assigns some newly computed security level to the tuple $\langle\{\}, m\rangle$ confirms that the receiver's level on m is influenced by the event that the constraint formalises, as opposed to the sender's level which is not.

3.6 Assessing the expected risk

Each network event involves some message. The events that expose their messages to the network, such as to send or receive or broadcast a message, clearly impose some *expected risk* on those messages – ideal message security is never to use that message. The *risk function* ρ expresses how the expected risk affects the security levels of the messages that are involved.

The actual definition of the risk function depends on the protocol policy, which should clearly state the expected risk for each network event when the protocol is executed in its intended environment. However, "often protocols are used in environments other than the ones for which they were originally intended" (Meadows, 2001), so the definition also depends on the specific environment that is considered.

The risk function should take as parameters the given security level and the network event that is influencing that level. The second parameter can be omitted for simplicity from this presentation because of the limited number of events we are modelling. Indeed, we will only have to compute the function for the network event whereby a message is sent on the network (either intercepted or not), whereas if we modelled, for example, a broadcast event, then the assessment for that particular event would have to yield *public*.

The risk function must enjoy the two following properties.

(i) *Extensivity*. This property means that $\rho(l) \leqslant l$ for any l. It captures the requirement that each manipulation of a message decreases its security level – each manipulation increases the risk of tampering.

(ii) *Monotonicity*. This property means that $l_1 \leqslant l_2$ implies $\rho(l_1) \leqslant \rho(l_2)$ for any l_1 and l_2. It captures the requirement that the expected risk preserve the $\leqslant$ relation between any pair of given security levels.

```
RiskAssessment(l)
1. let traded_i = l in
2.   if i = n + 1 then l' ← l
3.                 else l' ← traded_{i+1};
4. return l';
```

Fig. 4. Implementation for a simple risk function.

Notice that we have stated no restrictions on the values of the risk function. Therefore, an initial total order, e.g. $l_1 < l_2$, may at times be preserved, such as $\rho(l_1) < \rho(l_2)$, or at other times be hidden, such as $\rho(l_1) = \rho(l_2)$.

As a simple example of risk function we choose the following variant of the predecessor function. It takes a security level and produces its predecessor in the linear order induced by $+_{sec}$ on the set L of security levels, unless the given level is the lowest, *public*, in which case the function leaves it unchanged. Our algorithm RiskAssessment in general serves to implement the risk function. Figure 4 shows the algorithm for our example function.

We note that all considerations we advance in the sequel of this paper merely rely on the two properties we have required for a risk function and are therefore independent from the specific example function. However, the protocol analyser may take, depending on his focus, more detailed risk functions, such as for checking originator(s) or recipient(s) of the current event (conventional principals, trusted third principals, proxi principals, etc.), the network where it is being performed (wired or wireless), and so on.

One could think of embedding the risk function at the constraint level rather than at a meta-level as we have done. That would be possible by embedding the appropriate refinements in the entailment rules. For example, let us consider an agent's construction of a message $m = \{\!|m_1, m_2|\!\}$, which is currently $traded_i$, from concatenation of m_1 and m_2, which are $traded_{i_1}$ and $traded_{i_2}$ respectively. The entailment rule should first compute the maximum between the levels of the components (that is the minimum between the indexes), obtaining $traded_{min(i_1,i_2)}$. Then, it should compute the minimum between the level just computed and that of m (that is the maximum between the indexes), obtaining $traded_{max(min(i_1,i_2),i)}$. Finally, the rule should apply the risk function. With our example risk function, it should yield $traded_{max(min(i_1,i_2),i)+1}$. However, the security levels would be decremented every time the entailment relation were applied. This would violate a general requirement of constraint programming, that is $c^{\vdash} = c^{\vdash\vdash}$. Hence, the decrement at the meta level is preferable.

3.7 The imputable SCSPs

A real-world network history induced by a protocol $\mathscr{P}$ must account for malicious activity by some principals. Each such history can be viewed as a sequence of events of four different forms.

```
BUILDIMPUTABLESCSP(𝒫, nc)
        ⋮
2.  for each event ev in nc do
        ⋮
8.1.  if ev = (A sends m to B intercepted by C, for some A, m, B and C) then
8.2.    c ← Sol(p) ⇓_{A};
8.3.    let ⟨def, con⟩ = c^⊢ in newlevel ← RISKASSESSMENT(def(m));
8.4.    p ← p extended with binary constraint betweeen A and C that assigns
              newlevel to ⟨{}, m⟩ and unknown to all other tuples;
8.5.  if ev = (C cryptanalyses n from m, for some C, m and n) then
8.6.    p ← p extended with unary constraint on C that assigns
              private to n and unknown to all other messages;
        ⋮
```

Fig. 5. Algorithm to construct an imputable SCSP for $\mathscr{P}$ (fragment).

1. A principal invents a fresh message (typically a new nonce).
2. A principal sends a message (constructed by some sequence of applications of encryption, concatenation, decryption, and splitting) to another principal, and the message is delivered correctly.
3. A principal sends a message (constructed as in the previous event) to another principal, but a third principal intercepts it.
4. A principal discovers a message by cryptanalysing another message.

Unlike the first two events, which were formalised also for constructing the policy SCSP, the last two are new, as they are outcome of malicious activity. We remark that the third event signifies that the message reaches some unexpected principal rather than its intended recipient.

We can model any network configuration at a certain point in any real-world network history as an SCSP by modifying the algorithm given in Figure 3 as in Figure 5 (unmodified fragments are omitted). The new algorithm takes as inputs a protocol $\mathscr{P}$ and a network configuration *nc* originated from the protocol execution. The third type of event is processed as follows: when a message is sent by A to B and is intercepted by another principal C, the corresponding constraint must be stated on the pair A, C rather than A, B. The fourth type of event is processed by stating a unary constraint that assigns *private* to the cryptanalyser's security level on the discovered message.

