Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
Electronic Notes in Theoretical Computer Science 168 (2007) 191–205
www.elsevier.com/locate/entcs

On Relating and Integrating Two Trust Management Frameworks

Fabio Martinelli[1] Marinella Petrocchi[2]

Istituto di Informatica e Telematica, CNR
Area della Ricerca di Pisa, Via G. Moruzzi 1, 56124 Pisa, Italy

Abstract

In this paper, we continue our line of research for defining an integrated framework for the specification and analysis of security and trust, aimed at providing the basis for the modeling and analysis of security and trust aspects of emergent dynamic coalitions (*e.g.*, mobile ad hoc networks, peer to peer systems, web services). In particular, we consider two well-known languages: 1) the Role-based Trust-management framework (RT) and, in particular, its most basic language RT_0, and 2) the transitive trust model, for defining trust and recommendation relationships. First, we show an encoding of the transitive trust model into part of RT_0; then, this subset is mapped into the inference construct of the Crypto-CCS process algebra. Also, we introduce in the languages operators dealing with levels of trust. The relationships among these languages could allow us to model and analyze trust and recommendation issues in distributed systems by means of standard formal techniques, based on inference systems.

Keywords: trust management languages, process algebras

1 Introduction

In [14], we started an investigation about the development of a uniform framework for the specification and (automated) analysis for security and trust in distributed protocols. In particular, it has been shown how the same machinery based on inference systems used for the formal verification of security protocols may be used to analyze access control policies based on trust management systems (*e.g.,* see [3,6,16,11,20]). However, we did not show a full integration, since the same framework could be used to express several theories, but not at the same time. For instance, both the role-based trust management approach advocated in [11] and the transitive trust model of [7,6] have been modeled. However, no attempt was made to compare them or to integrate these two frameworks in a single one. Here, we extend that work by providing an integrated inference system for describing both

[1] Email: fabio.martinelli@iit.cnr.it

[2] Email: marinella.petrocchi@iit.cnr.it

doi:10.1016/j.entcs.2006.12.005

trust and recommendation relationships. In doing so, we also relate the Role-based Trust-management framework (in particular, the basic RT_0, [11]) and the transitive trust model of [7,6], by showing an encoding of the latter into part of the former. In addition, we present a flexible framework for giving metrics on trust relationships, and we provide a suitable inference system modeling the unified framework that results from our relationships.

Relating trust management languages and trying to unify trust features in a pre-existent framework already dealing with security is particularly appealing for our aims.

Indeed, dealing with trust and recommendation is a peculiarity of so called dynamic coalitions, sets of electronic devices typically belonging to different security domains, and possibly driven by different purposes, that should cooperate in order to maintain active basic functionalities of the whole network. An example of a dynamic coalition is a mobile ad hoc network: here, communication is typically multihop, and all the nodes are called to actively participate in, *e.g.,* routing functionalities, *i.e.,* they should forward packets towards a destination on behalf of the source. There is no reason, however, to assume that the nodes in the network will eventually cooperate with one another, since, *e.g.,* network operation consumes energy, a scarce resource in environments like ad hoc networks. Recently, researchers have proposed several mechanisms based on trust, reputation and recommendation, in order to enforce cooperation between nodes, [1].

Another interesting field of application is the area of web services. Several authors advocate the usage of formal methods, and in particular of process algebra, to model web services and their composition (*e.g.,* see [4,17]). Crypto-CCS, [12,13], the language we have chosen to adopt for showing the attempt of a unified framework modeling both security and trust, can implement several sets of inference rules. Some of them can be considered not only for message manipulation, but also they can encode complex reasoning systems aimed at describing preferences or attitudes. Imagine a service that needs to interact with others. To protect itself from malicious or selfish services, it can previously collect information (often in the form of cryptographic messages) on the behavior of others. Thus, an inference system may be suitably used to express trust towards others, by means of direct experience, or it can be used to infer trust relationships through recommendations of third services.

