REASONING ABOUT SECURE INTEROPERATION USING SOFT CONSTRAINTS

Stefano Bistarelli[1,2], Simon N. Foley[3], Barry O'Sullivan[3,4]

[1]*Istituto di Informatica e Telematica, CNR, Pisa, Italy,*
stefano.bistarelli@iit.cnr.it

[2]*Dipartimento di Scienze, Universita degli Studi "G. D'Annunzio", Pescara, Italy,*
bista@sci.unich.it

[3]*Department of Computer Science, University College Cork, Ireland*
{s.foley,b.osullivan}@cs.ucc.ie

[4]*Cork Constraint Computation Centre, University College Cork, Ireland*
b.osullivan@4c.ucc.ie

Abstract The security of a network configuration is based not just on the security of its individual components and their direct interconnections, but also on the potential for systems to interoperate indirectly across network routes. Such interoperation has been shown to provide the potential for circuitous paths across a network that violate security. In this paper we propose a constraint-based framework for representing access control configurations of systems. The *secure reconfiguration* of a system is depicted as a constraint satisfaction problem.

1. Introduction

In its most general case, determining the security of a system is undecidable [Harrison et al., 1976] (the safety problem). This has led to the design of a wide range of decidable security mechanisms that are based on more restrictive forms of security, for example, [Amman and Sandhu, 1992, Bertino et al., 1998]. These mechanisms decide whether an access by a subject is authorized according to the rules set out in a security policy. A system is secure (upholds its security policy) if it is not possible for a subject to gain unauthorized access.

The composition of secure systems is not necessarily secure. A user may be able to gain unauthorized access to an object by taking a circuitous access route across individually secure but interoperating systems [Gong and Qian, 1994, Gong and Qian, 1996]. Determining security is based not just on the individual system authorization mechanisms but also on how the systems are configured

to interoperate. For example, if Alice is permitted to have access to Bob's files on the Administration system, and Clare is permitted access Alice's files on the Sales system, then is it safe to support file sharing between these systems? The extent of system interoperation must be limited if the administration security policy states that Clare is not permitted access to Bob's (administration) files.

The computational challenges of secure interoperation for access control systems is considered in [Gong and Qian, 1994, Gong and Qian, 1996]. In their research Gong and Qian represent access control as an abstract graph of system entities (files, users, etc.) with arcs representing (binary) potential for access. System interoperation is defined as a form of graph composition, and determining whether an interoperation is secure can be performed in polynomial time. However, given systems whose interoperation is not secure, then *optimally* re-configuring the interoperation such that composition is secure is NP-complete. Finding an optimal re-configuration is desirable in order to minimize the extent of the additional access restrictions and maximize desired interoperation: reconfiguring access control to deny all access, while secure, is overly restrictive.

We are interested in the development of practical tools for modelling and analyzing complex system configurations. In this paper we describe how constraints [Bistarelli et al., 1997, Bistarelli, 2004, Freuder and Wallace, 1992] provide a practical and natural approach to modelling and solving the secure interoperation problem. Constraint solving is an emerging software technology for declarative description and effective solving of large problems. The advantages of expressing secure interoperation as a constraint satisfaction problem is that there exists a wide body of existing research results on solving this problem for large systems of constraints in a fully mechanized manner.

In Section 3 we propose a constraint-based framework for representing access control configurations of systems. By building on a semiring of permissions, our framework is sufficiently general to be applied to models such as [Gong, 1999, Sandhu et al., 1996]. Section 4 defines what it means to securely reconfigure a system as a constraint satisfaction problem and Section 5 uses this definition to formulate the meaning of secure interoperation. The advantage of taking the constraint approach is that information about all possible interoperation vulnerabilities are effectively available during analysis. This provides the potential for managing tradeoffs between vulnerabilities using techniques such as [Bistarelli and O'Sullivan, 2003]. Conventional tests for interoperation [Gong and Qian, 1994, Gong and Qian, 1996] are designed to find just one vulnerability. Section 6 considers a special case of secure interoperation that is not unlike the approach described in [Gong and Qian, 1994, Gong and Qian, 1996].

