Confidentiality Levels and Deliberate/Indeliberate Protocol Attacks

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

[1] Computer Laboratory, University of Cambridge
15 JJ Thomson Avenue, Cambridge CB3 0FD, (UK)
giampaolo.bella@cl.cam.ac.uk
[2] Dipartimento di Matematica e Informatica, Università di Catania
Viale A. Doria 6, I-95125 Catania (Italy)
giamp@dmi.unict.it
[3] Istituto di Informatica e Telematica, C.N.R.
Via G. Moruzzi 1, I-56124 Pisa, Italy
stefano.bistarelli@iit.cnr.it

Abstract. A formal definition of confidentiality is developed using soft (rather than crisp) constraints. The goal is no longer considered as a mere yes/no property as in the existing literature, but gains an extra parameter, the *security level*. The higher the security level, the stronger the goal. For example, different messages may enjoy different levels of confidentiality, and the same message may enjoy different levels of confidentiality for different principals. On this basis, the notion of *indeliberate* confidentiality attack can be captured, whereby a principal learns some message not meant for him because of someone else's tampering. The analysis of Lowe's attack on the Needham-Schroeder protocol reveals a new weakness.

1 Introduction

A major goal of security protocols is *confidentiality*, confirming that a message remains undisclosed to malicious principals. However, it must be stressed that different messages require "specific degrees of protection against disclosure" [11]. 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.

Confidentiality has been essentially formalised as a mere "yes or no" property thus far, so one can just claim that a key is confidential or not. The motivation for our research was studying a finer formal notion for the goal. We have developed the notion of *l-confidentiality*, 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 we adopt 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 message. This lets us formalise that different levels of a goal are granted to different principals.

The framework presented here extends and supersedes an existing kernel [1, 2]. We can conduct a *preliminary analysis* to study in detail what goals the protocol achieves in ideal conditions where no principal acts maliciously. If we are interested, we can also conduct an *empirical analysis* to study what goals the protocol achieves on *a* specific network configuration arising from the protocol execution in the real world. We have observed that Lowe's attack by C against B (after A initiated with C) causes the byproduct that B unexpectedly receives a nonce he is not entitled to know. It can be concluded that B *indeliberately* mounted a confidentiality attack on the nonce. We have discovered that B can later exploit that nonce to mount an attack against C. This outline anticipates that our notation is uniform: there is not one attacker only, but *all principals who perform, either deliberately or not, some operation that is not admitted by the protocol policy are attackers.* Our use of a semiring is loosely inspired to Denning's use of a lattice to characterising secure flows of information through computer systems [7]. The idea of using *levels* to formalise access rights is in fact due to her. Denning too sees an attack whenever an object is assigned a label worse than that initially specified, rather than relying on a single attacker.

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. After an outline on semiring-based SCSPs (Section 2), our framework for protocol analysis is described (Section 3). The paper continues with the presentation of the crucial SCSPs for analysing security protocols (Section 4), and with the definitions concerning confidentiality (Section 5). Then, an empirical analysis of the Needham-Schroeder protocol (Section 6) highlights the possible consequences of an indeliberate confidentiality attack. Finally, some conclusions (Section 7) are given.

2 Soft constraints

Several formalisations of the concept of *soft constraints* are currently available [18, 8, 10]. In the following, we refer to that by Bistarelli et al., which is based on c-semirings [4, 5]. It can be shown to generalise and express a number of other approaches to soft constraints.

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: 1. A is a set, the *career set*, and $\mathbf{0}, \mathbf{1} \in A$; 2. $+$ is commutative, associative and $\mathbf{0}$ is its unit element; 3. $\times$ 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 [4]): 1. $\leq_S$ is a partial order; 2. $+$ and $\times$ are monotone on $\leq_S$; 3. $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum; 4. $\langle A, \leq_S\rangle$ is a complete lattice and, for all $a, b \in A$, $a + b = lub(a, b)$. Moreover, if $\times$ is idempotent, then: $+$ distributes 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, $\leq_S$ will be abbreviated by $\leq$.

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

A *soft constraint satisfaction problem* (SCSP) is a pair $\langle C, con\rangle$ where $con \subseteq \mathcal{V}$ and C is a set of constraints: con is the set of variables of interest for the constraint set C, which however may concern also variables not in con. Notice that a classical constraint satisfaction problem (CSP) [14, 13] is an SCSP where the chosen c-semiring is $S_{CSP} = \langle \{false, true\}, \vee, \wedge, false, true\rangle$.

