# SCSPs for Modelling Attacks to Security Protocols

Giampaolo Bella[1,2] and Stefano Bistarelli[3]

[1] Technische Universität München, Fakultät für Informatik, Arcistr. 31, D-80290 München, Germany.
Email: `Bella@in.tum.de`

[2] University of Cambridge, Computer Laboratory, Pembroke Street, Cambridge CB2 3QG, England.
Email: `Giampaolo.Bella@cl.cam.ac.uk`

[3] Università di Pisa, Dipartimento di Informatica, Corso Italia 40, I-56125 Pisa, Italy.
Email: `Bista@di.unipi.it`

**Abstract.** *Security protocols* are executed over an *insecure* network by a finite but unbounded population of agents. We show that any network configuration arising from the execution of a specific protocol can be modelled as a soft constraint satisfaction problem (SCSP). In particular, the *policy SCSP* models the network configuration where all admissible protocol sessions have terminated successfully. A network configuration suffers an attack if the solution of the corresponding SCSP is *lower* than that for the policy SCSP. Different attacks bear more or less drastic consequences, hence soft constraints are necessary. For the sake of demonstration, we model the asymmetric Needham-Schroeder protocol, and the attack discovered by Lowe on that protocol.
**Keywords:** soft constraint, security protocol, security level, protocol attack.

## 1 Introduction

Modern computer networks are *insecure* in the sense that the ongoing traffic can be intercepted and eavesdropped by a malicious attacker, called *spy* below. Agents trying to communicate on an insecure network execute suitable *security protocols* to take advantage of the protocol *goals*. A typical goal is *confidentiality*, which holds of a message that remains undisclosed to the spy. Failure to achieve the claimed goals of a protocol [AN96,Low96,LR97] has motivated a number of approaches for reasoning formally on security protocols (e.g. [Low95,BR97,Pau98,Bel99]).

Our original contribution to the field is an approach to specify each network configuration arising from the execution of a protocol as a soft constraint satisfaction problem (SCSP), and to detect attacks mounted by the spy. Recall that an SCSP may be viewed as a classical constraint satisfaction problem (CSP) [Mac92,Wal96] where each assignment of values to variables in the constraints is associated to an element taken from a partially ordered set. These elements can then be interpreted as levels of preference, or of certainty, etc.

When modelling security protocols, the partially ordered set contains *security levels*. For example, the Kerberos protocol [BP98] relies on two separate sets of session keys: one contains the "authorisation keys", the other contains the "service keys". Each authorisation key *authK* is used to encrypt several service keys $servK_i$, hence, if confidentiality is violated on *authK*, it will be also violated on all the $servK_i$. Clearly, different levels of security must be associated to different violations of confidentiality. We demonstrate our approach on the *asymmetric* Needham-Schroeder protocol [NS78], which is one of the the most discussed security protocols. Despite its apparent simplicity, the protocol hides a subtle but realistic attack, which was discovered by Lowe [Low95] nearly two decades after the protocol publication.

We assume a basic familiarity with the concepts of encryption and decryption, which are extensively presented elsewhere [Nat77,RSA76]. We indicate encryption by fat braces, so that $\{\!|m|\!\}_K$ represents the ciphertext obtained by encrypting message $m$ under key $K$. We say that encryption is **perfect** when $m$ can be recovered from $\{\!|m|\!\}_K$ if and only if $K^{-1}$ is available. In this definition, the keys $K$ and $K^{-1}$ are interchangeable. A major feature is that $K$ cannot be obtained from $K^{-1}$ or vice versa. While most cryptographic algorithms are perfect at the specification level, several of their implementations no longer enjoy that desirable property. Designing a protocol that achieves the goals for which it was conceived is not trivial even if perfect cryptography is available, as confirmed by Lowe's attack.

There are many formalisations of SCSPs. We consider the one based on semirings [BMR97,BMR95], where the semiring specifies the partially ordered set and the appropriate operation that must be used to combine constraints together. Below, we briefly review the basics of SCSPs (§2), and then describe how to use them to model attacks to security protocols (§3). Then, we describe the asymmetric Needham-Schroeder protocol (§4), use our approach to model that protocol (§5), and conclude (§6).