2. Soft Constraints

Constraints have been successfully used in the analysis of a wide variety of problems ranging from network management, for example [Fruehwirth and Brisset, 1997], to complex scheduling such as [Bellone et al., 1992]. They have also been used to analyze security protocols [Bella and Bistarelli, 2001, Bella and Bistarelli, 2002, Bella and Bistarelli, 2004], to represent integrity policy [Bistarelli and Foley, 2003a, Bistarelli and Foley, 2003b], for secure systems interoperation [Bistarelli et al., 2004b, Bistarelli et al., 2004a] and in the development of practical security administration tools [Konstantinou et al., 1999]. In [Konstantinou et al., 1999] constraints are used to help the System Administrator to easily describe network configurations and relations among servers, firewalls and services for the final users. Constraints are used to represent, in a declarative manner, the relations among network objects. This permits the use of local propagation techniques to reconfigure the network when hardware/software changes occur (particularly in a wireless environment). Such automatic reconfiguration would not be possible if the network policy was encoded using conventional shell scripts.

The constraint programming process consists of the generation of requirements (constraints) and solution of these requirements, by specialized constraint solvers. When the requirements of a problem are expressed as a collection of boolean predicates over variables, we obtain what is called the *crisp* (or classical) Constraint Satisfaction Problem (CSP). In this case the problem is solved by finding any assignment of the variables that satisfies all the constraints.

Sometimes, when a deeper analysis of a problem is required, *soft* constraints are used instead [Bistarelli et al., 1997, Bistarelli et al., 2002, Bistarelli, 2004]. Soft constraints associate a qualitative or quantitative value either to the entire constraint or to each assignment of its variables. More precisely, they are based on a semiring structure $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ and a set of variables V with domain D. In particular the semiring operation $\times$ is used to combine constraints together, and the $+$ operator for disjunction, projection and for comparing levels (a partial order $\leq_S$ is defined over A such that $a \leq_S b$ iff $a + b = b$).

Technically, a *constraint* is a function which, given an assignment $\eta : V \rightarrow D$ of the variables, returns a value of the semiring. So $\mathcal{C} = \eta \rightarrow A$ is the set of all possible constraints that can be built starting from S, D and V (values in A are interpreted as levels of preference or importance or cost).

When using soft constraints it is necessary to specify, via suitable combination operators, how the level of preference of a global solution is obtained from the preferences in the constraints. The combined weight of a set of constraints is computed using the operator $\otimes : \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$ defined as

$(c_1 \otimes c_2)\eta = c_1\eta \times_S c_2\eta$. Disjunction of constraints $\oplus : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ is instead defined as follows: $(c_1 \oplus c_2)\eta = c_1\eta +_S c_2\eta$

By using the $\oplus_S$ operator we can easily extend the partial order $\leq_S$ over $\mathcal{C}$ by defining $c_1 \sqsubseteq_S c_2 \iff c_1 \oplus_S c_2 = c_2$. In the following, when the semiring will be clear from the context, we will use $\sqsubseteq$.

Moreover, given a constraint $c \in \mathcal{C}$ and a variable $v \in V$, the *projection* of c over $V - \{v\}$, written $c \Downarrow_{(V-\{v\})}$ is the constraint c' s.t. $c'\eta = \sum_{d \in D} c\eta[v := d]$.

3. Access Configuration

Let ENT represent the domain of all possible entities (subjects, objects, principals) that are of interest across all systems in a network. Access relationships are defined in terms of the permission that one entity holds for another. The current access constraints in a system are represented as a soft-constraint $\mathcal{C}(X, Y)$ over variables X, Y, where for $a, b \in ENT$ then $\mathcal{C}(a, b) \in PERM$ is the access permission that entity a holds for entity b.

Permissions are represented using a semiring $S = \langle PERM, +, \times, \bot, \top \rangle$ where $PERM$ represents the set of all possible permissions, $+$ (union) and $\times$ (intersection) are used to combine permissions. $\bot$ represents the no-access permission and $\top$ represents full-access permission. In general, an entity with permission $p \in PERM$ implicitly has permission $p' \leq p$, where $\leq$ is the partial order relation on the semiring S. Encoding permissions using a partial order is common, for example, [Bell and Padula, 1976] is based on a partial order of security classes, Java Security permissions are partially ordered [Gong, 1999] and [Bharadwaj and Baras, 2003] codifies Role and Permission orderings within a semiring.