Fuzzy CSPs [8, 16, 17] can be modelled in the SCSP framework by choosing the c-semiring: $S_{FCSP} = \langle [0, 1], max, min, 0, 1\rangle$. Figure 1 shows the graph representation of a fuzzy CSP. Variables and constraints are represented respectively by nodes and by undirected (unary for c_1 and c_3 and binary for c_2) arcs, and semiring values are written to the right of the corresponding tuples. The variables of interest (that is the set con) are represented with a double circle. Here we assume that the domain D of the variables contains only elements a and b. In the following, we omit the double circle when all the variables are of interest.

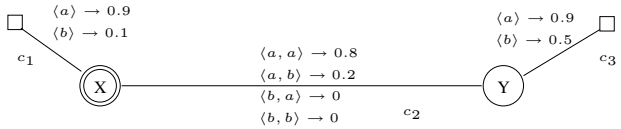

Fig. 1: A fuzzy CSP

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 which asso-

ciates to each tuple of domain values for such variables a semiring element that is obtained by multiplying the elements associated by the original constraints to the appropriate subtuples.

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

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

Solutions. The *solution* of an SCSP problem $\mathsf{P} = \langle C, con \rangle$ is the constraint $Sol(\mathsf{P}) = (\bigotimes C) \Downarrow_{con}$. That is, we combine all constraints, and then project over the variables in *con*. In this way we get the constraint over *con* that 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 sets out by a gentle presentation of our framework and then describes it in more detail.

Using soft constraints requires the definition of a c-semiring. Our *security semiring* (Section 3.1) is used to specify each principal's trust on the security of each message, that is each principal's *security level* on each message. The security levels range from the most secure (highest) 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 that A believes authorised to know m. For simplicity, we state no relation between the granularity of the security levels and the number of principals authorised to know m.

Using the security semiring, we define the *network constraint system* (Section 3.2), which represents the computer network on which the security protocols can be executed. The development of the principals' security levels from manipulation of the messages seen during the protocol sessions can be formalised as a *security entailment* (Section 3.3), that is an entailment relation between constraints. Then, given a specific protocol to analyse, we represent its assumptions

in the *initial SCSP* (Section 4). All admissible network configurations arising from the protocol execution as prescribed by the protocol designers can in turn be represented in the *policy SCSP* (Section 4). We also explain how to represent any network configuration arising from the protocol execution in the real world as an *imputable SCSP* (Section 4).

Given a security level l, establishing whether our definitions of *l-confidentiality* (Section 5) holds in an SCSP requires calculating the solution of the imputable SCSP and projecting it on certain principals of interest. The higher l, the stronger the goal.

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, an attack on a message whose security level is *private* is more serious than an attack on a message whose security level is *public*.

By an *empirical analysis*, we can study what goals the protocol achieves on a specific network configuration arising from the protocol execution under realistic threats. 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 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 operations contributing to the network configuration modelled by the imputable SCSP have taken place so to lower some of the security levels stated by the policy SCSP. It is important to stress that any principal might have performed, either deliberately or not, those operations.

3.1 The Security Semiring

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

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

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: $traded_i\ \times_{sec}\ traded_j\ =\ traded_{min(i,j)}$

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

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 (Section 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.

Notice that CS_n does not depend on any protocols, for it merely portrays a computer network on which any protocol can be implemented. Members of $\mathcal{V}$ 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. They are presented in Figure 2, where function *def* is associated to a generic constraint projected on a generic principal A. Our rules establish that the security level of a message gets somewhat lower each time the message is manipulated by encryption or decryption. Different rules could be studied if one wanted to capture other features.

We now define a binary relation between constraints.

Definition 1 (Relation $\vdash$). *Consider two constraints $c_1, c_2 \in C$ such that $c_1 = \langle def_1, con \rangle$ and $c_2 = \langle def_2, con \rangle$. The binary relation $\vdash$ is such that $c_1 \vdash c_2$ iff def_2 can be obtained from def_1 by a number (possibly zero) of applications of the computation rules for security levels (Figure 2).*

Theorem 1 (Relation $\vdash$ as entailment relation). *The binary relation $\vdash$ is an entailment relation.*

Proof hint. Relation $\vdash$ enjoys the reflexivity and transitivity properties, which characterise an entailment relation.