The new algorithm outputs what we name an *imputable SCSP* for $\mathscr{P}$. Both the initial SCSP and the policy SCSP may be viewed as imputable SCSPs. Because we have assumed all our objects to be bounded, the number of possible network configurations is bounded and so is the number of imputable SCSPs for $\mathscr{P}$.

3.8 Formalising confidentiality

"*Confidentiality is the protection of information from disclosure to those not intended to receive it*" (Neuman and Ts'o, 1996). This definition is often simplified into one that is easier to formalise within Dolev and Yao's (1983) model with a single attacker: a message is confidential if it is not known to the attacker. The latter definition is

somewhat weaker: if a principal C who is not the attacker gets to learn a session key for A and B, the latter definition holds but the former does not. To capture the former definition, we adopt the following threat model: *all principals are attackers if they perform,* either deliberately or not*, any operation that is not admitted by the protocol policy*. As we have discussed in the introduction to this paper, our threat model exceeds the limits of Dolev and Yao's by allowing us to analyse scenarios with an unspecified number of non-colluding attackers.

A formal definition of confidentiality should account for the variety of requirements that can be stated by the protocol policy. For example, a message might be required to remain confidential during the early stages of a protocol but its loss during the late stages might be tolerated, as is the case with SET (Bella *et al*., 2003). That protocol typically uses a fresh session key to transfer some certificate once, so the key loses its importance after the transfer terminates.

Another possible requirement is that certain messages, such as those signed by a *root certification authority* to associate the principals to their public keys (Bella *et al*., 2003), be entirely reliable. Hence, at least those messages must be assumed to be safe from cryptanalysis. Also, a protocol may give different guarantees about its goals to different principals (Bella, 2000), so our definition of confidentiality must depend on the specific principal that is considered.

Using the security levels, we develop uniform definitions of confidentiality and of confidentiality attack that account for any policy requirement. Intuitively, if a principal's security level on a message is l, then the message is *l-confidential* for the principal because the security level in fact formalises the principal's trust on the security, meant as confidentiality, of the message (see the beginning of section 3). Thus, if an imputable SCSP features a principal with a lower security level on a message w.r.t. the corresponding level in the policy SCSP, then that imputable SCSP bears a *confidentiality attack*.

Here, l denotes a generic security level, m a generic message, A a generic principal. Also, P indicates the policy SCSP for a generic security protocol, and p and p' some imputable SCSPs for the same protocol. We define $Sol(\mathsf{P}) \Downarrow_{\{A\}} = \langle Def_A, \{A\}\rangle$, $Sol(\mathsf{p}) \Downarrow_{\{A\}} = \langle def_A, \{A\}\rangle$, and $Sol(\mathsf{p}') \Downarrow_{\{A\}} = \langle def'_A, \{A\}\rangle$.

Definition 4 (*l-confidentiality*)
l-confidentiality of m for A in $\mathsf{p} \iff def_A(m) = l$.

3.8.1 Preliminary analysis of confidentiality

The preliminary analysis of the confidentiality goal can be conducted on the policy SCSP for the given protocol.

Let us calculate the solution of the policy SCSP, and project it on some principal A. Let us suppose that two messages m and m' get security levels l and l' respectively, $l' < l$. Thus, even if no principal acts maliciously, m' must be manipulated more than m, so A trusts that m' will be more at risk than m. We can conclude that the protocol achieves a *stronger confidentiality goal on m than on m′* even if it is executed in ideal conditions. Also, m may be used to encrypt m', as is the case with Kerberos (section 5.1), for example. Therefore, losing m to a malicious principal would be

more serious than losing m'. We address a principal's loss of m as *confidentiality attack on m.* A more formal definition of confidentiality attack cannot be given within the preliminary analysis because no malicious activity is formalised. So, the following definition concerns potential confidentiality attacks that may occur during the execution.

Definition 5 (*Potential, worse confidentiality attack*)
Suppose that there is l-confidentiality of m in P for A, that there is l'-confidentiality of m' in P for A, and that $l' < l$; then, *a confidentiality attack on m would be worse than a confidentiality attack on* m'.

3.8.2 Empirical analysis of confidentiality

By an empirical analysis, we consider a specific real-world scenario arising from the execution of a protocol and build the corresponding imputable SCSP p. If the imputable SCSP achieves a weaker confidentiality goal of some message for some principal than the policy SCSP does, then the principal has mounted, either deliberately or not, a confidentiality attack on the message.

Definition 6 (*Confidentiality attack*)
Confidentiality attack by A on m in $\mathsf{p} \iff l$-confidentiality of m in P for $A \ \wedge \ l'$-confidentiality of m in p for $A \ \wedge \ l' < l$.

Therefore, there is a confidentiality attack by A on m in p iff $def_A(m) < Def_A(m)$. The more an attack lowers a security level, the worse that attack, so confidentiality attacks can be variously compared. For example, let us consider two confidentiality attacks by some agent on a message. If the message is l-confidential for the agent in the policy SCSP, but is l'-confidential and l''-confidential respectively in some imputable SCSPs p and p' for the same agent, then $l > l' > l''$ implies that the attack mounted in p' is worse than that in p. Likewise, let us consider two messages m and m' that are both l-confidential for some agent in the policy SCSP. If m is l'-confidential, and m' is l''-confidential in p, then $l > l' > l''$ implies that the attack mounted on m' is worse than that on m.