The structure of the paper is as follows. Next section recalls the Crypto-CCS language. Section 3 recalls part of the RT family of languages for describing role-based credentials. In Section 4, we rephrase, in a formalization more suitable for our purposes, the transitive trust model proposed by [6,7]. Then, we propose a simplification of that trust model, in order to make it compliant to the basic language RT_0 in the RT family (Subsection 4.1). Subsections 4.2 and 4.3 adequately extend our formalization with operators dealing with levels of trust, according to the original intention of [6]. In Section 5, we show how the simplified trust model can be naturally encoded into part of RT. Section 6 shows how Crypto-CCS can, in turn, efficiently model the given encoding. Finally, Section 7 gives some final remarks.

2 Crypto-CCS

Here, we recall syntax and semantics of the formal language Crypto-CCS ([12,13]). The language consists of a (parametric) data-handling part and a control part.

The data-handling part consists of a message set *Msgs* and a (parametric) inference system. The set *Msgs* is defined by the grammar:

$$m ::= x \mid b \mid F^1(m_1, \ldots, m_{k_1}) \mid \ldots \mid F^l(m_1, \ldots, m_{k_l})$$

where m ranges over *Msgs*, F^i (for $1 \leq i \leq l$) are the constructors for messages, $x \in V$, a countable set of variables, $b \in B$, a collection of basic messages, and k_i, for $1 \leq i \leq l$, gives the number of arguments of the constructor F^i. Messages without variables are closed messages.

Inference systems model the possible operations on messages. These systems consist of a set of rules r:

$$r = \frac{m_1 \quad \ldots \quad m_n}{m_0}$$

where $m_1, \ldots, m_n$ are premises (possibly empty) and m_0 is the conclusion. An instance of the application of the rule r to closed messages $m_1, \ldots, m_n$ is denoted as $m_1 \quad \ldots \quad m_n \vdash_r m_0$. For each rule r and set of closed messages $\{m_1, \ldots m_n\}$, we assume that the set $\{m \mid m_1, .., m_n \vdash_r m\}$ is decidable and that we can effectively establish whether it is empty or not.

The control part of the language defines terms standing for processes in a concurrent system. The terms are defined as follows:

$$P, Q ::= \mathbf{0} \mid c(x).P \mid \bar{c}m.P \mid \tau.P \mid P \,|\, Q \mid P \backslash L \mid$$

$$A(m_1, \ldots, m_r) \mid [\langle m_1, \ldots, m_r \rangle \vdash_{rule} x]P; Q$$

where $m, m_1, \ldots, m_r$ are either closed messages or variables, c is a channel and L is a set of channels. Both the operators $c(x).P$ and $[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]P; Q$ bind variable x in P.

We assume the usual conditions about closed and guarded processes, as in [15]. We call $\mathcal{P}$ the set of all the Crypto-CCS closed and guarded terms. The set of actions is $Act = \{c(m) \mid m \text{ closed and } c \in I\} \cup \{\bar{c}m \mid m \text{ closed and } \bar{c} \in O\} \cup \{\tau\}$ (τ is the internal, invisible action), ranged over by a. We give an informal overview of Crypto-CCS operators:

- $\mathbf{0}$ is a process that does nothing.
- $c(x).P$ represents the process that can get as input a closed message m on channel c behaving like $P[m/x]$).
- $\bar{c}m.P$ is the process that can send m on channel c, and then behaves like P.
- $\tau.P$ is the process that executes the invisible τ and then behaves like P.
- $P \,|\, Q$ (*parallel*) is the parallel composition of processes that can proceed in an asynchronous way but they must synchronize on complementary actions to make a communication, represented by a τ.

- $P \backslash L$ is the process that cannot send and receive messages on channels in L; for all the other channels, it behaves exactly like P;
- $A(m_1, \dots, m_r)$ behaves like the respective defining term P where all the variables $x_1, \dots, x_r$ are replaced by the closed messages $m_1, \dots, m_r$;
- $[\langle m_1, \dots, m_r \rangle \vdash_{rule} x]P; Q$ is the process used to model message manipulation as cryptographic operations. Indeed, the process $[\langle m_1, \dots, m_r \rangle \vdash_{rule} x]P; Q$ tries to deduce an information z from the tuple $\langle m_1, \dots, m_r \rangle$ through the application of rule $\vdash_{rule}$; if it succeeds then it behaves like $P[z/x]$, otherwise it behaves as Q. The set of rules that can be applied is defined through an inference system (*e.g.,* see Figure 1 for an instance).