Following Theorem 1, we address the relation $\vdash$ as *security entailment*. So, if $c_1 \vdash c_2$, we say that c_1 *entails* c_2. We will use the security entailment to compute the security levels that each principal associates to the messages he derives by concatenation, splitting, encryption and decryption.

4 The Initial, Policy and Imputable SCSPs

The designer of a protocol must develop a *policy* to accompany the protocol.

Concatenation:

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

Splitting:

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

Encryption:

$$\frac{\begin{array}{l} traded_{l_1}, traded_{l_2} < unknown; \\ def(m_1) = traded_{l_1}; \; def(m_2) = traded_{l_2}; \; def(\{\!|m_1|\!\}_{m_2}) = traded_{l_3} \end{array}}{def(\{\!|m_1|\!\}_{m_2}) = (traded_{l_1+1} +_{sec} traded_{l_2}) \times_{sec} traded_{l_3}}$$

Decryption:

$$\frac{\begin{array}{l} traded_{l_2}, traded_{l_3} < unknown; \\ def(m_1) = traded_{l_1}; \; def(m_2^{-1}) = traded_{l_2}; \; def(\{\!|m_1|\!\}_{m_2}) = traded_{l_3} \end{array}}{def(m_1) = traded_{l_1} \times_{sec} traded_{l_2+1} \times_{sec} traded_{l_3}}$$

Fig. 2: Computation rules for security levels

The Initial SCSP. 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 (Section 3.1). Precisely, these rules can be translated into unary constraints for the network constraint system. 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, timestamps and public keys; 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); 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.

The Policy SCSP. The policy for a protocol $\mathcal{P}$ also specifies how the messages that must be exchanged during a session between a pair of principals are formed. The protocol designer typically writes a single step as $A \rightarrow B : m$, meaning that principal A sends message m to principal B. Each principal is typically allowed to participate in a number of protocol sessions inventing a number of secrets, namely fresh messages. Assuming both these numbers to be unbounded but finite, a finite number of *events* may take place [9]. These events consist of prin-

cipals' inventing fresh messages, and principals' sending messages constructed by concatenation and/or encryption. So, when a new message is invented, the corresponding constraint is added to the store along with all constraints extracted by security entailment. No message is intercepted because no malicious principal is assumed to be active: A's sending m to B implies that B, and B only, receives it.

BUILD_POLICY_SCSP($\mathcal{P}$)

1. $\mathsf{p} \leftarrow$ *initial SCSP for* $\mathcal{P}$;
2. **for** each event ev allowed by the policy for $\mathcal{P}$ **do**
3. **if** ev = (A *invents* n, for some A and n) **then**
4. $\mathsf{p} \leftarrow \mathsf{p}$ 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** $\langle def, con\rangle = Sol(\mathsf{p}) \Downarrow_{\{A\}} \ \wedge \ def(m) = traded_i$ **in**
7. $\mathsf{p} \leftarrow \mathsf{p}$ extended with binary constraint between A and B that assigns $traded_i$ to $\langle \{\!|\}\!|, m\rangle$ and *unknown* to all other tuples;
8. return p;

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

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 semiring value, alias security level, associated to message m over A is considered. This level is computed by entailment (Section 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). 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.

This reasoning is repeated for each of the unbounded (but finite) number of events allowed by the policy. 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.

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 replaced by vertical dots). 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)
⋮
7.1. **if** ev = (A *sends* m *to* B *intercepted by* C, for some A, m, B and C)
then
7.2. **let** $\langle def, con\rangle = Sol(\mathsf{p}) \Downarrow_{\{A\}} \ \wedge \ def(m) = traded_i$ **in**
7.3. $\mathsf{p} \leftarrow \mathsf{p}$ extended with binary constraint between A and C that assigns $traded_i$ 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.

5 Formalising Confidentiality

"*Confidentiality is the protection of information from disclosure to those not intended to receive it*" [15]. This definition is often simplified into one that is easier to formalise in a model with a single attacker: a message is confidential if it is not known to the attacker. The latter definition is somewhat weaker: if a principal C who is not the attacker manages to learn a session key for A and B, the latter definition holds but the former does not. By considering all principals as potential attackers, we can capture the former definition.

We remark that 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, which appear to capture 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, that is confidentiality, of the message.