DEFINITION 1 *Access Configuration.* An access configuration of a system is represented as a collection of constraints on the access permissions between entities from ENT. □

EXAMPLE 1 Given an arbitrary semiring $S = \langle PERM, +, \times, \bot, \top \rangle$ of permissions, an access configuration that denies all access for all entities in $X, Y \in ENT$ is defined as:

$$\mathcal{C}_\bot(X, Y) \hat{=} \bot$$

A system that places no access restrictions on entities is specified as the null constraint $\mathcal{C}_\top$, where $\mathcal{C}_\top(X, Y) = \top$ for all X, Y. △

EXAMPLE 2 Consider a simple system $S1$ with permissions no-access (**F**) and full-access (**T**) that are represented by the Boolean algebra

$$S_{Bool} \hat{=} \langle \{\mathbf{F}, \mathbf{T}\}, \vee, \wedge, \mathbf{F}, \mathbf{T} \rangle.$$

The system has entities: a, b and c with access constraints

$$\begin{aligned} \mathcal{C}_{S1}(\mathsf{c},\mathsf{b}) &\hat{=} \mathbf{F} \\ \mathcal{C}_{S1}(\mathsf{b},\mathsf{a}) &\hat{=} \mathbf{F} \end{aligned}$$

In this constraint network we can evaluate $\mathcal{C}_{S1}(\mathsf{a},\mathsf{b}) = \mathcal{C}_{S1}(\mathsf{b},\mathsf{c}) = \mathbf{T}$ and, by transitivity, $\mathcal{C}_{S1}(\mathsf{a},\mathsf{c}) = \mathbf{T}$. In practice, access control need not always be transitive and many interesting and useful requirements can be described by, what are effectively, non-transitive access configurations [Lee, 1988, Foley, 1992, Foley, 1997, Foley, 2000]. To model non-transitive access flows, prohibitions on transitive access must be explicitly specified within the system of constraints. For example, adding the constraint $\mathcal{C}_{S1}(\mathsf{a},\mathsf{c}) \hat{=} \mathbf{F}$ implies that $\mathcal{C}_{S1}(\mathsf{a},\mathsf{c})$ is evaluated as $\mathbf{F}$ (the greatest lower bound on the weights of all paths that connect a to c). The class of all access configurations that are based on the boolean semiring of permissions is equivalent to the set of reflexive policies described in [Foley, 1992, Foley, 1997]. △

EXAMPLE 3 A system supports read and write access control, as defined by the semiring $S_{rw} \hat{=} \langle\{2^{\{\mathsf{r},\mathsf{w}\}}, \cup, \cap, \{\}, \{\mathsf{r},\mathsf{w}\}\rangle$. The system has constraints specified as

$$\begin{aligned} \mathcal{C}_{S_{rw}}(\mathsf{b},\mathsf{c}) \hat{=} \{\mathsf{r}\} \qquad & \mathcal{C}_{S_{rw}}(\mathsf{b},\mathsf{a}) \hat{=} \{\} \\ & \mathcal{C}_{S_{rw}}(\mathsf{c},\mathsf{b}) \hat{=} \{\} \end{aligned}$$

and all other accesses are permitted. For example, $\mathcal{C}_{S_{rw}}(\mathsf{a},\mathsf{b}) = \{\mathsf{r},\mathsf{w}\}$ and $\mathcal{C}_{S_{rw}}(\mathsf{a},\mathsf{c}) = \{\mathsf{r}\}$ over entities a, b and c. △

4. Access Reconfiguration

An existing access configuration may be safely re-configured by further restricting (decreasing permission levels) the existing access relationships. Increasing (according to the semiring) permissions between existing system entities is not permitted as it may lead to an entity having access that was previously denied.

DEFINITION 2 *Secure Reconfiguration.* We say that $\mathcal{C}_{S'}$ is a suitable reconfiguration of access configuration $\mathcal{C}_S$ if $\mathcal{C}_{S'} \sqsubseteq \mathcal{C}_S$, where for any assignment η of variables to domain values from ENT, then $\mathcal{C}_{S'}\eta \leq \mathcal{C}_S\eta$. □

It follows by definition that $\sqsubseteq$ is a partial order with most restrictive configuration $\mathcal{C}_\perp$ and least restrictive configuration $\mathcal{C}_\top$. We have for any configuration $\mathcal{C}_S$ that $\mathcal{C}_\perp \sqsubseteq \mathcal{C}_S \sqsubseteq \mathcal{C}_\top$.

EXAMPLE 4 Configuration $\mathcal{C}_{S_{rw}}$ can be securely reconfigured as $\mathcal{C}_{S'_{rw}}$, where

$$\mathcal{C}_{S_{rw}}(\mathsf{b},\mathsf{c}) \hat{=} \mathcal{C}_{S_{rw}}(\mathsf{b},\mathsf{a}) \hat{=} \mathcal{C}_{S_{rw}}(\mathsf{c},\mathsf{b}) \hat{=} \{\}$$

We have $\mathcal{C}_\perp \sqsubseteq \mathcal{C}_{S'_{rw}} \sqsubseteq \mathcal{C}_{S_{rw}} \sqsubseteq \mathcal{C}_\top$. △

5. Access Interoperation

A network is composed of a number of different interoperating systems. For the purposes of this paper we assume that interoperation is represented by entities that are common to the individual systems. For example, a system with user a and a shared filesystem b, interoperates with any system that has the same user a or mounts the same file system b. While a system has control over its own system it has no jurisdiction over access control on other systems. Therefore, when a system interoperates with another, we need to ensure that the interoperation is such that it is not possible for the access rules of the original system to be bypassed by taking a circuitous route through the connected system.