Crypto-CCS syntax, its semantics and the results obtained are completely parametric with respect to the inference system used. Figure 1 shows an instance inference system to model message handling and public key cryptography: it allows to combine two messages, obtaining a pair (rule $\vdash_{pair}$); to extract one message from a pair (rules $\vdash_{fst}$ and $\vdash_{snd}$); to digitally sign a message m with a key k^{-1} obtaining $\{m\}_{k^{-1}}$ and, finally, to verify a digital signature on a message of the form $\{m\}_{k^{-1}}$ by applying the corresponding public key k (rules $\vdash_{sign}$ and $\vdash_{ver}$, respectively).

In a similar way, inference systems can contain rules for handling the basic arithmetic operations and boolean relations among numbers, so that the value-passing CCS if-then-else construct can be obtained via the $\vdash_{rule}$ operator.

Example 2.1 Equality check among messages can be implemented through the usage of the inference construct. Consider, *e.g.,* the following rule:
$\dfrac{x \quad x}{Equal(x,x)}$ *equal*. Then, $[m = m']P$ (with the expected semantics) may be equivalently expressed as $[m \quad m' \vdash_{equal} y]P$ where y does not occur in A. Similarly, we can define inequalities, *e.g.,* $\leq$, among numbers.

The operational semantics of a Crypto-CCS term is described by means of the *labelled transition system* (*lts*, for short) $\langle \mathcal{P}, Act, \{\xrightarrow{a}\}_{a \in Act} \rangle$, where $\{\xrightarrow{a}\}_{a \in Act}$ is the least relation between Crypto-CCS processes induced by the axioms and inference rules of Figure 2.

3 *RT*: A Family of Languages for Trust Management

In the *RT* family of languages, [11], credentials carry information on policies to define attributes of principals by starting from assertions of other principals. *RT* combines the strenghts of Role-based Access Control (RBAC), [18], by inheriting the notion of role, interposed in the assignment of permissions to users, and of Trust Management, [3], by inheriting principles for managing distributed authority through credentials.

Thus, in *RT* a central concept is the notion of role. A role is formed by a principal and a role term. If principals are denoted as $A, B, C...$ and role terms are denoted as $r, r_1, r_2...$, then $A.r$ is role term r defined by principal A. A role may define a set of principals who are members of this role, and each principal A

$$\frac{m \quad m'}{(m,m')}(\vdash_{pair}) \qquad \frac{(m,m')}{m}(\vdash_{fst}) \qquad \frac{(m,m')}{m'}(\vdash_{snd})$$

$$\frac{m \quad k^{-1}}{\{m\}_{k^{-1}}}(\vdash_{sign}) \qquad \frac{\{m\}_{k^{-1}} \quad k}{m}(\vdash_{ver})$$

Fig. 1. An example inference system for public key cryptography.

$$(inp)\frac{m \in Msgs,\ m\ closed}{c(x).P \xrightarrow{c(m)} P[m/x]} \qquad (out)\frac{m \in Msgs,\ m\ closed}{\bar{c}m.P \xrightarrow{\bar{c}m} P} \qquad (int)\frac{}{\tau.P \xrightarrow{\tau} P}$$

$$(\backslash L)\frac{P \xrightarrow{c(m)} P' \quad c \notin L}{P\backslash L \xrightarrow{c(m)} P'\backslash L} \qquad (|)_1\frac{P_1 \xrightarrow{a} P_1'}{P_1 \,|\, P_2 \xrightarrow{a} P_1' \,|\, P_2} \qquad (|)_2\frac{P_1 \xrightarrow{c(x)} P_1' \quad P_2 \xrightarrow{\bar{c}m} P_2'}{P_1 \,|\, P_2 \xrightarrow{\tau} P_1' \,|\, P_2'}$$

$$(Def)\frac{P[m_1/x_1,\ldots,m_n/x_n] \xrightarrow{a} P' \quad A(x_1,\ldots,x_n) \doteq P}{A(m_1,\ldots,m_n) \xrightarrow{a} P'}$$