Definition 2 (l-confidentiality). *Given an imputable SCSP* p *and a principal* A*, we say that there is* l-confidentiality of m for A in p *iff* $Sol(\mathsf{p}) \Downarrow_{\{A\}} (m) = l$.

Preliminary analysis on confidentiality. By studying the policy SCSP for a given protocol, we can conduct what we name a *preliminary analysis* of the protocol goals. Here, we concentrate on confidentiality.

First, we calculate the solution of the policy SCSP, and project it on some principal of interest A. Let us suppose that two messages m and m' get security levels l and l' respectively, and that $l' < l$. Thus, even if no principal acts maliciously, m' must be manipulated more than m, so A trusts that m will be less at risk than m'. We conclude that the protocol achieves a *stronger confidentiality goal on* m *than on* m' even if it is executed in ideal conditions. 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.

Empirical analysis on 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 3 (Confidentiality attack).
Given a policy SCSP P*, an imputable SCSP* p *for the same protocol, and a principal* A*, we say that* there is a confidentiality attack by A on m in p *iff there is* l*-confidentiality of* m *in* P *for* A*,* l'*-confidentiality of* m *in* p *for* A*, and* $l' < l$.

Therefore, there is a confidentiality attack by A on m in p iff $Sol(\mathsf{P}) \Downarrow_{\{A\}} (m) < Sol(\mathsf{P}) \Downarrow_{\{A\}} (m)$. The more an attack lowers a security level, the worse that attack, so confidentiality attacks can be variously compared.

6 An empirical analysis of Needham-Schroeder

The protocol. Figure 5 presents the asymmetric Needham-Schroeder protocol, which is so popular that it requires little comments.

The goal of the protocol is *authentication*: at completion of a 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. Assuming that encryption is perfect and that the nonces are truly random, authentication is achieved here by confidentiality of the nonces. Indeed, upon reception of Na inside message 2, A would conclude that she is interacting with B, the only principal who could retrieve Na from message 1. In the same fashion, when B receives Nb inside message 3, he would conclude that A was at the other end of the network

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

Fig. 5: The asymmetric Needham-Schroeder protocol

because Nb must have been obtained from message 2, and no-one but A could perform this operation.

Lowe discovers [12] that the protocol suffers the attack in Figure 6, whereby a malicious principal C masquerades as a principal A with a principal B, after A initiated a session with C. The attack, which sees C interleave two sessions, indicates failure of the authentication of A with B, which follows from failure of the confidentiality of Nb. The security levels of all other principals on the nonces Na and Nb are *unknown*. So, by Definition 2, those nonces are *unknown*-confidential for any principal different from A or B.

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

Fig. 6: Lowe's attack to the Needham-Schroeder Protocol

An empirical analysis. We start off by building the initial SCSP, whose fragment for principals A and B is in Figure 7 (the following only features suitable SCSP fragments pertaining to the principals of interest).

Fig. 7: Fragment of the initial SCSP for Needham-Schroeder protocol

Then, we build the policy SCSP for the protocol by BUILD_POLICY_SCSP. Figure 8 presents the fragment pertaining to a single session between principals A and B. The figure indicates that, while A's security level on her nonce Na was initially *private*, it is now lowered to $traded_1$ by entailment because of the binary constraint formalising step 2 of the protocol. Similarly, B's security level on Nb is now $traded_2$ though it was originally *private*. The figure omits the messages that are not relevant to the following discussion.