When (securely) composing systems $S1$ and $S2$, the new 'combined' system $S3$ must represent a secure reconfiguration of $S1$ and $S2$, that is, $\mathcal{C}_{S3} \sqsubseteq C_{S1}$ and $\mathcal{C}_{S3} \sqsubseteq C_{S2}$. It is clear that $\mathcal{C}_{\perp}$ is a secure re-configuration as it prohibits all access. However, $\mathcal{C}_{\perp}$ is overly restrictive; we seek the least restrictive secure re-configuration of $S1$ and $S2$.

DEFINITION 3 *Secure Configuration Composition*. The (secure) configuration of interoperating systems $S1$ and $S2$ is configured as $\mathcal{C}_{S1} \otimes \mathcal{C}_{S2}$, ENT, then $(c_1 \otimes c_2)\eta = c_1\eta \times_S c_2\eta$. This corresponds to conjunction of constraints. □

The set of all possible secure access configurations forms a lattice, with partial order $\sqsubseteq$, lowest upper bound operator $\otimes$ and unique lowest bound $\mathcal{C}_{\perp}$. Therefore, the configuration specified by $\mathcal{C}_{S1} \otimes \mathcal{C}_{S2}$ provides the least restrictive secure re-configuration for the interoperation of systems $S1$ and $S2$.

EXAMPLE 5 Using the semiring from Example 2, a system $S3$ manages entities {a,c,d} and has access configuration

$$\mathcal{C}_{S3}(\mathsf{a},\mathsf{c}) \hat{=} \mathbf{F} \qquad \mathcal{C}_{S3}(\mathsf{a},\mathsf{d}) \hat{=} \mathbf{F}$$
$$\mathcal{C}_{S3}(\mathsf{d},\mathsf{c}) \hat{=} \mathbf{F} \qquad \mathcal{C}_{S3}(\mathsf{a},\mathsf{d}) \hat{=} \mathbf{F}$$

Since the system does not control access to entity b, no access constraints can be placed on this entity. The least restrictive re-configuration of the composed system is depicted as $\mathcal{C}_{S1} \otimes \mathcal{C}_{S3}$ in Figure 1, where solid (green) line represents permitted flows (**T**), and dashed (red) lines represent not permitted flow (**F**). This new configuration ensures (under the $\sqsubseteq$ ordering) that the access restrictions of the original configurations are preserved. For example, while $\mathcal{C}_{S1}(\mathsf{d},\mathsf{c}) = \mathbf{T}$ we have $\mathcal{C}_{S1} \otimes \mathcal{C}_{S3}(\mathsf{d},\mathsf{c}) = \mathbf{F}$ since $\mathcal{C}_{S3}(\mathsf{d},\mathsf{c}) \hat{=} \mathbf{F}$.

△

Configuration intersection can be used to guide the re-configuration of the original systems. A system $S1$ that is to be (securely) composed with a system

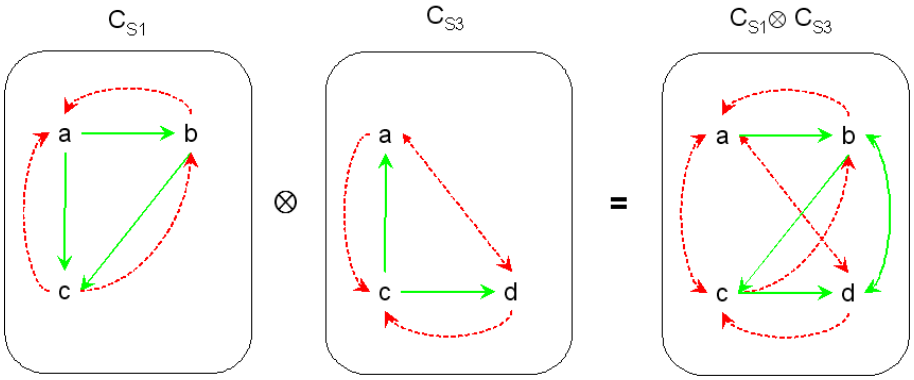

Figure 1. Configurations $\mathcal{C}_{S1}$, $\mathcal{C}_{S3}$ and $\mathcal{C}_{S1} \otimes \mathcal{C}_{S3}$

