Consiglio Nazionale delle Ricerche

Soft Constraint Programming to Analysing Security Protocols

G. Bella, S. Bistarelli

IAT B4-13/2001

Technical Report

Novembre 2001

Istituto per le Applicazioni Telematiche

Soft Constraint Programming to Analysing Security Protocols

TR IAT-B4-2001-013

Giampaolo Bella and Stefano Bistarelli

[1] Universitị Catania, Dipartimento di Matematica e Informatica,
Viale A. Doria 6, I-95125 Catania, Italy.
giamp@dmi.unict.it
[2] CNR Pisa, Istituto per le Applicazioni Telematiche
Via G. Moruzzi, 1 I-56124 Pisa, Italy.
Stefano.Bistarelli@iat.cnr.it

Abstract. Security protocols stipulate how 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. 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 [BB01], we have recently discovered a new attack on that protocol. We briefly describe that attack, and demonstrate the framework on a bigger, largely deployed protocol consisting of three phases, Kerberos.

1 Introduction

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 [AN96,Low96], so a number of approaches for analysing them formally have been developed [Low95,BR97,Pau98,Bel99]). 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.

However, both goals are a lot more complicated than the intuition just given. Focusing on confidentiality, we remark that different messages require "specific degrees of protection against disclosure" [Gra01]. 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 the protocol in order to sign the message.

It is somewhat surprising that confidentiality and authentication are formalised in a mere "yes or no" fashion in the existing literature, so one can just say whether a key is confidential or not, or whether a principal authenticates himself with another or not. The motivation for our research was the development of a finer formal notion for the various forms of the two goals. We have developed the notions of *l-confidentiality* and of *l-authentication*, where l is the *security level* signifying the strength with which the goal is met. The security level belongs to the career set of a semiring, as our notions rest on semiring-based soft constraint programming. 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. In the following, we indicate by $\{\!|m|\!\}_K$ the ciphertext obtained encrypting message m with key K, and avoid external brackets of concatenated messages. We assume the reader to be familiar with the basic concepts of cryptography.

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. These outlines suggest that our notation is uniform: there is not a single attacker, but *all principals are attackers if they perform, either deliberately or not, some operation that is not admitted by the protocol policy.* For example, Lowe's popular attack on the asymmetric Needham-Schroeder protocol [Low95] follows from a malicious principal C's masquerading 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 [Low95, §3]

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

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

Applying our framework to the same scenario, we have discovered a new confidentiality attack whereby B unintentionally learns A's nonce Na for C. This attack may have dangerous consequences on C in case B realises, possibly much later, what Na is. Readapting Lowe's example, we observe that, 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. Our empirical analysis has highlighted uniformly 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 didn't act maliciously.

Our framework, which extends and supersedes an existing kernel [BB01], is mature to be demonstrated on a largely deployed protocol, Kerberos. Our preliminary analysis of the protocol highlights that the loss of an *authorisation key* would be more serious than the loss of a *service key*, and that the authentication of the protocol responder with the initiator is somewhat weaker than the authentication of the initiator with the responder. Our empirical analysis focuses on how cryptanalysis undermines the protocol goals. We remark that, since we only deal with unbounded but finite quantities, our framework is amenable to mechanisation by model checking, although this exceeds the purposes of this paper.

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

2 Soft constraints

Several formalisations of the concept of *soft constraints* are currently available [SFV95,DFP93,FW92,FL93]. In the following, we refer to one that is based on c-semirings [BMR95,BMR97,Bis01], which can be shown to generalise and express many of the others [BFM$^+$96,BFM$^+$99].

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, $\mathbf{1}$ as its absorbing element and $\times$ is commutative.

Let us consider the relation $\leq_S$ over A such that $a \leq_S b$ iff $a + b = b$. Then it is possible to prove that (see [BMR97]):

- $\leq_S$ is a partial order;

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

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

Informally, 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, $a \leq_S b$ 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 $def : \mathcal{D}^{|con|} \rightarrow A$. 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 problem* 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*.

Notice that a classical CSP is a SCSP where the chosen c-semiring is:

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

Fuzzy CSPs [DFP93,Rut94,Sch92] can instead be modelled in the SCSP framework by choosing the c-semiring:

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

Figure 1 shows a fuzzy CSP. Variables are inside circles, constraints are represented by undirected arcs, and semiring values are written to the right of the corresponding tuples. Here we assume that the domain $\mathcal{D}$ of the variables contains only elements a and b.