3.9 Formalising authentication

The authentication goal enforces the principals' presence in the network and possibly their participation in specific protocol sessions. It is achieved by means of messages that "*speak about*" principals. For example, in a symmetric cryptography setting, given a session key Kab relative to the session between principals A and B and known to both, message $\{\!|A, Na|\!\}_{Kab}$ received by B informs him that A is running the session based on nonce Na and key Kab, namely the message authenticates A with B. An equivalent message in an asymmetric setting could be $\{\!|Nb|\!\}_{Ka^{-1}}$, which B can decrypt using A's public key. Also B's mere knowledge of Ka as being A's public key is a form of authentication of A with B. Indeed, A must be a legitimate principal because Ka is typically certified by a certificate of the form $\{\!|A, Ka|\!\}_{K_{ca}}$, K_{ca}

being the public key of a certification authority. It follows that security protocols may use a large variety of message forms to achieve the authentication goal — the ISO standard in fact does not state a single form to use (International Organization for Standardization, 1989).

In consequence, we declare a predicate $speaksabout(m, A)$, but do not provide a formal definition for it because this would necessarily have to be restrictive. However, the examples above provide the intuition of its semantics. There is *l-authentication* of B with A if there exists a message such that A's security level on it is l, and the message speaks about B. This signifies that A received a message conveying B's aliveness.

Definition 7 (*l-authentication*)
l-authentication of B with A in $\mathsf{p} \iff \exists\ m$ s.t. $def_A(m) = l < unknown \wedge speaksabout(m, B) \wedge def_B(m) < unknown$.

The definition says that there is l-authentication of B with A whenever both A and B's security levels on a message that speaks about B are less than unknown, l being A's level on the message. The intuition behind the definition is that messages that B sends A for authentication will produce a strong level of authentication if they reach A without anyone else's tampering. Otherwise the level of authentication gets weaker and weaker. Precisely, the lower A's security level on m, the weaker the authentication of B with A.

Weaker forms of authentication hold when, for example, B sends a message speaking about himself via a trusted third principal, or when a malicious principal overhears the message (recall that each event of sending decreases the security level of the sent message). Our definition applies uniformly to both circumstances by the appropriate security level.

Another observation is that the weakest form, *public*-authentication, holds for example of B with A in an asymmetric-cryptography setting by the certificate for B's public key in any imputable SCSP where A received the certificate. Likewise, an attacker could always forge a *public* message that speaks about B, e.g. a message containing B's identity. But in fact *public*-authentication always holds between any pairs of principals because principals' names are known to all.

3.9.1 Preliminary analysis of authentication

As done with the confidentiality goal (section 3.8.1), the preliminary analysis of the authentication goal can be conducted on the policy SCSP for the given protocol.

Once we calculate the solution of that SCSP, we can apply our definition of l-authentication, and verify what form of authentication is achieved. In particular, if there is l-authentication of B with A, and l'-authentication of D with C, $l' < l$, then we can conclude that the protocol achieves a *stronger authentication goal of B with A, than of D with C*. We address a principal's masquerading as B with A as *authentication attack on A by means of B*. A more formal definition of authentication attack cannot be given at this stage, since no principal acts maliciously in the policy

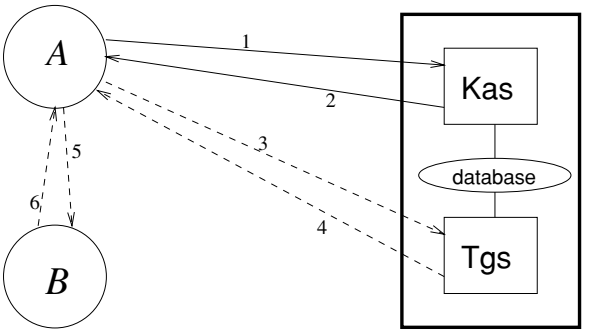

Fig. 6. The Kerberos layout.

SCSP, However, we can compare potential authentication attacks in case they happen during the protocol execution.

Definition 8 (*Potential, worse authentication attack*)
Suppose that there is l-authentication of B with A by m in P, that there is l'-authentication of D with C by m' in P, and that $l' < l$; then *an authentication attack on A by means of B would be worse than an authentication attack on C by means of D*.

3.9.2 Empirical analysis of authentication

If the policy SCSP P achieves l-authentication of B with A by m, and an imputable SCSP p achieves a weaker form of authentication between the same principals by the same message, then the latter SCSP bears an authentication attack.

Definition 9 (*Authentication attack*)
Authentication attack on A by means of B in $\mathsf{p} \iff l$-authentication of B with A in $\mathsf{P} \wedge l'$-authentication of B with A in $\mathsf{p} \wedge \ l' < l$.

If a malicious principal has intercepted a message m that authenticates B with A, and forwarded m to B in some imputable SCSP p, then, according to the previous definition, there is an authentication attack on A by means of B in p.

4 The Kerberos protocol

Kerberos is a protocol based on symmetric cryptography meant to distribute session keys with authentication over local area networks. The protocol has been developed in several variants (e.g. Miller *et al.* (1999)), and also integrated with smart cards (Itoi and Honeyman, 1999). Here, we refer to the version by Bella and Riccobene (1997).