## 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 semirings [BMR95,BMR97], 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 $a$*.

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 $con$.

Note 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 $B$.

**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 $def(t) = def_1(t \downarrow^{con}_{con_1}) \times def_2(t \downarrow^{con}_{con_2})$, where $t \downarrow^{X}_{Y}$ denotes the tuple of values over the variables in $Y$, obtained by projecting tuple $t$ from $X$ to $Y$. In words, combining two constraints means building a new constraint involving all the variables of the original ones, and

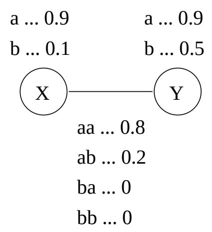


**Fig. 1.** A fuzzy CSP.

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.

**Solutions** The *solution* of an SCSP problem $P = \langle C, con\rangle$ is the constraint $Sol(P) = (\bigotimes C) \Downarrow_{con}$. 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.

Sometimes it may be useful to find the semiring value of the optimal solutions. This is called the *best level of consistency* of an SCSP problem $P$ and it is defined by $blevel(P) = Sol(P) \Downarrow_\emptyset$. We also say that: $P$ is $\alpha$-consistent if $blevel(P) = \alpha$; $P$ is consistent iff there exists $\alpha >_S \mathbf{0}$ such that $P$ is $\alpha$-consistent; $P$ is inconsistent if it is not consistent.

## 3 Using SCSPs for Modelling Protocol Attacks

The intuition underlying the machinery developed below is to express protocol attacks as properties of the solutions for appropriate soft constraint problems.

Verifying whether there is an attack on a given network configuration arising from the execution of a protocol will reduce to solving a constraint problem and analyse its solution. Given a security protocol, we aim at modelling each possible network configuration as a constraint problem over a specific constraint system — hence, a semiring is required. The following, general treatment is demonstrated in §5.

### 3.1 The Security Semiring

We define the set $L$ as to contain *unknown*, *private*, *debug* and *public*. Each of these elements represents a possible *security level* that a protocol associates to messages. The security levels regulate the agents' knowledge of messages. They may be seen as an extension of the two-valued property of known/unknown. For example, *unknown* will be assigned to those messages that a protocol does not encompass, and obviously to those messages that a given agent does not know; *debug* will be assigned to the messages that are created and then exchanged during a protocol session. The remaining levels are self-explanatory.

Figure 2 defines a multiplicative operator, $\times_{sec}$, and an additive one, $+_{sec}$. The following theorem introduces the *security semiring*.

| $\times_{\text{sec}}$ | *unknown* | *private* | *debug* | *public* |
|---|---|---|---|---|
| *unknown* | *unknown* | *private* | *debug* | *public* |
| *private* | *private* | *private* | *debug* | *public* |
| *debug* | *debug* | *debug* | *debug* | *public* |
| *public* | *public* | *public* | *public* | *public* |

| $+_{\text{sec}}$ | *unknown* | *private* | *debug* | *public* |
|---|---|---|---|---|
| *unknown* | *unknown* | *unknown* | *unknown* | *unknown* |
| *private* | *unknown* | *private* | *private* | *private* |
| *debug* | *unknown* | *private* | *debug* | *debug* |
| *public* | *unknown* | *private* | *debug* | *public* |

**Fig. 2.** The definitions of $\times_{sec}$ and $+_{sec}$

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

*Proof. There exists an isomorphism between the fuzzy semiring (§2) and* $\mathcal{S}_{sec}$*: the security levels can be mapped into the values in the range* $[0, 1]$ *(unknown being mapped into* 1*, public being mapped into* 0*),* $+_{sec}$ *can be mapped into function max, and* $\times_{sec}$ *into function min.*

Since $\times_{sec}$ is idempotent, it is also the *glb* operator in the total order of the carrier set $L$ of the secrecy levels. While the current four levels will suffice to model most protocols, it is understood that more complex protocols may require additional ones, such as *notice*, *warning*, *error*, *crit*. The security semiring can be easily extended by upgrading the definitions of $\times_{sec}$ and $+_{sec}$.