$S2$ should be re-configured using the access restrictions of $(\mathcal{C}_{S1} \otimes \mathcal{C}_{S2})$. Since $\otimes$ gives the lowest upper bound on configurations according to the secure re-configuration ($\sqsubseteq$) relation, then $(\mathcal{C}_{S1} \otimes \mathcal{C}_{S2})$ gives the least restrictive secure re-configuration of $\mathcal{C}_{S1}$ that also ensures the access restrictions of $\mathcal{C}_{S2}$.

DEFINITION 4 *Strict Secure Interoperation.* Systems $S1$ and $S2$ securely interoperate in a strict manner if they enforce the access constraints of each other, that is, if $\mathcal{C}_{S1}$ can be regarded as a secure re-configuration of $\mathcal{C}_{S2}$ and vice-versa.

To ensure strict secure interoperation, system $S1$ can be (securely) re-configured as $\mathcal{C}'_{S1} \hat{=} (\mathcal{C}_{S1} \otimes \mathcal{C}_{S2})$ and, similarly $\mathcal{C}'_{S2} \hat{=} (\mathcal{C}_{S1} \otimes \mathcal{C}_{S2})$. □

The above definition of secure interoperation is overly restrictive as it requires each system to be able to enforce the access restrictions of the other. While the constraint $(\mathcal{C}_{S1} \otimes \mathcal{C}_{S2})$ represents the best secure (according to $\sqsubseteq$) re-configuration for the 'combined' system (defined in terms of entities from both systems), in practice, the system $S1$ can only enforce the restrictions on the entities that it manages, and similarly for $S2$. It may not be feasible to securely re-configure $S1$ with $\mathcal{C}_{S1} \otimes \mathcal{C}_{S2}$ if $S1$ has no jurisdiction over entities that are managed only by $S2$. We therefore consider a weaker notion of secure interoperation.

Let the *alphabet* $ENT_S \subseteq ENT$ of a system S define the set of entities over which the system S can exercise access control. If we do not require a system to be responsible for access control on entities that are not in its alpabet then for secure interoperation between $S1$ and $S2$ we need only ensure that $\mathcal{C}_{S1}$ enforces the access constraints of the combined system for elements of ENT_{S1}, that is, whenever we have domain entities a,$\mathsf{b} \in ENT_{S1}$ then

$\mathcal{C}_{S1}(\mathsf{a},\mathsf{b}) \leq (\mathcal{C}_{S1} \otimes \mathcal{C}_{S2})(\mathsf{a},\mathsf{b})$. This can be defined in terms of the secure re-configuration relation as follows.

DEFINITION 5 *Loose Secure Interoperation.* Let $\mathcal{C}_S^{\top}$ representing a system S that places/assumes no access constraint over elements in ENT_S, and completely denies flows among entities when one of them is not in ENT_S. More formally, we have $\mathcal{C}_S^{\top}(X,Y) = \top$ when both X and Y are elements of ENT_S, and $\mathcal{C}_S^{\top}(X,Y) = \bot$ when either X or Y (or both) are not elements of ENT_S. Systems $S1$ and $S2$ securely interoperate in a loose manner if they uphold the constraints (with respect to elements from their alphabet) in their composition, that is,

$$\begin{aligned} \mathcal{C}_{S1} \otimes \mathcal{C}_{S1}^{\top} &\sqsubseteq (\mathcal{C}_{S1} \otimes \mathcal{C}_{S2}) \\ \mathcal{C}_{S2} \otimes \mathcal{C}_{S2}^{\top} &\sqsubseteq (\mathcal{C}_{S1} \otimes \mathcal{C}_{S2}) \end{aligned}$$

To ensure loose secure interoperation, system $S1$ should be (securely) re-configured as $C'_{S1}(X,Y) \hat{=} (\mathcal{C}_{S1} \otimes \mathcal{C}_{S2})$ when $X, Y \in ENT_S$, and similarly for $S2$. □

EXAMPLE 6 Continuing Example 5, $S1$ and $S3$ are re-configured for loose secure interoperation as depicted in Figure 2. Note that in practice, networks