The layout in Figure 6 shows that Kerberos relies on two servers, the *Kerberos Authentication Server* (Kas in brief), and the *Ticket Granting Server* (Tgs in brief). The two servers are trusted, namely they are assumed to be secure from External tampering. They have access to an internal database containing the long-term keys of all principals. The database is in turn assumed to be secure. Only the first two

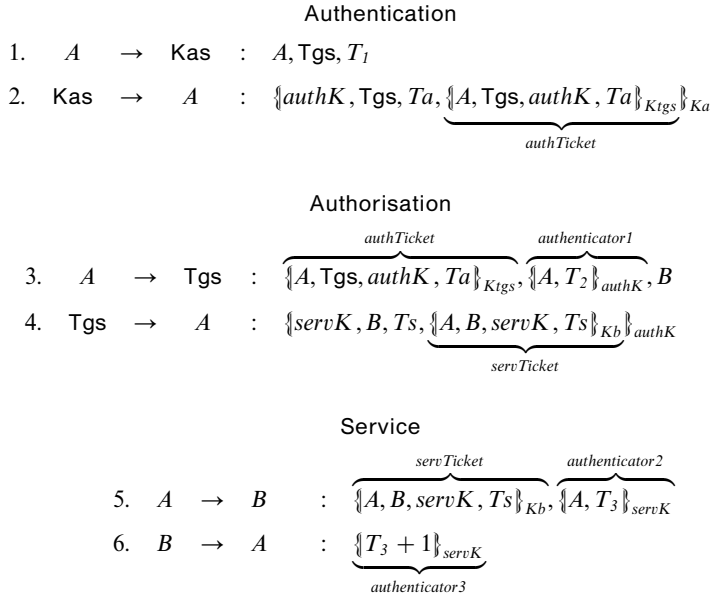

Fig. 7. The Kerberos protocol.

steps of the protocol are mandatory, corresponding to a principal A's authentication with Kas. The remaining steps are optional as they are executed only when A requires access to a network resource B.

In the authentication phase, the initiator A queries Kas with her identity, Tgs and a timestamp T_1; Kas invents a session key and looks up A's shared key in the database. It replies with a message sealed by A's shared key containing the session key, its timestamp Ta, Tgs and a ticket. The session key and the ticket are the credentials to use in the subsequent authorisation phase, so we address them as *authkey* and *authticket* respectively.

Now, A may start the authorisation phase. She sends Tgs a three-component message including the authticket, an authenticator sealed by the authkey containing her identity and a new timestamp T_2, and B's identity. The lifetime of an authenticator is a few minutes. Upon reception of the message, Tgs decrypts the authticket, extracts the authkey and checks the validity of its timestamp Ta, namely that Ta is not too old with respect to the lifetime of authkeys. Then, Tgs decrypts the authenticator using the authkey and checks the validity of T_2 with respect to the lifetime of authenticators. Finally, Tgs invents a new session key and looks up B's shared key in the database. It replies with a message sealed by the authkey containing the new session key, its timestamp Ts, B and a ticket. The session key and the ticket are the credentials to use in the subsequent service phase, so we address them as *servkey* and *servticket* respectively. The lifetime of a servkey is a few minutes.

Hence, A may start the service phase. She sends B a two-component message including the servticket and an authenticator sealed by the servkey containing her identity and a new timestamp T_3. Upon reception of the message, B decrypts the servticket, extracts the servkey and checks the validity of its timestamp Ts. Then, B

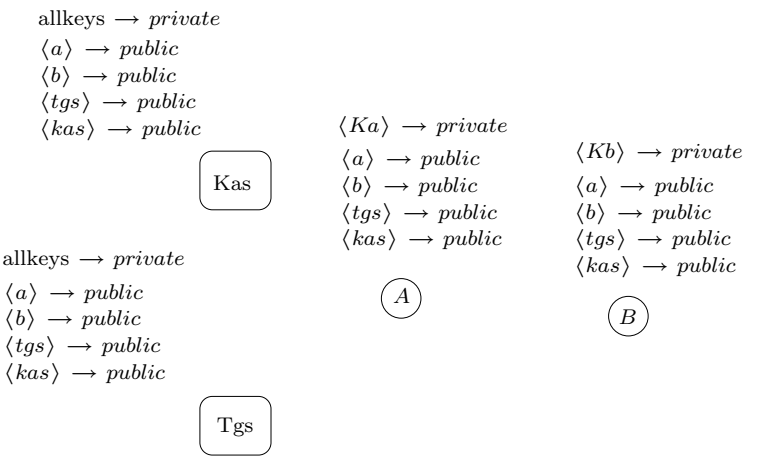

Fig. 8. The initial SCSP for Kerberos (fragment).

decrypts the authenticator using the servkey and checks the validity of T_3. Finally, B increments T_3, seals it by the servkey and sends it back to A.

5 Analysing Kerberos

As a start, we build the initial SCSP for Kerberos. Figure 8 shows the fragment pertaining to principals A and B. The assignment *allkeys* → *private* signifies that the constraint assigns level *private* to all principals' long-term keys.

Then, we build the policy SCSP for Kerberos using algorithm BUILDPOLICYSCSP (Figure 3). Figure 9 shows the fragment pertaining to principals A and B. The components that are specific of the session between A and B, such as timestamps and session keys, are not indexed for simplicity. We remark that the security levels of all other principals on the authkey *authK* and on the servkey *servK* are *unknown*.

5.1 Confidentiality

The preliminary analysis of confidentiality conducted on the policy SCSP in Figure 9 highlights that the late protocol messages get worse security levels than the initial ones do. For example, by definition 4, there is $traded_3$-confidentiality of *servK* for B. By the same definition, it is crucial to observe that A gets *authK* as $traded_1$-confidential, but gets *servK* as $traded_3$-confidential. So, if we consider a potential confidentiality attack whereby A looses *authK* to some malicious principal other than B, and another potential confidentiality attack whereby A or B loose *servK* to some malicious principal, the former would be a worse confidentiality attack than the latter, by definition 5. Indeed, having *authK* available, one can obtain *servK* from decryption and splitting of message 4.