### 3.2 The Network Constraint System

A computer network can be modelled as a constraint problem over a constraint system $CS_n = \langle \mathcal{S}, \mathcal{D}, \mathcal{V} \rangle$ defined as follows:

- $\mathcal{S}$ is the security semiring (§3.1);
- $\mathcal{V}$ is a finite but unbounded set of variables, including $SPY$, each representing a network agent;
- $\mathcal{D}$ is a finite but unbounded set of values including all atomic messages, as well as all messages recursively obtained by concatenation and encryption. Intuitively, $\mathcal{D}$ represents all agents' possible knowledge.

Atomic messages typically are agent names, timestamps, nonces and cryptographic keys. For $\mathcal{D}$ to be finite, we consider an unbounded number of atomic messages. Also, each agent can only open an unbounded number of protocol sessions with any other agent, and so can only employ an unbounded number of nonces. Similarly, timestamps are drawn from an unbounded interval of time, and the number of concatenation and encryption operations that may be applied to build $D$ is unbounded. $D$ also includes the empty message $\{\!|\,|\!\}$.

We name $CS_n$ as *network constraint system*. Note 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 The Initial SCSP

Each security protocol $\mathcal{P}$ is associated with a policy that should, at least, state

- which messages are public;
- which messages are private for which agents.

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 agent $A \in \mathcal{V}$, we define a unary constraint that states the security levels of $A$'s knowledge as follows. It associates security level *public* to all agent names and to timestamps (if $\mathcal{P}$ uses them); level *private* to $A$'s initial secrets,[1] such as keys (i.e. $Ka$ if $\mathcal{P}$ uses symmetric encryption, or $Ka^{-1}$ if it uses asymmetric encryption) or nonces; level *unknown* to all remaining domain values (including, e.g., other agents' initial secrets, or $A$'s secrets created during the protocol execution).

This procedure defines what we name *initial SCSP for* $\mathcal{P}$, which portrays a network where $\mathcal{P}$ can be executed but none of its sessions has started yet. The

[1] As opposed to the secrets created during the protocol execution.

initial SCSP for $\mathcal{P}$ also highlights the agents' initial knowledge, which typically consists of atomic messages such as agent names, timestamps, some keys and some nonces.

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 pinpoint unknown deficiencies or ambiguities in the policy.

### 3.4 The Policy SCSP

The policy for a protocol $\mathcal{P}$ also establishes which messages must be exchanged during a session between a given pair of agents.

We extend the initial SCSP for $\mathcal{P}$ (§3.3) with further constraints. Each step of a session of $\mathcal{P}$ between any pair of agents can be translated into, at most, two constraints. Precisely, for each protocol step whereby $A$ sends a message $m$ to $B$, the following rules must be followed.

$R_1$) If $A$ issues a new secret $n$ (i.e. typically a new nonce) and uses it to build $m$, then add a unary constraint on $A$ that assigns security level *debug* to $n$, and level *unknown* to all remaining messages.

$R_2$) Add a binary constraint between $A$ and $B$ that assigns security level *debug* to the tuple $\langle \{\!|\!\}, m \rangle$, and level *unknown* to all other possible tuples.

The unary constraint advanced by $R_1$ corresponds to $A$'s off-line creation of $m$, while the binary constraint stated by $R_2$ corresponds to $A$'s sending $m$ to $B$, and $B$'s receiving $m$. The two rules yield what we name *policy SCSP for* $\mathcal{P}$. This SCSP formalises a network where each agent has successfully terminated a finite number of protocol sessions with every other agent, while the spy has performed no malicious activity.

### 3.5 The Imputable SCSP

A finite network history induced by a protocol $\mathcal{P}$ may be viewed as a repeated sequence of three steps in various order: agents' creating, sending and receiving messages. However, in the real world, not all messages that are sent are then received by the intended recipient or received at all. This is due to the malicious activity of the spy. Hence, to model the configuration of the network at a certain point in a possible history as an SCSP, we need a variation of rule $R_2$ (§3.4) in order to allow for the malicious activity of the spy. So, for each protocol step whereby $A$ sends a message $m$ to $B$, we constrain the initial SCSP for $\mathcal{P}$ (§3.3) additionally, as stated by rule $R_1$ and by the following variation of rule $R_2$.