$$(\mathcal{D})\frac{\langle m_1,\ldots,m_r\rangle \vdash_{rule} m \quad P[m/x] \xrightarrow{a} P'}{[\langle m_1,\ldots,m_r\rangle \vdash_{rule} x]P;Q \xrightarrow{a} P'}$$

$$(\mathcal{D}_1)\frac{\nexists m \text{ s.t. } \langle m_1,\ldots,m_r\rangle \vdash_{rule} m \quad Q \xrightarrow{a} Q'}{[\langle m_1,\ldots,m_r\rangle \vdash_{rule} x]P;Q \xrightarrow{a} Q'}$$

Fig. 2. Structured Operational Semantics for Crypto-CCS (symmetric rules for $|_1, |_2$ and $\backslash L$ are omitted)

defines who are the members of each role of the form $A.r$. Also, roles can be seen as attributes, *i.e.,* a principal is a member of a role if and only if it has the attribute identified by the role, [11].

In the following, we recall the basic statements of the *RT* family.

3.1 RT_0

The most basic language in *RT* family was presented in [11]. In RT_0 the statements take the form of role definitions.

- $A.r \leftarrow D$ (**simple member**)
  A and D are (possibly the same) principals. The statement defines that D is member of role $A.r$, or, equivalently, A says that D has attribute r.
- $A.r \leftarrow B.r_1$ (**simple containment**)
  A and B are (possibly the same) principals. r and r_1 are (possibly the same) role terms. With this statement, A defines all members of role $B.r_1$ to be members of role $A.r$. In other words, $B.r_1$ is a more powerful role than $A.r$, since everybody who has role $B.r_1$ automatically has role $A.r$. Alternatively, we may say that A delegates part of its capability to assign its role r to B. In terms of attributes, the statement says that if B defines someone to have attribute r_1, than the same

has attribute r according to A.

- $A.r \leftarrow A.r_1.r_2$ (**linking containment**)
  This statement defines a linked role. If B has role $A.r_1$ and D has role $B.r_2$, then D has role $A.r$. In terms of attributes, the statement says that, if A says that B has attribute r_1, and B says that D has attribute r_2, then A says that D has attribute r. Note that this works as a sort of attribute-based delegation. Indeed, A recognizes to B the authority to assign attribute r_2 not on behalf of B's identity, but on behalf of some attribute r_1 that B holds.
- $A.r \leftarrow A_1.r_1 \cap A_2.r_2$ (**Intersection containment**)
  This statement defines that if D has both roles $A_1.r_1$ and $A_2.r_2$, then D has role $A.r$.

Example 3.1 Consider the following set of credentials.

$$Shop.rentmovies \leftarrow Shop.user$$
$$Shop.user \leftarrow MP$$

The principal *Shop* rents movies to anyone who is a registered user of *Shop* itself. Since the second statement assigns the role of a registered user to MP, then it follows that principal MP has permission for having a movie rented.

3.2 RT_1 and RT_1^C

In RT_0, role terms do not take any parameters. RT_1, [10], adds parameterized roles. Indeed, a role term takes the form $r(p_1, \dots, p_n)$, where each $p_i | 1 \le i \le n$ is a role parameter that can be either a constant or a variable, possibly ranging over a set. The policy statements are the same as in RT_0, except for the presence of the parameters. Thus, if one considers, for the sake of simplicity, role terms with single parameters, the policy statement for, *e.g.,* the simple member and the simple containment are as follows:

- $A.r(p) \leftarrow D$ (**simple member with parameter**)

- $A.r(p) \leftarrow A_1.r_1(p_1)$ (**simple containment with parameters**)

In the simple member statement A and D are possibly the same entities. In the simple containment statement A and A_1 are possibly the same entities and $r(p)$ and $r_1(p_1)$ are possibly the same role terms.

Example 3.2 Consider the following credential ([10]):

$$Alpha.evaluator(?employee) \leftarrow Alpha.manager(?employee)$$

This expresses the fact that a company *Alpha* allows the manager of an employee to evaluate an employee (the question mark expresses a variable).