We also conduct an empirical analysis of confidentiality by considering, as example a *known-ciphertext attack* (Stinson, 1995) mounted by some malicious principal C

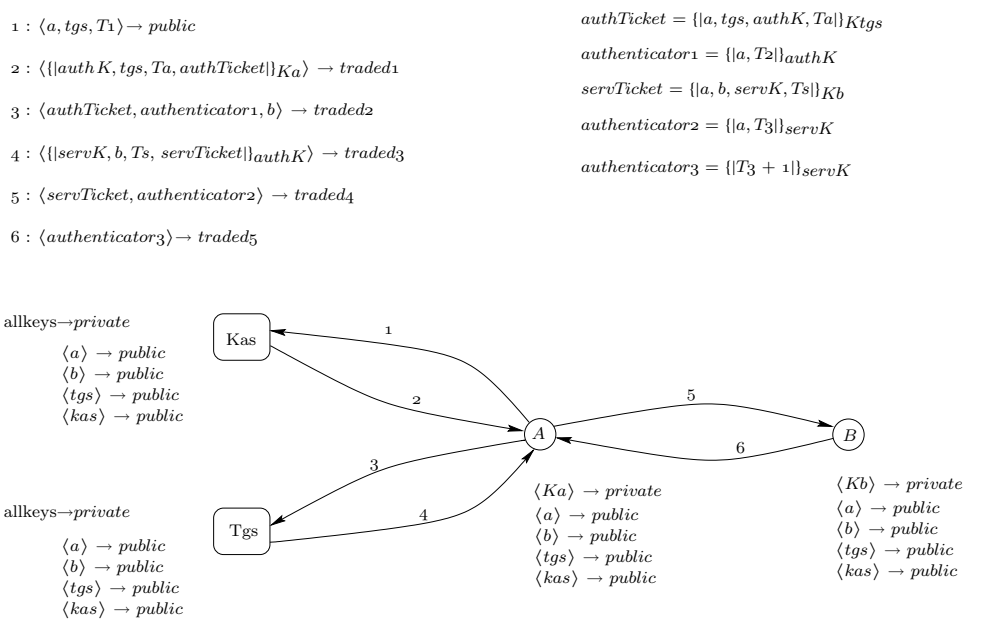

Fig. 9. The policy SCSP for Kerberos (fragment).

on the authenticator of message 3 to discover the authkey pertaining to a principal A (and Tgs). We briefly remind how such attack works. Since both principal names and timestamps are public, C knows the body of the authenticator with a good approximation – she should just try out all timestamps of, say, the last day. First, she invents a key, encrypts the known body with it, and checks whether the result matches the encrypted authenticator fetched from the network. If not, C "refines" her key (Stinson, 1995) and iterates the procedure until she obtains the same ciphertext as the authenticator. At this stage, she holds the encryption key, alias the authkey, because encryption is injective. The entire tampering took place off line.

Along with the authkey for A, principal C also saves a copy of the corresponding authticket by splitting message 3 into its components. Then, C forwards message 3, unaltered, to Tgs, so A can continue and terminate the session accessing some resource B. A glimpse to Figure 7 shows that C is now in a position to conduct, for the lifetime of the authkey, the Authorisation and Service phases while he masquerades as A with some principal D. To do so, C forges an instance $3'$ of message 3 by using the authticket just learnt, by refreshing the timestamp inside the authenticator (which he can do because he knows the authkey), and by mentioning the chosen principal D. As Tgs believes that the message comes from A, Tgs replies to A with a message $4'$ containing some fresh servkey meant for A and D. Having intercepted $4'$, C learns the servkey and therefore can forge an instance $5'$ for D of message 5. Finally, C intercepts $6'$ and the session terminates without A's participation.

$3 : \langle authTicket, authenticator_1, b\rangle \rightarrow traded_2$
$\bar{3} : \langle authTicket, authenticator_1, b\rangle \rightarrow traded_3$
$4 : \langle \{|servK, b, Ts, servTicket|\}_{authK}\rangle \rightarrow traded_3$

$3' : \langle authTicket, authenticator_1', d\rangle \rightarrow traded_3$
$4' : \langle \{|servK', d, Ts', servTicket'|\}_{authK}\rangle \rightarrow traded_4$
$5' : \langle servTicket', authenticator_2'\rangle \rightarrow traded_5$
$6' : \langle authenticator_3'\rangle \rightarrow traded_6$

$authTicket = \{|a, tgs, authK, Ta|\}_{Ktgs}$
$authenticator_1 = \{|a, T_2|\}_{authK}$
$servTicket = \{|a, b, servK, Ts|\}_{Kb}$

$servTicket' = \{|a, d, servK', Ts'|\}_{Kd}$
$authenticator_1' = \{|a, T_2'|\}_{authK}$
$authenticator_2' = \{|a, T_3'|\}_{servK'}$
$authenticator_3' = \{|T_3'+_1|\}_{servK'}$

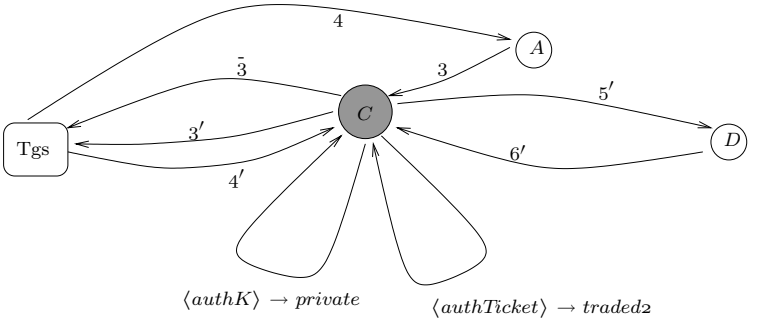

Fig. 10. An imputable SCSP for Kerberos (fragment).