$R'_2$) If $C$ receives $m$, then add a binary constraint between $A$ and $C$ that assigns security level *debug* to the tuple $\langle \{\!|\!\}, m \rangle$, and level *unknown* to all other possible tuples.

Note that $C$ could either be $B$ (in case the spy as not interfered) or be the spy (no other agent can act maliciously). In particular, should the spy, in the latter case, also deliver $m$ to $B$, the rule quoted above would apply to the triple *SPY*, $m$ and $B$. Or, should the spy tamper with $m$ obtaining $m'$, and then deliver it to another agent $D$, then the rule would apply to the triple *SPY*, $m'$, and $D$.

We name the originated soft constraint problem as *imputable SCSP* for the network of agents running protocol $\mathcal{P}$. There exist a finite number of imputable SCSPs for $\mathcal{P}$ because the number of agents, the number of protocol steps, and the length of a network history up to a certain point are all finite values.

### 3.6 Agent knowledge as an enhanced *Security Entailment*

The agents' knowledge increases dynamically while they are running a protocol. This can be represented using a language like *cc* [Sar89] suitably extended to deal with soft constraints[2]. The extension relies on the notion of $\alpha$-consistency, rather than on the notion of consistency/inconsistency. Moreover, the entailment relation $\vdash$ [Sco82], which captures from the store the constraints implied but not explicitly added by the agents, must be modified to deal with the semiring values.

The entailment relation on the *network constraint system* models directly the possible operations — message encryption, decryption, concatenation and splitting — that the agents may perform on the messages that they see. The agents' knowledge increases by means of these operations.

The definition of the entailment relation must be augmented by the following rules, where the *def* function is associated to a generic constraint over a generic agent $A$.

**Encryption**

$$\begin{aligned}&\mathit{def}(m_1) = v_1, \quad \mathit{def}(m_2) = v_2, \quad \mathit{def}(\{\!|m_1|\!\}_{m_2}) = v\\ &\quad\vdash\ \mathit{def}(\{\!|m_1|\!\}_{m_2}) = (v \times_{sec} (v_1 +_{sec} v_2))\end{aligned}$$

If $A$ knows two messages $m_1$ and $m_2$, then he also knows the encrypted message $\{\!|m_1|\!\}_{m_2}$. Precisely, if $m_1$ and $m_2$ have respectively security level $v_1$ and $v_2$, then encryption bears level $(v_1 +_{sec} v_2)$; if $A$ knows already $\{\!|m_1|\!\}_{m_2}$ with security level $v$, then this must be accounted for by means of $\times_{sec}$.

**Decryption**

$$\begin{aligned}&\mathit{def}(m_1) = v_1, \quad \mathit{def}(m_2) = v_2, \quad \mathit{def}(\{\!|m_1|\!\}_{m_2}) = v\\ &\quad\vdash\ \mathit{def}(m_1) = (v_1 \times_{sec} v), \quad \mathit{def}(m_2) = (v_2 \times_{sec} v)\end{aligned}$$

An agent $A$ can decrypt a ciphertext and obtain its body, whose security level comes from the application of $\times_{sec}$ to the levels of the ciphertext and of the encryption key.

[2] For a more detailed treatment and general views of the SCSP framework see [Bis00]

**Concatenation**

$$def(m_1) = v_1, \quad def(m_2) = v_2, \quad def(\{\!\mid m_1, m_2 \mid\!\}) = v$$
$$\vdash \; def(\{\!\mid m_1, m_2 \mid\!\}) = (v \times_{sec} (v_1 +_{sec} v_2))$$

The security level for compound messages is computed as that for encrypted messages.

**Splitting**

$$def(m_1) = v_1, \quad def(\{\!\mid m_1, m_2 \mid\!\}) = v \quad \vdash \quad def(m_1) = (v_1 \times_{sec} v)$$

Once a component is extracted from a compound message, its security level must be normalised with that already owned by the agent.

Recall that, in the initial SCSP for any protocol, all domain values for all agents are associated with security level *unknown*, which is the unit element of $\times_{sec}$.