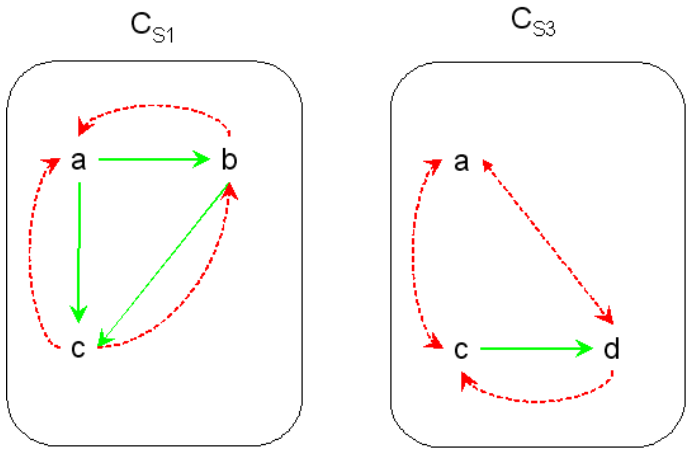

Figure 2. Re-configurations $\mathcal{C}'_{S1}$ and $\mathcal{C}'_{S3}$

$\mathcal{C}'_{S1}$ and $\mathcal{C}'_{S2}$ would also include nodes d and b, respectively, but with no connecting arcs (unconstrained permissions). Notice also that as in Example 5 solid (green) line represents permitted flows (**T**), and dashed (red) lines represent not permitted flow (**F**).

If systems $\mathcal{C}_{S1}$ and $\mathcal{C}_{S3}$ are reconfigured in this way then we can be confident that their interoperation will be secure. △

6. Access Transitivity

Reconfiguration for (loose) secure interoperation gives the most permissive reconfiguration (that does not violate the original configurations). If a system does not include an entity in its alphabet then it is assumed that it places no restrictions on access to it.

It is useful to consider variations of this operation for more restrictive scenarios. In particular, some entities that are common to interoperating systems may induce transitive relationships between entities. For example, suppose that c is a service that is shared between systems $S1$ and $S3$ (Example 5), and $\mathcal{C}_{S1}(\mathsf{b},\mathsf{c})$ and $\mathcal{C}_{S3}(\mathsf{c},\mathsf{d})$. Rather than permitting all accesses between b and d (as computed by $\otimes$, since there are no explicit restrictions these entities), we could instead assume that there is an implicit transitive restriction and compute a limited transitive closure, allowing access from b to d, but not vice-versa.

DEFINITION 6 *Secure Reconfiguration For Transitivity*. A system $S1$ with configuration $\mathcal{C}_{S1}$ is securely reconfigured as $\mathcal{C}_{S1}^{*A}$ to deal with transitive entities A, where

$$\mathcal{C}_{S1}^{*A}(X,Z) \;\hat{=}\; \mathcal{C}_{S1}(X,Z) \otimes (\mathcal{C}_{S1}(X,Y) \otimes \mathcal{C}'_{S1}(Y,Z)) \Downarrow_{\{X,Z\}}$$

where $\mathcal{C}'_{S1}$ is defined as follows:

- for each entity $\mathsf{e} \in ENT$, $\mathcal{C}'_{S1}(\mathsf{e},\mathsf{e}) = \top$;
- if $\langle \mathsf{e},\mathsf{g} \rangle \in \mathcal{C}_{S_1}$, and $\mathsf{e} \in A$ (that is e is a transitive entity), then $\mathcal{C}'_{S1}(\mathsf{e},\mathsf{g}) = \top$;

Note that we have $\mathcal{C}_{S1}^{*A} \sqsubseteq \mathcal{C}_{S1}$ □

EXAMPLE 7 The secure transitive interoperation reconfiguration of $S1$ and $S3$ (Example 5) with transitive entity c is depicted in Figure 3. △

DEFINITION 7 *(Loose) Secure Transitive Interoperation*. Systems $S1$ and $S2$ securely interoperate given transitive entities A if they do not have to be securely reconfigured for interoperation, that is,

$$\begin{array}{rcl} \mathcal{C}_{S1} \otimes \mathcal{C}_{S1}^{\top} & \sqsubseteq & (\mathcal{C}_{S1} \otimes \mathcal{C}_{S2})^{*A} \\ \mathcal{C}_{S2} \otimes \mathcal{C}_{S2}^{\top} & \sqsubseteq & (\mathcal{C}_{S1} \otimes \mathcal{C}_{S2})^{*A} \end{array}$$

□

Another variation of the scenarios consider a different type of transitivity. In the previous example we assume that there is an implicit transitive restriction on entity c and we compute a limited transitive closure, allowing access from b to d, but not vice-versa. Here, we assume instead that entity b can have