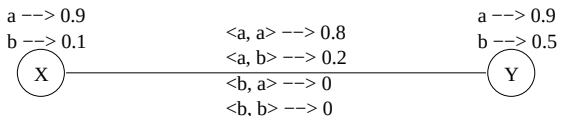

Fig. 1. A fuzzy CSP.

2.1 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 $\mathcal{V}$, 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 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.

2.2 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.

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* (§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) level *unknown* to the least secure (lowest) 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* (§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 (§3.3), that is an entailment relation between constraints. Then, given a specific protocol to analyse, we represent its assumptions in the *initial SCSP* (§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* (§3.5). We also explain how to represent any network configuration arising from the protocol execution in the real world as an *imputable SCSP* (§3.6).

Given a security level l, establishing whether our definitions of *l-confidentiality* (§3.7) or *l-authentication* (§3.8) hold in an SCSP requires calculating the solution of the imputable SCSP and projecting it on certain principals. 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 attacks 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. 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 from the imputable is lower than the corresponding level from the policy, then there is an attack in the imputable SCSP. In fact, some malicious activity contributing to the network configuration modelled by the imputable SCSP has taken place so to lower some security level allowed by the policy SCSP.

The following, general treatment is demonstrated in §4.

3.1 The Security Semiring

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

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

Although our security levels may appear to resemble Abadi's *types* [Aba97,Aba99], there is in fact little similarity. Abadi associates each message to either type *public*, or *secret*, or *any*, whereas we define an unbounded number of security levels, 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 unbounded number of $traded_i$ levels, the four levels that we have presented elsewhere [BB01].

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:
– $unknown \equiv traded_{-1}$
– $private \equiv traded_0$
– $public \equiv traded_{n+1}$

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

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

Proof hint. *Clearly,* $\mathcal{S}_{sec}$ *enjoys the same properties as the structure* $S_{finite-fuzzy} = \langle \{-1, \ldots, n+1\}, min, max, , -1, n+1 \rangle$*. 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 min;* $\times_{sec}$ *can be mapped into function max. Moreover,* $S_{finite-fuzzy}$ *can be proved a c-semiring as done with the fuzzy semiring [BMR97].*

3.2 The Network Constraint System

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

– $\mathcal{S}_{sec}$ is the security semiring (§3.1);
– $\mathcal{V}$ is an unbounded set of variables.
– $\mathcal{D}$ is an unbounded 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 $\mathcal{V}$ stand for the network principals, and the elements of $\mathcal{D}$ represent all possible messages. Atomic messages typically are principal names, timestamps, nonces and cryptographic

keys. Concatenation and encryption operations can be applied an unbounded number of times.

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

Encryption:

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

Concatenation:

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

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:

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

Fig. 2. Computation rules for security levels.

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 1 (Security entailment). *Consider two constraints $c_1, c_2 \in C$ such that $c_1 = \langle def_1, con\rangle$ and $c_2 = \langle def_2, con\rangle$. We say that c_1* entails *c_2, and write $c_1 \vdash c_2$, iff def_2 can be obtained from def_1 by applying 0, 1 or more times the rules in Figure 2 . We name $\vdash$ as* security entailment*.*

Theorem 2 (Security entailment). *The relation $\vdash$ from Definition 1 is an entailment relation.*

Proof hint. *The relation $\vdash$ enjoys the reflexivity and transitivity properties.*

3.4 The Initial SCSP

The designer of a protocol must also develop a *policy* to accompany the protocol. The policy for a protocol $\mathcal{P}$ 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 (§3.1). Precisely, these rules can be translated into unary constraints. For each principal $A \in \mathcal{V}$, 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 $\mathcal{P}$ uses symmetric cryptography, or A's private key if $\mathcal{P}$ uses asymmetric cryptography, or A's pin if $\mathcal{P}$ uses smart cards) or nonces; level *unknown* to all remaining domain values (including, e.g., the secrets that A will invent during the protocol execution, or other principals' initial secrets).

This procedure defines what we name *initial SCSP for $\mathcal{P}$*, which specifies the principals' security levels when no session of $\mathcal{P}$ 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 (§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 [BMPT00] 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.