### 3.7 Modelling Protocol Attacks

Let us consider a security protocol $\mathcal{P}$. Our aim is to decide whether a given configuration of the network that arises when some agents execute some sessions of $\mathcal{P}$ hides an attack.

First, we build the policy SCSP for $\mathcal{P}$ and the imputable SCSP for the given configuration. Then, we calculate the solutions of both SCSPs. Finally, if the solution of the imputable SCSP is *lower* (that is close to security level *public*) than that for the policy SCSP, then and only then the given configuration has an attack.

In particular, the solution of the imputable SCSP being lower (than that for the policy SCSP) signifies that some messages have a lower security level. The attack can be understood by focusing on those messages.

It is straightforward to prove that the above procedure for modelling attacks is sound and complete. Indeed, all attacks rely on the spy's intercepting and learning certain messages, which lowers the security levels of the spy for those messages (typically from *unknown* to *debug*, §3.6). Vice versa, if this happens, then the spy has got hold of messages that the policy of the protocol does not allow her to. Hence, there is an attack.

## 4 The Asymmetric Needham-Schroeder Protocol

The "asymmetric" attribute signifies that the protocol makes use of an asymmetric cryptographic algorithm (e.g. RSA [RSA76]) rather than a symmetric one (e.g. DES [Nat77]). Each agent $A$ is endowed with a *public* key $Ka$, which is known to all, and a *private* key $Ka^{-1}$, which should be known only to $A$. Note that limiting the knowledge of $Ka^{-1}$ only to $A$ is an assumption explicitly required by the protocol, rather than a property that is enforced.

The protocol version that is presented in figure 3 omits the initial phase where a trusted certification authority makes it public that the bit sequences $Ka$ and $Kb$, are respectively the public keys for agents $A$ and $B$. That phase is is irrelevant to the following discussion. Recall that a *nonce* is a "number that is used only once" [NS78]. The protocol assumes that agents can invent *truly-random* nonces, so that, given a nonce $N$ invented by an agent $P$, the probability that agents other than $P$ guess $N$ is negligible.

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

**Fig. 3.** The asymmetric Needham-Schroeder protocol

The first step sees an *initiator* $A$ initiate the protocol with a *responder* $B$. $A$ issues a nonce $Na$ and encrypts it along with her identity under $B$'s public key. Upon reception of that message, $B$ decrypts it and extracts $A$'s nonce. Then, he issues a nonce $Nb$ and encrypts it along with $Na$ under $A$'s public key. When $A$ receives message 2, she extract $Nb$ and sends it back to $B$, encrypted under his public key.

The goal of the protocol is *authentication*: at completion of a protocol session initiated by $A$ with $B$, $A$ should get evidence to have communicated with $B$ and, likewise, $B$ should get evidence to have communicated with $A$. An informal analysis would probably confirm that the goal is met. Upon reception of $Na$ inside message 2, $A$ would conclude that she is interacting with $B$, the only agent who could retrieve $Na$ from message 1, since $Na$ is a truly-random nonce and encryption is perfect. In the same fashion, when $B$ receives $Nb$ inside message 3, he would conclude that $A$ was at the other end of the network because $Nb$ must have been obtained from message 2, and no-one but $A$ could perform this operation. How rigorous is this reasoning?

### 4.1 Lowe's attack to the Needham-Schroeder Protocol

Lowe shows [Low95] that an informal reasoning of the sort presented above is flawed because of the scenario depicted in figure 4.

Security protocols are implemented as distributed concurrent programs. A malicious agent $C$ can interleave two of the protocol sessions, provided that some agent $A$ initiates one session with $C$. Note that $C$ could be a registered user of the network, so no-one could suspect his tampering. Since $A$ initiates with $C$, she encrypts her nonce and her identity under $C$'s public key. Once obtained these data, $C$ initiates another session (indicated by the primes) with another agent $B$, quoting $A$'s data rather than his own. From this message, $B$ deduces

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

**Fig. 4.** Lowe's attack to the Needham-Schroeder Protocol

that $A$ is trying to communicate with him. Therefore, $B$ replies to $A$, quoting her nonce and his own, $Nb$. Since the entire network is under $C$'s control, $C$ intercepts this message before it is delivered to $A$ but cannot decrypt it because encryption is perfect. So, $C$ forwards it to $A$. The message is of the form that $A$ was expecting, hence $A$ extracts $Nb$ and sends it to the agent with whom she had initiated the first session, $C$. This reveals $Nb$ to $C$, who can use it to complete the session with $B$ by issuing message 3', which is of the form that $B$ was expecting.