In [9] it is also introduced a constraint-based extension of RT_1, called RT_1^C, that will be considered in Subsection 5.1. RT_1^C allows some kinds of costraints on parameters.

4 The Transitive Trust Model

Here, we give a formalization of the transitive trust model proposed by Jøsang *et al.*, [6,7]. Part of the formalization has been given in [14]. Here, we recall and extend it by introducing a rule for the management of the recommenders.

The formalization we are going to give describes the trust model within a series of inference rules, that will allow us to manage trust relationships by means of standard mechanisms (*i.e.,* with the same mechanisms used with process algebras and similar methodologies, see, *e.g.,* [5]). Moreover, by using an encoding based on inference rules, the comparison between the transitive trust model and the RT family of languages, which we are going to give in the following sections, will result more natural.

The transitive trust model in [6,7] has been introduced according to the consideration that trust is always linked to a purpose. The most natural situation is when one trusts another for performing a certain function/task. In the model, we denote $A \xrightarrow{f} D$ as the situation in which A trusts D for performing function f. Moreover, it is often common that principal A asks principal D to suggest/recommend a third one for doing a given task, *e.g.,* f. This could be expressed by credential $A \xrightarrow{rf} D$.

Thus, one can easily recognize a difference in the use of such credentials, the former involving a sort of *functional* trust (*e.g.,* A trusts D for actually doing something), the latter asserting trust towards a *recommendation*, *i.e.,* towards a third party's opinion about capabilities of someone else to do something.

In other words, when one calculates whether a given chain trust exists, it must always consider that the last step in the chain is a functional trust one, while possible other steps are recommendation steps. A third kind of credential in the model is $A \xrightarrow{r} B \xrightarrow{f} D$, asserting that A trusts D for performing f via the recommendation of B (there could be a chain of recommendations, not explicitly shown here).

Thus, the trust model can be described by the following rules:

$$A \xrightarrow{f} D \qquad (1)\ \text{FUNCTIONAL TRUST}$$

$$A \xrightarrow{rf} D \qquad (2)\ \text{RECOMMENDATION}$$

$$\frac{A \xrightarrow{rf} B \quad B \xrightarrow{rf} D}{A \xrightarrow{rf} D} \qquad (3)\ \text{TRANSITIVITY}$$

$$\frac{A \xrightarrow{rf} B \quad B \xrightarrow{f} D}{A \xrightarrow{r} B \xrightarrow{f} D} \qquad (4)\ \text{FUNCTIONAL TRUST VIA RECOMMENDATION}$$

$$\frac{A \xrightarrow{r} B \xrightarrow{f} D \quad A \xrightarrow{r} C \xrightarrow{f} D}{A \xrightarrow{r} \{B\} \cup C \xrightarrow{f} D}\ B \notin C \qquad (5)\ \text{SET OF RECOMMENDERS}$$

Rule (1) and (2) are the basic credentials of the model. Rule (3) expresses the transitivity of a recommending chain, *i.e.,* if A trusts B for a recommending for a certain purpose and B trusts D analogously, then A can directly trust D for recommending for that purpose. (Note that it is not required that direct trust is transitive in this model.) Rule (4) says that, if A trusts B for recommendation and B trusts D for doing f, then conclusion is that A trusts D for doing f through B's recommendation. The conclusion of rule (4) gives premises for a fifth rule, newly introduced by us. Rule (5) says that, if A trusts a set C of recommenders for recommending someone for doing f and A trusts recommender $B \notin C$ for the same purpose, then A trusts the enlarged set $\{B\} \cup C$. This rule permits to explicitly represent the sources of trust from A to D.

4.1 A Simplified Transitive Trust Model

Considering RT, it should be noted that no explicit relation appears, involving more than two principals. On the other hand, the transitive trust model can express both recommendation and direct functional trust, and, in particular, a trace still exists of the recommender (see, *e.g.,* rule (4), Section 4).