Our algorithm BUILDIMPUTABLESCSP executed on the network configuration just described produces the imputable SCSP in Figure 10. The SCSP omits the constraint corresponding to the Authentication phase between A and Kas. Because C intercepts message 3, constraint 3 is stated between A and C. Projecting that constraint on C, we have that C's security level on message $authTicket, authenticator1, b$ is $traded_2$. By splitting this message, C discovers the authticket, so the entailment relation states a unary constraint on C assigning $traded_2$ to $authTicket$. Another unary constraint on C assigns *private* to $authK$, which is found by cryptanalysis.

Constraint $\bar{3}$ between C and Tgs assigns $traded_3$ to message 3 because of C's rerouting. Projecting that constraint on Tgs, we have by entailment that Tgs's security level on $authK$ goes down to $traded_3$, whereas it was $traded_2$ in the policy SCSP. Constraint 4 formalises Tgs's reply to A, while the constraints for the rest of the session between A and B are omitted. Constraints $3'$, $4'$, $5'$, and $6'$ formalise the session between C, Tgs, and D.

At this stage, we can conduct an empirical analysis of confidentiality for each of the agents involved in this imputable SCSP. By definition 4, $authTicket$, $authenticator1'$, and $authK$ are each $traded_3$-confidential for Tgs in this problem. Since they were $traded_2$-confidential in the policy SCSP, we conclude by definition 6 that there is a confidentiality attack by Tgs on each of these messages in the imputable SCSP considered here. The attacks signal C's manipulation of messsage 3.

The imputable SCSP also achieves *private*-confidentiality of $authK$ for C, whereas the policy SCSP achieved *unknown*-confidentiality of $authK$ for C. Therefore, there is a confidentiality attack by C on $authK$ in this SCSP. Likewise, there is a confidentiality attack by C on $authTicket$. From constraint $4'$ we have by entailment

that C's security level on $servTicket'$ and on $servK'$ is $traded_4$ rather than *unknown* as in the policy SCSP, hence we find other confidentiality attacks by C on each of these messages.

There are also confidentiality attacks by D, who gets $servTicket'$, $authenticator2'$, and $servK'$ as $traded_5$, rather than $traded_4$.

5.2 Authentication

We now focus on the fragment of policy SCSP for Kerberos given in Figure 9 to conduct the preliminary analysis of the authentication goal.

By definition 7, there is $traded_2$-authentication of A with Tgs in the policy SCSP. The definition holds for message 3, whose first two components speak about A. Also, there is $traded_4$-authentication of A with B thanks to message 5, and $traded_5$-authentication of B with A due to message 6. While it is obvious that message 5 speaks about A, it is less obvious that message 6 speaks about B. This is due to the use of a servkey that is associated to B.

We observe that authentication of B with A is weaker than authentication of A with B even in the ideal conditions formalised by the policy SCSP. Intuitively, this is due to the fact that the servkey has been handled both by A and B rather than just by A. Hence, by definition 8, a principal C's masquerading as A with B would be a worse authentication attack than a principal D's masquerading as B with A.

An empirical analysis of authentication can be conducted on the imputable SCSP in Figure 10. That SCSP achieves $traded_5$-authentication of A with B thanks to message 5, and $traded_6$-authentication of B with A due to message 6. Comparing these properties with the equivalent ones holding in the policy SCSP, which we have seen above, we can conclude by Definition 9 that the imputable SCSP considered hides an authentication attack on B by means of A, and an authentication attack on A by means of B. They are due to C's interception of message 3, which has lowered the legitimate protocol participants' security levels on the subsequent messages.

It is important to emphasize that these authentication attacks could not be captured by an equivalent definition of authentication based on crisp, rather than soft, constraints. The definition in fact holds in the policy SCSP as well as in the imputable SCSP. What differentiates the two SCSPs is merely the security level characterising the goal.

6 Conclusions

We have developed a new framework for analysing security protocols, based on a recent kernel (Bella and Bistarelli, 2001, 2002). Soft constraint programming allows us to conduct a fine analysis of the confidentiality and authentication goals that a protocol attempts to achieve. Using the security levels, we can formally claim that a configuration induced by a protocol achieves a certain level of confidentiality or authentication. That configuration may be ideal if every principal behaves according to the protocol, as formalised by the policy SCSP; or, it may arise from the protocol execution in the real world, where some principal may have acted maliciously, as

formalised by an imputable SCSP. We can formally express that different principals participating in the same protocol session obtain different forms of those goals. We might even compare the forms of the same goal as achieved by different protocols.

Our new threat model where each principal is a potential attacker working for his own sake has allowed us to detect a novel attack on the asymmetric Needham-Schroeder protocol. Once C masquerades as A with B, agent B indeliberately gets hold of a nonce that was not meant for him. At this stage, B might decide to exploit this extra knowledge, and begin to act maliciously. Our imputable SCSP modelling the scenario reveals that B's security level on the nonce is lower than that allowed by the policy.

There is some work related to our analysis of Kerberos, such as Mitchell et al.'s analysis by model checking (Mitchell *et al.*, 1997). They consider a version of Kerberos simplified of timestamps and lifetimes – hence authkeys and servkeys cannot be distinguished – and establish that a small system with an initiator, a responder, `Kas` and `Tgs` keeps the two session keys secure from the Dolev-Yao attacker. Bella and Paulson (1998) verify by theorem proving a version with timestamps of the same protocol. They do prove that using a lost authkey will let the spy obtain a servkey. On top of this, one can informally deduce that the first key is more important than the second in terms of confidentiality. By contrast, our preliminary analysis of the protocol states formally that the authkey is $traded_1$-confidential and the servkey is $traded_3$-confidential (section 5.1). Another finding is the difference between authentication of initiator with responder and vice versa (section 5.2).