As a result, $B$ believes to have communicated with $A$, while $A$ was communicating with $C$. In other words, $C$ impersonates $A$ with $B$: the protocol has failed to achieve authentication. The consequences of the attack are that $B$ will consider every future messages quoting $Nb$ as coming from $A$. If $B$ is a bank and $A$ and $C$ are account holders, $C$ could ask $B$ to transfer money from $A$'s account to $C$'s without $A$ realising it.

Lowe proposes to quote $B$'s identity in message 2, so that $A$ would address message 3 to the agent mentioned in that message rather than to the one with whom she initiated the session. This upgrade would prevent the attack in figure 4 because message 3 would be encrypted under $B$'s public key, so $C$ would not discover $Nb$ and could not issue message $3'$ to complete the session with $B$.

## 5 Modelling Needham-Schroeder

We demonstrate the machinery developed above on the Needham-Schroeder protocol. It is understood that all SCSPs presented below are in fact fragments pertaining to two or three variables.

Figure 5 contains the initial SCSP for our protocol. It may be seen that each agent's knowledge contains public information (agent names and their public keys), private information (the specific agent's private key). All other domain elements that the protocol does not use are set to the more secure level *unknown* (and are omitted in the figure).

Now, we build the imputable SCSP corresponding to the network on which only a protocol session between $A$ and $B$ has initiated and terminated (Figure 6).


**Fig. 5.** Initial SCSP for the Needham-Schroeder protocol

The general procedure (§3.3) must be followed: we extend the initial SCSP with the unary constraints that assign security level *debug* to the nonces invented at run-time; we add the binary constraints that set all exchanged messages to *debug*, and all *remaining* messages to *unknown*.

Intuitively, the more steps the protocol has, the more constraints we put in the store, while the bigger is the probability to introduce insecurity (in this case inconsistency) in the network (respectively in the SCSP).

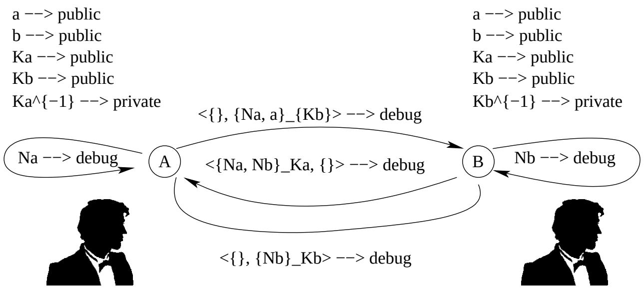


**Fig. 6.** Imputable SCSP for completion of protocol session between $A$ and $B$

We also build the *policy SCSP* (omitted here), and observe that the solution of our imputable SCSP is not lower than that for the policy SCSP. Hence, we can formally confirm that the completion of a session of Needham-Schroeder between two agents bears no attacks.

### 5.1 The attack

Figure 7 shows the imputable SCSP corresponding to Lowe's attack: an agent $C$ is acting as *middle person* between $A$ and $B$.

A larger fragment of the initial SCSP is in Figure 8, while a simplified fragment of the policy SCSP is given in in Figure 9.