3.5 The Policy SCSP

The policy for a protocol $\mathcal{P}$ 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 unbounded (not infinite), an unbounded number of *events* may take place [DLMS99][1]. These events consist of principals' inventing fresh messages (typically new nonces) and principals' sending messages constructed by concatenation and/or encryption. No message is intercepted because no malicious principal is assumed to be active: A's sending m to B implies that B receives it.

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* $\mathcal{P}$ by the algorithm in figure 3. The algorithm adds new constraints to the initial SCSP according to the event that is considered. If that event is a principal A's inventing a message n, then a unary constraint is added on variable A assigning security level *private* to the domain value n, and *unknown* to all other values. If that event is a principal A's sending a message m to a principal B, then the solution of the current SCSP is computed and projected on the sender variable A. The semiring value, alias security level, associated to message m over A is considered. This level is computed by entailment (§3.3) whenever m is obtained by manipulation of other messages (rather than m being e.g. a fresh nonce just invented with security level *private* by the previous case of the algorithm). Sending m on the network exposes it to risks, hence we compute a new security level for it, which is one level lower in the total order of L (§3.1) unless m is already *public*. Our choice of decrementing the security level by one is generic and not restrictive. If specific policies required a different choice, our algorithm could be trivially modified accordingly.

A binary constraint that assigns the newly computed security level to the tuple $\langle \{\!|\}\!|, m \rangle$ and *unknown* to all other tuples is now added to the current SCSP on the pair of variables A and B. Incidentally, we remark that the constraint store used to compute the solution is updated by entailment every time a new constraint is added, namely every time a principal invents a new message or sends a message computed by concatenation and/or encryption. This reasoning is repeated for each of the unbounded number of events allowed by the policy.

[1] If either number is infinite, then we may have infinite events, and protocol security is undecidable [DLMS99].

When there are no more events to process, the current SCSP is returned as policy SCSP for $\mathcal{P}$, which is our formal model for the protocol.

```
BUILD_POLICY_SCSP(P)
 1. p ← initial SCSP for P;
 2. for each event ev allowed by the policy for P do
 3.   if ev = (A invents n, for some A and n) then
 4.     p ← 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.     let ⟨def, con⟩ = Sol(p) ⇓_{A}  ∧  def(m) = traded_i in
 7.       if i = n + 1 then newlevel ← public
                       else newlevel ← traded_{i+1};
 8.       p ← p extended with binary constraint between A and B that
              assigns newlevel to ⟨{|}, m⟩ and unknown to all other tuples;
 9. return p;
```

Fig. 3. Algorithm to construct the policy SCSP for $\mathcal{P}$.

Termination of the algorithm is guaranteed by finiteness of the number of allowed events. The algorithm is clearly linear in the number of allowed events, which is in turn exponential in the length of the exchanged messages [DLMS99]. No correctness result can be stated of the algorithm as it merely builds a formal model from an informal one. This is a limitation of formal analysis in general, which is always offset by the adherence of the formal model to reality. According to Börger, that adherence should merely arise by *inspection* [B̈99], and we believe this is the case here. However, as with Ryan and Schneider's analysis, "our models are, like all mathematical models, only ever approximations to reality" [RS00].

As we have observed, decrementing the security level on a message when it is sent signifies that the more the message is manipulated the higher its risks. That also adds uniformity to our treatment of confidentiality attacks: if a principal merely intercepts a message m, the principal's security level on m will be worse than that stated by the policy SCSP, a condition that we will capture as an attack (§3.7).

3.6 The Imputable SCSP

A real-world network history induced by a protocol $\mathcal{P}$ must account for malicious activity by some principals. Each such history can be viewed as a sequence of events of the forms: a principal's inventing new messages, a principal's sending messages that are not intercepted, and a principal's sending messages that are intercepted. While the second event signifies that the intended recipient of a message indeed gets the message, the third signifies that some malicious principal prevents the delivery of the message that is sent.

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 4 (unmodified fragments are omitted). The new algorithm takes as inputs a protocol $\mathcal{P}$ and a network configuration nc originated from the protocol execution. The processing of the third type of event is added: 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.