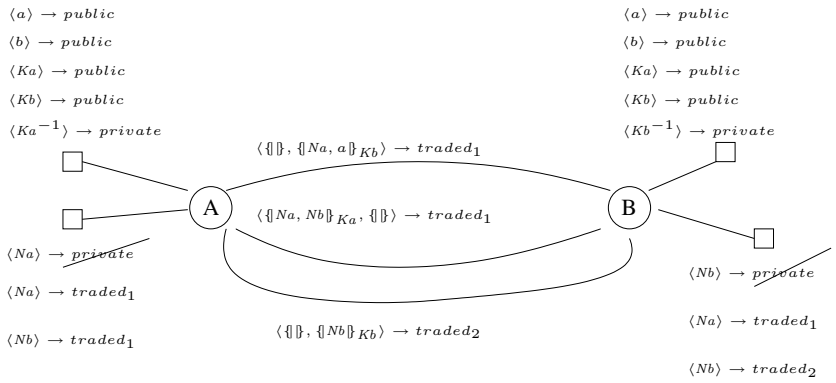

Fig. 8: Fragment of the policy SCSP for the Needham-Schroeder protocol

At this stage, we use BUILD_IMPUTABLE_SCSP to build the imputable SCSP given in Figure 9. It formalises the network configuration defined by Lowe's attack. The solution of this SCSP projected on variable C is a constraint that associates security level $traded_4$ to the nonce Nb. Following Definition 2, Nb is $traded_4$-confidential for C in this SCSP. Hence, by Definition 3, there is a deliberate confidentiality attack by C on Nb in this problem, because Nb got level *unknown* in the policy SCSP. This leads to Lowe's attack.

We discover another attack in the same problem. The problem solution projected on variable B associates security level $traded_2$ to the nonce Na, which instead got level *unknown* in the policy SCSP. This signifies that B has learnt a nonce that he was not allowed to learn by policy, that there is an indeliberate confidentiality attack by B on Na. Notice that the two attacks are uniformly formalised.

As a consequence of the former attack, Lowe reports that, if B is a bank, C can steal money from A's account as follows

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

and the bank B would honour the request believing it came from the account holder A. As a consequence of the attack we have discovered, if A is a bank, B can steal money from C's account as follows

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

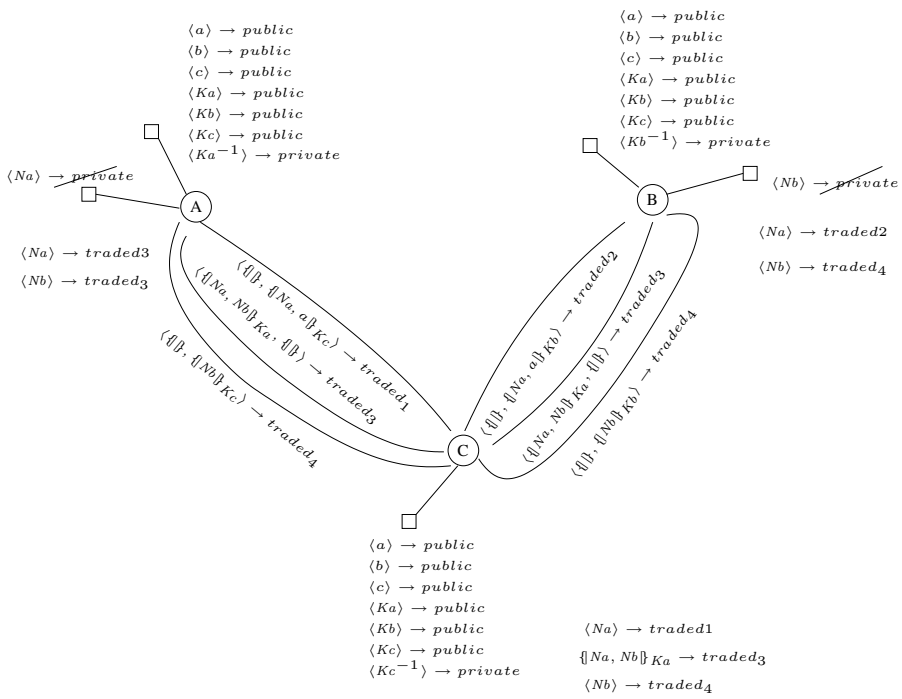

Fig. 9: Fragment of the Imputable SCSP corresponding to Lowe's attack

and the bank A would honour the request believing it came from the account holder C. In practice, it would be sufficient that B realises what Na is for the latter crime to succeed.

There are also less serious attacks. The nonce Na is $traded_3$-confidential for A in this SCSP, while it was $traded_1$-confidential in the policy SCSP. The discrepancy highlights that the nonce has been handled differently from the policy prescription — in fact C reused it with B. Also, Nb's security level for A is $traded_3$ instead of $traded_1$ as in the policy SCSP. Similar considerations apply to Nb, whose security level for B is $traded_4$ instead of $traded_2$. This formalises C's abusive use of the nonce.

7 Conclusions

We have developed a new framework for analysing security protocols, based on a recent kernel [1, 2]. Soft constraint programming allows us to conduct a fine analysis of the confidentiality 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. That configuration may be ideal if every principal behaves according to the protocol, as formalised by the policy SCSP; or, it may arise from the protocol execution in the real world, where some

principal may have acted maliciously, as formalised by an imputable SCSP. We might even compare the forms of the same goal as achieved by different protocols.