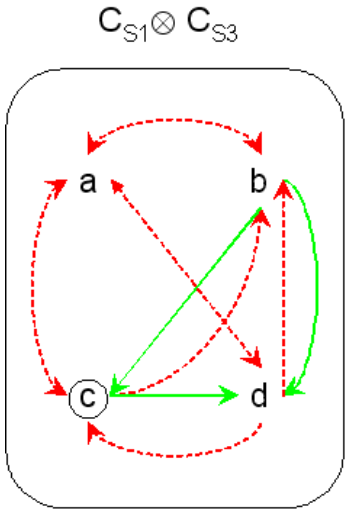

Figure 3. Reconfiguration $(\mathcal{C}_{S1} \otimes \mathcal{C}_{S3})^{*\{\mathsf{c}\}}$

implicit transitive permission (instead of restriction). This means that since we have a flow between a and b and between b and c, we must have also a flow between a and c.

EXAMPLE 8 The secure transitive interoperation reconfiguration of $S1$ and $S3$ (Example 5) with transitive entity c for permission flows is depicted in Figure 4. △

DEFINITION 8 *Secure Reconfiguration For Transitivity (permissions).* A system $S1$ with configuration $\mathcal{C}_{S1}$ is securely reconfigured as $\mathcal{C}_{S1}^{\Box A}$ to deal with transitive entities A, where

$$\mathcal{C}_{S1}^{\Box A}(X, Z) \quad \hat{=} \quad (\mathcal{C}_{S1}(X, Y) \otimes \mathcal{C}'_{S1}(Y, Z)) \Downarrow_{\{X,Z\}}$$

where $\mathcal{C}'_{S1}$ is defined as follows:

- for each entity $\mathsf{e} \in ENT$, $\mathcal{C}'_{S1}(\mathsf{e}, \mathsf{e}) = \top$;
- if $\langle \mathsf{e}, \mathsf{g} \rangle \in \mathcal{C}_{S_1}$, and $\mathsf{e} \in A$ (that is e is a transitive entity), then $\mathcal{C}'_{S1}(\mathsf{e}, \mathsf{g}) = \top$;

Note that we have $\mathcal{C}_{S1} \sqsubseteq \mathcal{C}_{S1}^{\Box A}$ □

7. Discussion and Conclusions

The approach that we present in this paper represents a paradigm shift in the modelling and analysis of interoperability. We present a constraint model that

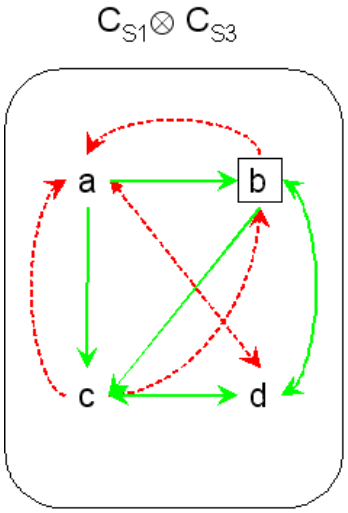

Figure 4. Reconfiguration $(\mathcal{C}_{S1} \otimes \mathcal{C}_{S3})^{\square\{\mathsf{C}\}}$

provides a natural description of a network of interoperating systems. While constraint solving is NP-complete in general, this has not detracted from its uptake as a practical approach to solving many real-world problems [Wallace, 1996]. Previous approaches determine secure interoperation in polynomial time, but re-configuring an existing network of systems for secure interoperation, in an optimal way, is NP-complete [Gong and Qian, 1994, Gong and Qian, 1996]. Using a constraint model, we can rely on a significant body of successful techniques from the field of constraint processing for finding the set of secure re-configurations with reasonable effort. As part of our future work in this area we plan to develop an constraint-based implementation with which to demonstrate our approach on some real world data.

References

Amman, P. and Sandhu, R. (1992). The extended schematic protection model. *Journal of Computer Security*, 1(4).

Bell, D.E. and Padula, L. J. La (1976). Secure computer system: unified exposition and MULTICS interpretation. Report ESD-TR-75-306, The MITRE Corporation.

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

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

Bella, G. and Bistarelli, S. (2004). Soft constraint programming to analysing security protocols. *Theory and Practice of Logic Programming (TPLP)*, 4(5):1–28. To appear.

Bellone, J., Chamard, A., and Pradelles, C. (1992). Plane - an evolutive planning system for aircraft production. In *Proc. 1st Interantional Conference on Practical Applications of Prolog (PAP92)*.