Some recent research exists that is loosely related to ours. Millen and Shamatikov (2001) map the existence of a *strand* representing the attack upon a constraint problem. Comon *et al.* (2001) and Amadio and Charatonik (2002) solve confidentiality and reachability using *Set-Based Constraint* (Pacholski and Podelski, 1997). By constrast, we build suitable constraint problems for the analysis of a global network configuration where any principals (not just one) can behave maliciously. In doing so, we also analyse the safety of the system in terms of the consequences of a deliberate attack on the environment. The idea of *refinements* (De Schreye *et al.*, 1995) is also somewhat related to our use of levels. In that case the original protocol must be specialised in order to be able to map the known/unknown level over the set of levels specified by the policy. The policy has also to specify how the levels have to be changed w.r.t. each operation described in the protocol. Abstract interpretation techniques (much in the spirit of those used by Bistarelli *et al.* (2002)) can be used as a next step to deal with unbounded participants/sessions/messages.

While mechanical analysis was outside our aims, we have implementated a mechanical checker for l-confidentiality on top of the existing *Constraint Handling Rule* (CHR) framework (Bistarelli *et al.*, 2002). For example, when we input the policy SCSP for the Needham-Schroeder protocol and the imputable SCSP corresponding to Lowe's attack, the checker outputs

```
checking(agent(a))
checking(agent(b))
   attack(n_a, policy_level(unknown), attack_level(traded_1))
```

```
checking(agent(c))
   attack(enk(k(a),pair(n_a,n_b)), policy_level(unknown),
                                   attack_level(traded_1))
   attack(n_b, policy_level(unknown), attack_level(traded1))
```

The syntax seems to be self-explanatory. Line two reveals the new attack we have found on B, who has lowered his security level on Na from *unknown* to $traded_1$. Likewise, line three denounces that not only has C got hold of the nonce Nb but also of the message $\{\!|Na, Nb|\!\}_{Ka}$ (which was meant for A and not for B) that contains it.

At this stage, integrating our framework with model-checking tools appears to be a straightforward exercise. The entailment relation can be extended by a rule per each of the protocol messages in order to compute their security levels. Hence, our constraints would be upgraded much the way multisets are rewritten in the work by Cervesato *et al.* (1999) (though they only focus on a single attacker and their properties are classical yes/no properties). Then, once suitable size limits are stated, the imputable SCSPs could be exhaustively generated and checked against our definitions of confidentiality and authentication. Alternatively, the protocol verifier might use our framework for a finer analysis of network configurations generated using other techniques.

Acknowledgements

We are indebted to Martin Abadi for invaluable suggestions, and to Michael Marte and Thom Fruewirth for the preliminary implementation of the checker. Discussions with Luca Viganò, Nancy Durgin and Fabio Massacci were important. Comments by the anonymous referees helped us improve the presentation considerably.

References

Abadi, M. 1997. Secrecy by typing in security protocols. In *Proc. TACS 1997*, 611–638.

Abadi, M. 1999. Secrecy by typing in security protocols. *Journal of the ACM 46,* 5 (September), 749–786.

Abadi, M. and Gordon, A. D. 1997. A calculus for cryptographic protocols: The spi calculus. In *Proceedings ACM Conference on Computer and Communications Security 1997*, pp. 36–47.

Abadi, M. and Needham, R. M. 1996. Prudent Engineering Practice for Cryptographic Protocols. *IEEE Transactions on Software Engineering 22,* 1, 6–15.

Amadio, R. M. and Charatonik, W. 2002. On name generation and set-based analysis in dolev-yao model. Research Report 4379, INRIA.

Anderson, R. 1993. Why Cryptosystems Fail. In *Proceedings of the 1st ACM Conference on Communications and Computer Security (CCS'93)*. ACM Press & Addison Wesley, pp. 217–227.

Bella, G. 2000. Inductive Verification of Cryptographic Protocols. PhD thesis, Research Report 493, Computer Laboratory, University of Cambridge. Available on the Internet.

Bella, G. and Bistarelli, S. 2001. Soft Constraints for Security Protocol Analysis: Confidentiality. In *Proc. of the 3rd International Symposium on Practical Aspects of Declarative Languages (PADL'01)*. LNCS 1990. Springer-Verlag, pp. 108–122.

Bella, G. and Bistarelli, S. 2002. Confidentiality levels and deliberate/indeliberate protocol attacks. In *Proc. Security Protocols 10th International Workshop, Cambridge, UK, April, 2002, Revised Papers*. LNCS 2845. Springer-Verlag, pp. 104–119.

Bella, G., Massacci, F. and Paulson, L. 2003. Verifying the SET Registration Protocols. *IEEE Journal of Selected Areas in Communications 1,* 21, 77–87.

Bella, G. and Paulson, L. C. 1998. Kerberos Version IV: Inductive Analysis of the Secrecy Goals. In *Proc. of European Symposium on Research in Computer Security (ESORICS'98)*. LNCS 1485. Springer-Verlag, pp. 361–375.

Bella, G. and Riccobene, E. 1997. Formal Analysis of the Kerberos Authentication System. *Journal of Universal Computer Science 3,* 12, 1337–1381.

Bistarelli, S. 2001. Soft constraint solving and programming: a general framework. PhD thesis, Dipartimento di Informatica, Università di Pisa.

Bistarelli, S., Codognet, P. and Rossi, F. 2002. Abstracting soft constraints: Framework, properties, examples. *Artificial Intelligence 139,* 2, 175–211.

Bistarelli, S., Fruewirth, T., Marte, M. and Rossi, F. 2002. Soft constraint propagation and solving in chr. In *Proc. ACM Symposium on Applied Computing (SAC), Madrid, Spain*. ACM.