BUILD_IMPUTABLE_SCSP($\mathcal{P}$, 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. **let** $\langle def, con\rangle = Sol(\mathsf{p}) \Downarrow_{\{A\}}$ $\wedge$ $def(m) = traded_i$ **in**
8.3. **if** $i = n + 1$ **then** $newlevel \leftarrow public$
else $newlevel \leftarrow traded_{i+1}$;
8.4. $\mathsf{p} \leftarrow \mathsf{p}$ extended with binary constraint betweeen A and C that
assigns $newlevel$ to $\langle \{\!|\}, m\rangle$ and $unknown$ to all other tuples;
⋮

Fig. 4. Algorithm to construct an imputable SCSP for $\mathcal{P}$ (fragment).

The new algorithm outputs what we name an *imputable SCSP* for $\mathcal{P}$. Clearly, there exist an unbounded number of imputable SCSPs for $\mathcal{P}$, each representing a different network configuration. Both the initial SCSP and the policy SCSP may be viewed as imputable SCSPs.

3.7 Formalising Confidentiality

A message is confidential if it is not known to an attacker. We regard as attacker any principal other than the recipients that the protocol policy recommends for the message.

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 [BMPT00]. 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 [BMPT00], 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, 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 §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 2 (l-confidentiality). *l-confidentiality of m in p for A* $\iff$ $def_A(m) = l$.

A preliminary analysis of confidentiality A 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 (§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 a preliminary analysis because no malicious activity is formalised. So, the following definition concerns potential confidentiality attacks that may occur during the execution

Definition 3 (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'.

An 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 4 (Confidentiality attack).
Confidentiality attack by A on m in p $\Longleftrightarrow$ 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 brevity, we define only two possible forms of comparison.

Definition 5 (Worse confidentiality attack on messages). *Suppose that there is a confidentiality attack on m in p, and that there is a confidentiality attack on m' in p such that $def_A(m) = def_A(m') = l$, and $Def_A(m) < def_A(m')$; then p* bears a worse confidentiality attack on m' than on m.

Definition 6 (Worse confidentiality attack in SCSPs). *Suppose that there is a confidentiality attack by A on m in p such that $def_A(m) = l$, that there is a confidentiality attack by A on m in p' such that $def'_A(m) = l'$, and that $l < l'$; then p* bears a worse confidentiality attack on m than p' does.

3.8 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 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 [Int89].

In consequence, we declare a predicate $speaks\_about(m, A)$, 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, and is known to B. This signifies that B sent a message to A and A received it.

Definition 7 (l-authentication).
l-authentication of B with A in p $\Longleftrightarrow$ $\exists \ m$ s.t. $def_A(m) = l < unknown \wedge speaks\_about(m, B) \ \wedge \ def_B(m) < unknown$.

Notice that the lower A's security level on m, the weaker authentication. For example, in an asymmetric-cryptography setting, there is *public*-authentication

of B with A by the certificate for B's public key in any imputable SCSP where A has received that certificate. Also, the spy could easily send a *public* message that speaks about B (e.g. a message containing B's identity), hence *public*-authentication is very weak. Our definition also holds when B sends his message that speaks about A via a trusted third principal.

A preliminary analysis of authentication As done with the confidentiality goal (§3.7), a 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 SCSP, However, we can compare potential authentication attacks that may 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.

An 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 p $\Longleftrightarrow$ l-authentication of B with A in P $\wedge$ l'-authentication of B with A in 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. [MNSS89]), and also integrated with smart cards [IH99]. Here, we refer to the version by Bella and Riccobene [BR97].

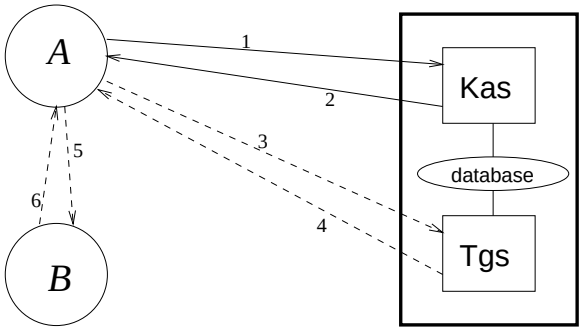

Fig. 5. The Kerberos layout.

The layout in figure 5 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 the spy's 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 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.