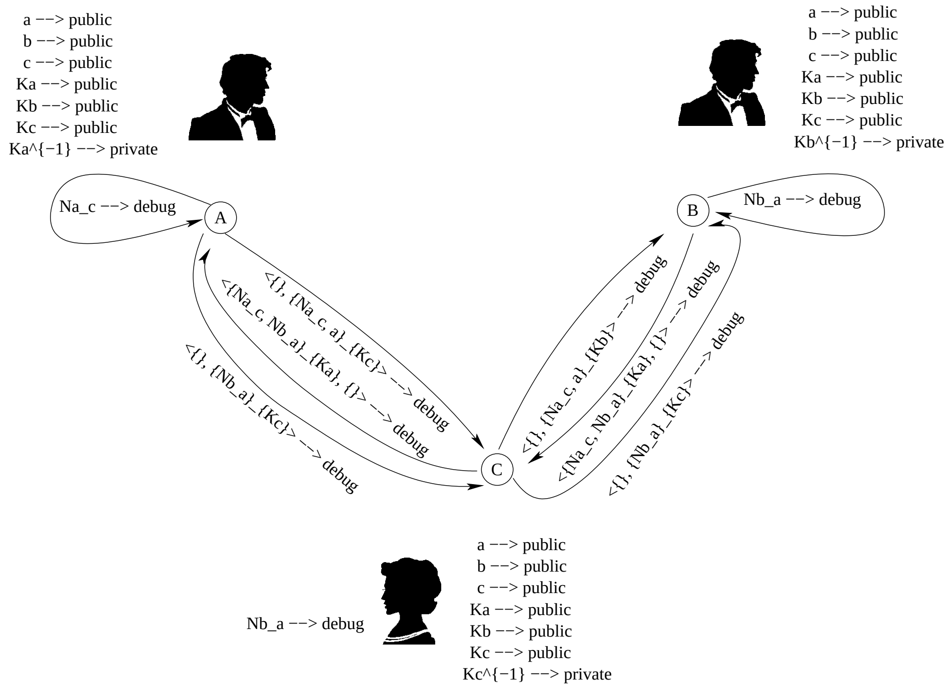


**Fig. 7.** Imputable SCSP corresponding to Lowe's attack

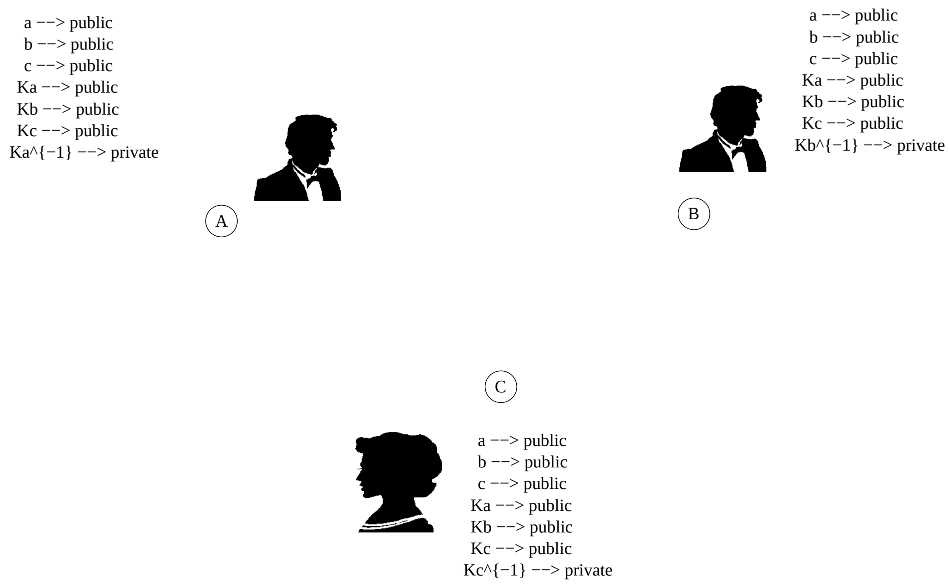


**Fig. 8.** Initial SCSP for the Needham-Schroeder protocol

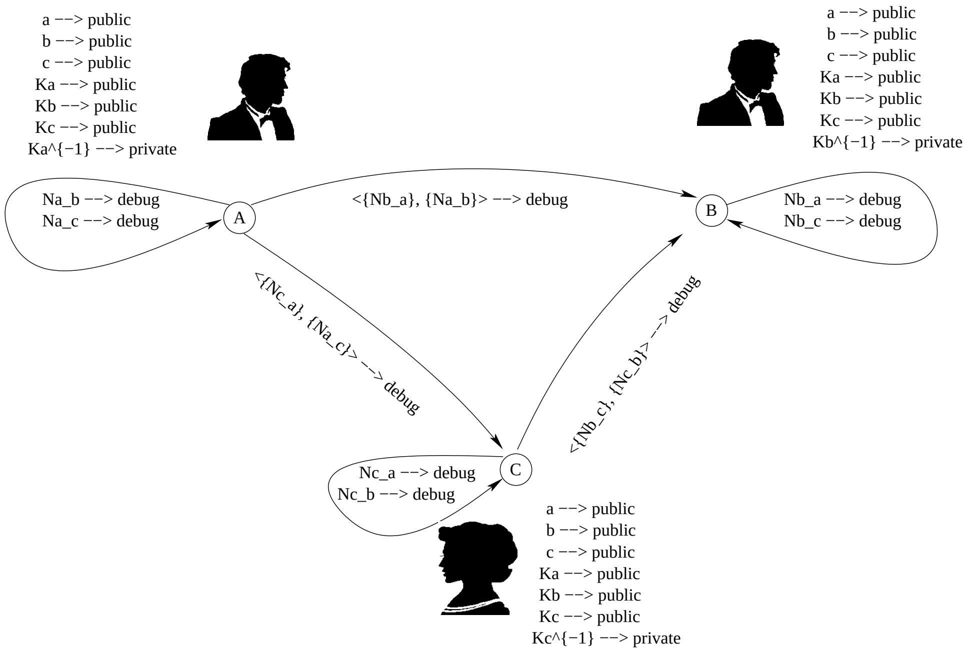


**Fig. 9.** Policy SCSP for the Needham-Schroeder protocol

By restricting our attention to the security levels assigned to the domain values of the variable $C$, we observe that they differ. While the domain value $Nb_a$ has level *unknown* in the policy SCSP, it has level *debug*, by security entailment, in the imputable SCSP. Hence, the given configuration has an attack on $Nb_a$ (which is the nonce that $B$ creates for the session with $A$) because the spy has lowered her security level of that nonce.

## 6 Conclusion and future work

Recent literature (e.g. [Low95,Pau98,Bel99,BR97]) shows that reasoning formally about security protocols is a challenging task. Our work brings to bear that SCSPs can be tailored to model protocol attacks. We have provided general guidance to build the initial SCSP, the policy SCSP for any protocol, and the imputable SCSP that corresponds to any network configuration arising from the execution of that protocol.

An attack is formally defined by comparing the solution of the imputable SCSP with that of the policy SCSP. The lower is the solution of the imputable SCSP w.r.t. that of the policy SCSP, the more significant is the attack on the message that has determined the difference. This feature, which is one of the advances that our approach brings to the field, allows a more scrupulous evaluation of the security weaknesses of the protocol under study.