Bistarelli, S., Montanari, U. and Rossi, F. 1997. Semiring-based Constraint Solving and Optimization. *Journal of the ACM*, 201–236.

Bistarelli, S., Montanari, U. and Rossi, F. 2001. Semiring-based Constraint Logic Programming: Syntax and Semantics. *ACM Transactions on Programming Languages and System (TOPLAS) 23*, 1–29.

Bistarelli, S., Montanari, U. and Rossi, F. 2002. Soft concurrent constraint programming. In *Proc. ESOP, Grenoble, France*. LNCS, vol. 2305. Springer, pp. 53–67.

Bodei, C., Degano, P., Nielson, F. and Nielson, H. R. 1996. Security analysis using flow logics. In *Current Trends in Theoretical Computer Science*. IEEE Press, pp. 525–542.

Bodei, C., Degano, P., Nielson, F. and Nielson, H. R. 2001. Static analysis for secrecy and non-interference in networks of processes. In *Proceedings PaCT 2001*, pp. 27–41.

Cervesato, I., Durgin, N. A., Lincoln, P., Mitchell, J. C. and Scedrov, A. 1999. A meta-notation for protocol analysis. In *Proceedings CSFW*, pp. 55–69.

Comon, H., Cortier, V. and Mitchell, J. 2001. Tree automata with one memory, set constraints and ping-pong protocols. In *Proc. 28th Int. Coll. Automata, Languages, and Programming (ICALP'2001)*. LNCS, vol. 2076. Springer-Verlag, pp. 682–693.

De Schreye, D., Leuschel, M., and Martens, B. 1995. Tutorial on program specialisation. In *Proc. of ILPS'95, the International Logic Programming Symposium, Portland, USA*, J. Lloyd, Ed., pp. 615–616.

Denning, D. E. 1976. A lattice model of secure information flow. *Comm. of ACM 19,* 5, 236–242.

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

Dubois, D., Fargier, H. and Prade, H. 1993. The Calculus of Fuzzy Restrictions as a Basis for Flexible Constraint Satisfaction. In *Proceedings IEEE International Conference on Fuzzy Systems*. IEEE Press, pp. 1131–1136.

Durgin, N., Lincoln, P., Mitchell, J. and Scedrov, A. 1999. Undecidability of bounded security protocols. In *Proceedings FMSP'99*. http://cm.bell-labs.com/cm/cs/who/nch/fmsp99/durgin.ps.

Fargier, H. and Lang, J. 1993. Uncertainty in Constraint Satisfaction Problems: a Probabilistic Approach. In *Proceedings European Conference on Symbolic and Qualitative Approaches to Reasoning and Uncertainty (ECSQARU)*. Springer-Verlag, pp. 97–104.

Focardi, R., Gorrieri, R. and Martinelli, F. 2000a. Information flow analysis in a discrete-time process algebra. In *Proc. CSFW 2000*. 170–184.

Focardi, R., Gorrieri, R. and Martinelli, F. 2000b. Non interference for the analysis of cryptographic protocols. In *Proceedings ICALP 2000*, pp. 354–372.

Foley, S. 2002. Personal conversations.

Freuder, E. C. and Wallace, R. J. 1992. Partial constraint satisfaction. *AI Journal 58*, 21–70.

Gray, E. 2001. American national standard T1.523-2001, telecom glossary 2000. published on the Web at http://www.its.bldrdoc.gov/projects/telecomglossary2000.

International Organization for Standardization. 1989. *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture*. ISO 7498-2.

Itoi, N. and Honeyman, P. 1999. Smartcard Integration with Kerberos V5. In *Proceedings USENIX Workshop on Smartcard Technology*.

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

Lowe, G. 1996. Some New Attacks upon Security Protocols. In *Proceedings Computer Security Foundations Workshop (CSFW96)*. IEEE Press, pp. 139–146.

Meadows, C. 2001. Private conversations.

Millen, J. and Shmatikov, V. 2001. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings 8th ACM Conference on Computer and Communication Security*. ACM, pp. 166–175.

Miller, S. P., Neuman, J. I., Schiller, J. I. and Saltzer, J. H. 1989. Kerberos Authentication and Authorisation System. Technical Plan Sec. E.2.1, MIT - Project Athena.

Mitchell, J. C., Mitchell, M. and Stern, U. 1997. Automated Analysis of Cryptographic Protocols Using Murphi. In *Proceedings IEEE Symposium on Security and Privacy*, pp. 141–153.

Neuman, B. C. and Ts'o, T. 1996. Kerberos: An authentication service for computer networks, from IEEE Communications Magazine, September, 1994. In *William Stallings, Practical Cryptography for Data Internetworks*. IEEE Press.

Pacholski, L. and Podelski, A. 1997. Set constraints: A pearl in research on constraints (invited tutorial). In *Proc. Principles and Practice of Constraint Programming*, G. Smolka, Ed. Vol. 1330. Springer-Verlag, pp. 549–562.

Paulson, L. C. 1998. The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security 6*, 85–128.

Ruttkay, Z. 1994. Fuzzy Constraint Satisfaction. In *Proc. of 3rd IEEE International Conference on Fuzzy Systems*, pp. 1263–1268.

Saraswat, V. A. 1993. *Concurrent Constraint Programming*. MIT Press.

Schiex, T. 1992. Possibilistic Constraint Satisfaction Problems, or "How to Handle Soft Constraints?". In *Proc. of 8th Conference on Uncertainty in AI*, pp. 269–275.

Schiex, T., Fargier, H. and Verfaille, G. 1995. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In *Proceedings 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*. Morgan Kaufmann, pp. 631–637.

Stinson, D. R. 1995. *Cryptography Theory and Practice*. CRC Press.