Authentication

1. $A \rightarrow \mathsf{Kas} : A, \mathsf{Tgs}, T1$
2. $\mathsf{Kas} \rightarrow A : \{\!|authK, \mathsf{Tgs}, Ta, \underbrace{\{\!|A, \mathsf{Tgs}, authK, Ta|\!\}_{Ktgs}}_{authTicket}|\!\}_{Ka}$

Authorisation

3. $A \rightarrow \mathsf{Tgs} : \overbrace{\{\!|A, \mathsf{Tgs}, authK, Ta|\!\}_{Ktgs}}^{authTicket}, \overbrace{\{\!|A, T2|\!\}_{authK}}^{authenticator}, B$
4. $\mathsf{Tgs} \rightarrow A : \{\!|servK, B, Ts, \underbrace{\{\!|A, B, servK, Ts|\!\}_{Kb}}_{servTicket}|\!\}_{authK}$

Service

5. $A \rightarrow B : \overbrace{\{\!|A, B, servK, Ts|\!\}_{Kb}}^{servTicket}, \overbrace{\{\!|A, T3|\!\}_{servK}}^{authenticator}$
6. $B \rightarrow A : \{\!|T3 + 1|\!\}_{servK}$

Fig. 6. The Kerberos protocol.

In the authentication phase, the initiator A queries Kas with her identity, Tgs and a timestamp $T1$; 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 $T2$, 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 $T2$ 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 $T3$. Upon reception of the message, B decrypts the servticket, extracts the servkey and checks the validity of its timestamp Ts. Then, B decrypts the authenticator using the servkey and checks the validity of $T3$. Finally, B increments $T3$, 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 7 shows the fragment pertaining to principals A and B. The assignment $all\_keys \rightarrow private$ signifies that the constraint assigns level *private* to all principals' long-term keys.

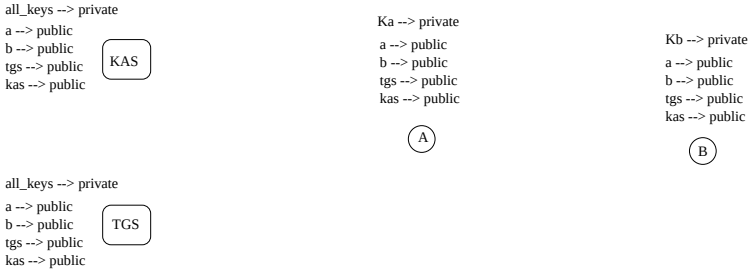

Fig. 7. The initial SCSP for Kerberos (fragment).

Then, we build the policy SCSP for Kerberos using algorithm Build_Policy_SCSP (figure 3). Figure 8 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*.

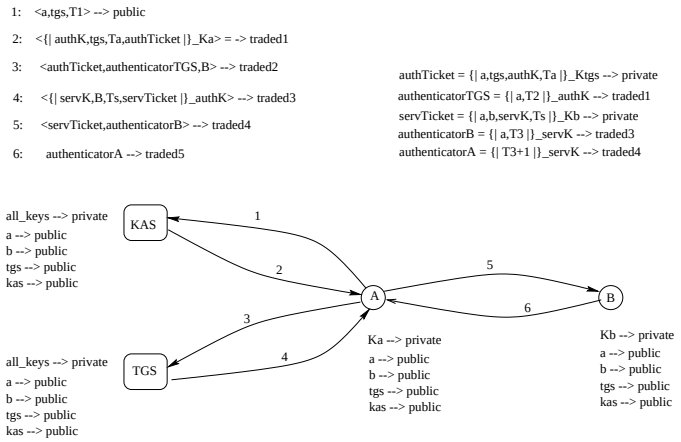

Fig. 8. The policy SCSP for Kerberos (fragment).

5.1 Confidentiality

A preliminary analysis of confidentiality conducted on the policy SCSP in figure 8 highlights that the late protocol messages get worse security levels than the initial ones do. So, let us 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. Since A's security level on *authK* is $traded_1$, and both A and B's security levels on *servK* are $traded_3$, the former would be a worse confidentiality attack than the latter, by definition 3. Indeed, having *authK* available, one can obtain *servK* from decryption and splitting of message 4.

We can also conduct an empirical analysis by considering, for example, a *known-cleartext attack* [RSA76] mounted by some malicious principal C on the authenticator of message 3 to discover the authkey. (In brief, such an attack works as follows. 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 [RSA76] 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.)