Our ongoing research includes formalising a dynamic update of the constraint store by means of agents' *telling* new constraints. This will make it possible to model the evolution of the network traffic while the protocol is executed.

## References


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

[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.

[Bis00] S. Bistarelli. *SCSP and SCLP: a general framework for soft constraint solving and programming*. PhD thesis, Dipartimento di Informatica, 2000. working draft.

[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.

[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.

[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.

[LR97] G. Lowe and B. Roscoe. Using CSP to Detect Errors in the TMN Protocol. *IEEE Transactions on Software Engineering*, 3(10), 1997.

[Mac92] A. K. Mackworth. Constraint Satisfaction. In *Encyclopedia of AI (second edition)*, pages 285–293. John Wiley & Sons, 1992.

[Nat77] National Bureau of Standards. *Data Encryption Standard*, Jan 1977. Federal Information Processing Standards Publications, FIPS Pub. 46.

[NS78] R. M. Needham and M. D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, 1978.

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

[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.

[Sar89] V.A. Saraswat. *Concurrent constraint programming language.* PhD thesis, Carnegie-Mellon University, 1989.

[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.

[Sco82] D. S. Scott. Domains for denotational semantics. In *Proc. ICALP*. Springer-Verlag, 1982.

[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.

[Wal96] M. Wallace. Practical Applications of Constraint Programming. *Constraints: An International Journal*, 1996.