Bertino, E. et al. (1998). An authorization model and its formal semantics. In *Proceedings of the European Symposium on Research in Computer Security*, pages 127–142. Springer LNCS 1485.

Bharadwaj, V.G and Baras, J.S. (2003). Towards automated negotiation of access control policies. In *Proc. of IEEE Workshop Policies for Distributed Systems and Networks*, pages 77–80.

Bistarelli, S. (2004). *Semirings for Soft Constraint Solving and Programming*, volume 2962 of *Lecture Notes in Computer Science*. Springer.

Bistarelli, S. and Foley, S.N. (2003a). Analysis of integrity policies using soft constraints. In *Proceedings IEEE 4th International Workshop on Policies for Distributed Systems and Networks (POLICY2003), Lake Como,Italy, June 4-6, 2003*, pages 77–80. IEEE Press.

Bistarelli, S. and Foley, S.N. (2003b). A constraint based framework for dependability goals: Integrity. In *22nd International Conference on Computer Safety, Reliability and Security (SAFECOMP2003), Proceedings, 23-26 September 2003, Edinburgh, Scotland, United Kingdom*, volume 2788 of *Lecture Notes in Computer Science*, pages 130–143. Springer.

Bistarelli, S., Foley, S.N., and O'Sullivan, B. (2004a). Detecting and eliminating the cascade vulnerability problem from multi-level security networks using soft constraints. In *Proceedings Innovative Applications of Artificial Intelligence Conference (IAAI-04)*, pages 808–813. AAAI Press.

Bistarelli, S., Foley, S.N., and O'Sullivan, B. (2004b). Modelling and detecting the cascade vulnerabiliy problem using soft constraints. In *Proc. ACM Symposium on Applied Computing (SAC 2004)*, pages 383–390. ACM Press.

Bistarelli, S., Montanari, U., and Rossi, F. (1997). Semiring-based constraint solving and optimization. *Journal of ACM*, 44(2):201–236.

Bistarelli, S., Montanari, U., and Rossi, F. (2002). Soft concurrent constraint programming. In *Programming Languages and Systems: 11th European Symposium on Programming, ESOP 2002 held as Part of the Joint European Conference on Theory and Practice of Software, ETAPS 2002, Proceedings, Grenoble, France, April 8-12, 2002*, volume 2305 of *Lecture Notes in Computer Science*, pages 53–67. Springer.

Bistarelli, S. and O'Sullivan, B. (2003). A theoretical framework for tradeoff generation using soft constraints. In *Research and Development in Intelligent Systems XX, Proceedings of AI-2003, the Twenty-third SGAI International Conference on Knowledge-Based Systems and Applied Artificial Intelligence*, pages 69–82. Springer, BCS Conference Series "Research and Development in Intelligent Systems xx".

Foley, S.N. (1992). Aggregation and separation as noninterference properties. *Journal of Computer Security*, 1(2):159–188.

Foley, S.N. (1997). The specification and implementation of commercial security requirements including dynamic segregation of duties. In *ACM Conference on Computer and Communications Security*, pages 125–134.

Foley, S.N. (2000). Conduit cascades and secure synchronization. In *ACM New Security Paradigms Workshop*.

Freuder, E.C. and Wallace, R.J. (1992). Partial constraint satisfaction. *AI Journal*, 58.

Fruehwirth, T. and Brisset, P. (1997). Optimal planning of digital cordless telecommunication systems. In *Proc. PACT97*, London, UH.

Gong, L. (1999). *Inside Java2 Platform Security*. Addison Wesley.

Gong, L. and Qian, X. (1994). The complexity and composability of secure interoperation. In *Proceedings of the Symposium on Security and Privacy*, pages 190–200, Oakland, CA. IEEE Press.

Gong, L. and Qian, X. (1996). Computational issues in secure interoperation. *IEEE Trans. Softw. Eng.*, 22(1):43–52.

Harrison, M., Ruzzo, W., and Ullman, J. (1976). Protection in operating systems. *Communications of the ACM*, 19:461–471.

Konstantinou, A.V., Yemini, Y., Bhatt, S., and Rajagopalan, S. (1999). Managing security in dynamic networks. In *Proc. USENIX Lisa'99*.

Lee, T.M.P. (1988). Using mandatory integrity to enforce 'commerical' security. In *Proceedings of the Symposium on Security and Privacy*, pages 140–146.

Sandhu, R. et al. (1996). Role based access control models. *IEEE Computer*, 29(2):38–47.

Wallace, M. (1996). Practical applications of constraint programming. *Constraints*, 1(1–2):139–168.