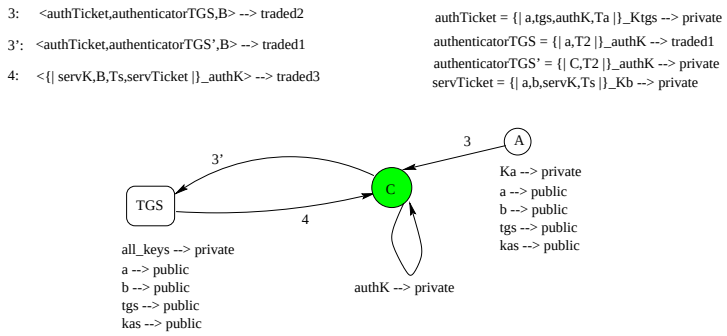

Fig. 9. The imputable SCSP for an attack on an authkey (fragment).

Figure 9 presents the imputable SCSP corresponding to the mentioned attack. This SCSP achieves *private*-confidentiality of *authK* for C, whereas the policy SCSP achieved *unknown*-confidentiality of *authK* for C. Therefore, by definition 4, there is a confidentiality attack by C on *authK* in the imputable SCSP considered here. Notice that the key being *private* for C signifies that C did not discover it because of a weakness of the protocol, otherwise the key would have been $traded_i$ for some i. Therefore, our notation also captures known-cleartext attacks, which are due to weaknesses of the underlying cryptosystem.

Following the attack, C can forge an instance of message 3, and mislead Tgs into thinking that the authkey is for use with C rather than with A.

5.2 Authentication

We now focus on the fragment of policy SCSP for Kerberos given in figure 8 to conduct a preliminary analysis of the authentication goal. In particular, we study the authentication goals achieved between the pair of principals A and B.

By definition 7, there is $traded_4$-authentication of A with B in the policy SCSP. The definition holds for message 5, which speaks about A because it mentions A. Likewise, there is $traded_5$-authentication of B with A in the policy SCSP. The definition holds for message 6, which speaks about B because it uses the 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.

Once we are given an authentication attack, we can build the corresponding imputable SCSP, and conduct an empirical analysis of authentication (omitted here), as described in §3.8.

6 Conclusions

We have developed a new framework for analysing security protocols, based on a recent kernel [BB01]. 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. 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.

In relation to our work, we mention Mitchell et al.'s analysis by model checking [MMS97]. They consider a version of Kerberos simplified of timestamps and lifetimes — hence authkeys and servkeys cannot be distinguished — establishing that a small system with an initiator, a responder, Kas and Tgs keeps the two session keys secure from the spy. Bella and Paulson [BP98] 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 (§5.1). Another finding is the difference between authentication of initiator with responder and vice versa (§5.2).

Our analysis is amenable to mechanisation by model checking: we could conduct an empirical analysis on all imputable SCSPs, which are in a finite number [DLMS99]. These could be generated by an inductive definition whereby a principal *tells* a new constraint at each step. The imputable SCSPs would be each other similar, so their solutions could be computed efficiently using existing tools [Geo99,GC98].

References

[Aba97] Martín Abadi. Secrecy by typing in security protocols. In *14th Symposium on Theoretical Aspects of Computer Science (STACS'97)*, Lecture Notes in Computer Science. Springer-Verlag, 1997.

[Aba99] Martín Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 46(5):749–786, September 1999.

[AN96] M. Abadi and R. M. Needham. Prudent Engineering Practice for Cryptographic Protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, 1996.

[Bö99] E. Börger. High level system design and analysis using abstract state machines. In D. Hutter, W. Stephan, P. Traverso, and M. Ullman, editors, *Current Trends in Applied Formal Methods (FM-Trends'98)*, volume 1641 of *LNCS*, pages 1–43. Springer-Verlag, 1999.

[BB01] G. Bella and S. Bistarelli. Soft Constraints for Security Protocol Analysis: Confidentiality. In *Proc. of the 3rd International Symposium on Practical Aspects of Declarative Languages (PADL'01)*, Lecture Notes in Computer Science. Springer-Verlag, 2001. In press.

[Bel99] G. Bella. Modelling Security Protocols Based on Smart Cards. In *Proc. of the International Workshop on Cryptographic Techniques & E-Commerce (CrypTEC'99)*, pages 139–146. City University of Hong Kong, 1999.

[BFM+96] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and Valued CSPs: Basic Properties and Comparison. In *Over-Constrained Systems*. Springer-Verlag, 1996.

[BFM+99] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based csps and valued csps: Frameworks, properties, and comparison. *CONSTRAINTS: An international journal. Kluwer*, 4(3), 1999.