We have discovered a new attack on the asymmetric Needham-Schroeder protocol — once C masquerades as A with B, principal B indeliberately gets hold of a nonce that was not meant for him. At this stage, B might decide to exploit this extra knowledge, and begin to act maliciously. Our imputable SCSP modelling the scenario reveals that B's security level on the nonce is lower than that allowed by the policy.

While mechanical analysis was outside our aims, we have implemented a mechanical checker for l-confidentiality on top of the existing *Constraint Handling Rule* (CHR) framework [3]. For example, when we input the policy SCSP for the Needham-Schroeder protocol and the imputable SCSP corresponding to Lowe's attack, the checker outputs

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

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

At this stage, integrating our framework with model-checking tools appears to be a straightforward exercise. The entailment relation must be extended by a rule per each of the protocol messages in order to compute their security levels. Hence, our constraints would be upgraded much the way multisets are rewritten in the work by Cervesato et al. [6] (though they only focus on a single attacker and their properties are classical yes/no properties). Then, once suitable size limits are stated, the imputable SCSPs could be exhaustively generated and checked against our definitions of confidentiality.

Acknowledgements. We are indebted to Martin Abadi for invaluable suggestions, and to Michael Marte and Thom Fruewirth for the preliminary implementation of the checker. Special thanks also to Larry Paulson and to Peter Ryan for useful discussions. Research was funded by the EPSRC grant GR/R01156/01 *Verifying Electronic Commerce Protocols*, by MIUR project *Tecniche e strumenti software per l'analisi della sicurezza delle comunicazioni in applicazioni telematiche di interesse economico e sociale*, by CNR project *Strumenti, ambienti ed applicazioni innovative per la società dell'informazione* and by CSP with the project *SeTAPS*.

References

1. G. Bella and S. Bistarelli. Protocol Analysis using Soft Constraints. Invited talk at S.R.I. Security group, Menlo Park, USA, February 2001.
2. G. Bella and S. Bistarelli. Soft Constraints for Security Protocol Analysis: Confidentiality. In I. V. Ramakrishnan, editor, *Proc. of the 3rd International Symposium on Practical Aspects of Declarative Languages (PADL'01)*, volume 1990 of *LNCS*, pages 108–122. Springer-Verlag, 2001.
3. S. Bistarelli, T. Fruewirth, M. Marte, and F. Rossi. Soft Constraint Propagation and Solving in CHR. In *Proc. ACM Symposium on Applied Computing (SAC), Madrid, Spain*. ACM, mar 2002.
4. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Solving and Optimization. *Journal of the ACM*, 44(2):201–236, Mar 1997.
5. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Logic Programming: Syntax and Semantics. *ACM Transactions on Programming Languages and System (TOPLAS)*, 23:1–29, jan 2001.
6. Iliano Cervesato, N. A. Durgin, Patrick Lincoln, John C. Mitchell, and Andre Scedrov. A Meta-Notation for Protocol Analysis. In *Proc. CSFW*, pages 55–69, 1999.
7. D.E. Denning. A Lattice Model of Secure Information Flow. *Comm. of ACM*, 19(5):236–242, 1976.
8. 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.
9. 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.
10. E. C. Freuder and R. J. Wallace. Partial Constraint Satisfaction. *AI Journal*, 1992.
11. Evelyn Gray. American national standard t1.523-2001, telecom glossary 2000. published on the Web at http://www.its.bldrdoc.gov/projects/telecomglossary2000, 2001.
12. G. Lowe. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Information Processing Letters*, 56(3):131–133, 1995.
13. A.K. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8(1):99–118, 1977.
14. U. Montanari. Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Science*, 7:95–132, 1974. Also Technical Report, Carnegie Mellon University, 1971.
15. B. C. Neuman and T. Ts'o. Kerberos: An Authentication Service for Computer Networks, from IEEE Communications Magazine, September, 1994. In *William Stallings, Practical Cryptography for Data Internetworks*. IEEE Press, 1996.
16. Zs. Ruttkay. Fuzzy Constraint Satisfaction. In *Proc. of 3rd IEEE International Conference on Fuzzy Systems*, pages 1263–1268, 1994.
17. 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.
18. 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.