[Bis01] S. Bistarelli. *Soft Constraint Solving and Programming: a general framework*. PhD thesis, Dipartimento di Informatica - Università di Pisa, 2001.

[BMPT00] G. Bella, F. Massacci, L. C. Paulson, and P. Tramontano. Formal Verification of Cardholder Registration in SET. In *Proc. of European Symposium on Research in Computer Security (ESORICS 2000)*, LNCS. Springer-Verlag, 2000. In press.

[BMR95] S. Bistarelli, U. Montanari, and F. Rossi. Constraint Solving over Semirings. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*. Morgan Kaufman, 1995.

[BMR97] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Solving and Optimization. *Journal of the ACM*, pages 201–236, 1997.

[BP98] G. Bella and L. C. Paulson. Kerberos Version IV: Inductive Analysis of the Secrecy Goals. In *Proc. of European Symposium on Research in Computer Security (ESORICS'98)*, volume 1485 of *LNCS*, pages 361–375. Springer-Verlag, 1998.

[BR97] G. Bella and E. Riccobene. Formal Analysis of the Kerberos Authentication System. *Journal of Universal Computer Science*, 3(12):1337–1381, 1997.

[DFP93] D. Dubois, H. Fargier, and H. Prade. The Calculus of Fuzzy Restrictions as a Basis for Flexible Constraint Satisfaction. In *Proc. of IEEE International Conference on Fuzzy Systems*, pages 1131–1136. IEEE Press, 1993.

[DLMS99] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In N. Heintze and E. Clarke, editors, *Proc. FMSP'99*, 1999.

[FL93] H. Fargier and J. Lang. Uncertainty in Constraint Satisfaction Problems: a Probabilistic Approach. In *Proc. of European Conference on Symbolic and Qualitative Approaches to Reasoning and Uncertainty (ECSQARU)*, pages 97–104. Springer-Verlag, 1993.

[FW92] E. C. Freuder and R. J. Wallace. Partial constraint satisfaction. *AI Journal*, 1992.

[GC98] Y. Georget and P. Codognet. Compiling semiring-based constraints with clp(fd,s). In *Proc. CP98*, number 1520 in LNCS. Springer-Verlag, 1998.

[Geo99] Y. Georget. *Extensions de la Programmation par Contraintes*. PhD thesis, Ecole Polytecnique, Paris, 1999.

[Gra01] Evelyn Gray. American national standard t1.523-2001, telecom glossary 2000. published on the Web at http://www.its.bldrdoc.gov/projects/telecomglossary2000, 2001.

[IH99] N. Itoi and P. Honeyman. Smartcard Integration with Kerberos V5. In *Proceedings of the USENIX Workshop on Smartcard Technology*, 1999.

[Int89] International Organization for Standardization. *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture*. ISO 7498-2, 1989.

[Low95] G. Lowe. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Information Processing Letters*, 56(3):131–133, 1995.

[Low96] G. Lowe. Some New Attacks upon Security Protocols. In *In Proc. of Computer Security Foundations Workshop (CSFW96)*, pages 139–146. IEEE Press, 1996.

[MMS97] J. C. Mitchell, M. Mitchell, and U. Stern. Automated Analysis of Cryptographic Protocols Using Murphi. In *Proceedings of the 16th IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1997.

[MNSS89] S. P. Miller, J. I. Neuman, J. I. Schiller, and J. H. Saltzer. Kerberos Authentication and Authorisation System. Technical Plan Sec. E.2.1, MIT - Project Athena, 1989.

[Pau98] L. C. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security*, 6:85–128, 1998.

[RS00] P. Y. A. Ryan and S. A. Schneider. *The Modelling and Analysis of Security Protocols: The CSP Approach*. Addison-Wesley, 2000. In press.

[RSA76] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1976.

[Rut94] Zs. Ruttkay. Fuzzy Constraint Satisfaction. In *Proc. of 3rd IEEE International Conference on Fuzzy Systems*, pages 1263–1268, 1994.

[Sch92] T. Schiex. Possibilistic Constraint Satisfaction Problems, or "How to Handle Soft Constraints?". In *Proc. of 8th Conference on Uncertainty in AI*, pages 269–275, 1992.

[SFV95] T. Schiex, H. Fargier, and G. Verfaille. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 631–637. Morgan Kaufmann, 1995.