In our attempt to relate the two languages, and limiting ourselves, in this first attempt, to consider only the basic language RT_0, we should give coherence to the translation. Thus, an immediate idea is to simplify the trust model of Jøsang *et al.*. Actually, $A \xrightarrow{r} B \xrightarrow{f} D$ asserts that there is a sort of *indirect* functional trust from A to D via B. One can simplify this at the cost of losing information on recommenders.

Consequently, rule (4) is simplified as follows:

$$\frac{A \xrightarrow{rf} B \quad B \xrightarrow{f} D}{A \xrightarrow{f} D} \quad (4^*) \text{ INDIRECT FUNCTIONAL TRUST}$$

The newly introduced rule says that if A trusts B for recommendating a third one for doing f and B trusts D for performing f, then A trusts D for performing f.

Obviously, by adopting rule (4*) in place of rule (4), the simplified trust model consists of the three first rules and rule (4*), while rule (5) is not included into the simplified model (by losing information on recommenders it appears nonsense to have a rule for aggregating the recommenders).

4.2 Extending the Model with Trust Measures

The transitive trust model in [6,7] can be enriched with measures of trust. In such a way, credentials are enhanced in order to express not only the fact that a principal trusts someone for performing f or for a recommendation, but also they specify the degree of this trust. Then, by applying a rule, one can derive the trust measure of the resulting conclusion by adequately combining the trust measures of the premises.

As an example, consider a credential enhanced with a trust measure, *e.g.,* $A \xrightarrow{f,v} B$. This could be read as follows: A trusts B for performing f, with a certain degree v. Thus, v gives the measure of how much A places confidence in B. The extension of the model with measures holds also for the credentials about recommendation, *i.e.,* notation $A \xrightarrow{rf,v} B$ means: A trusts B for recommending someone else able to perform f, with a certain degree v. Trust measures are hereafter denoted as $v, v_1, v_2, \ldots, w$.

In [6,7], the authors suggest that there must be explicit rules for combining trust measures either when dealing with transitivity of trust (*e.g.,* rule (3)), or in presence of multiple paths (*e.g.,* rule (5)), or when dealing with a chain of recommendation steps followed by a last step concerning with functional trust (*e.g.,* rule (4) and (4*) for the simplified model).

Aiming at following the original intention of [6,7], we try to introduce the rules for combining the trust measures. Thus, we consider two operators, namely the *link* operator $\otimes$ and the *aggregation* operator $\odot$, for combining the trust measures. Generally speaking, the former is used to compose trust paths, while the latter is used to compare, select or aggregate trust paths.

In particular, we define the *link* operator $\otimes$ over rule (3) TRANSITIVITY and over rule (4) FUNCTIONAL TRUST VIA RECOMMENDATION, while the *aggregation* operator $\odot$ is defined over rule (5) SET OF RECOMMENDERS and, more generally, in case of multiple trust paths. Below, we show the model enriched with trust measures and rules for combining them.

$$A \xrightarrow{f,v} D \qquad (1)$$

$$A \xrightarrow{rf,v} D \qquad (2)$$

$$\frac{A \xrightarrow{rf,v_1} B \quad B \xrightarrow{rf,v_2} D}{A \xrightarrow{rf,v_1 \otimes v_2} D} \qquad (3)\ \text{LINK 1}$$

$$\frac{A \xrightarrow{rf,v} B \quad B \xrightarrow{f,w} D}{A \xrightarrow{r} B \xrightarrow{f,v \otimes w} D} \qquad (4)\ \text{LINK 2}$$

$$\frac{A \xrightarrow{r} B \xrightarrow{f,v} D \quad A \xrightarrow{r} C \xrightarrow{f,w} D}{A \xrightarrow{r} \{B\} \cup C \xrightarrow{f,v \odot w} D}\ B \notin C \qquad (5)\ \text{AGGREGATION - SET OF RECOMMENDERS}$$

$$\frac{A \xrightarrow{f,v_1} B \quad A \xrightarrow{f,v_2} B}{A \xrightarrow{f,v_1 \odot v_2} B} \qquad (6)\ \text{AGGREGATION}$$

Remark 4.1 In order to have a coherent semantics, some conditions must be imposed on the link and aggregation operators. The first, natural, requirement is that $\otimes$, denoting transitivity, must be associative, thus allowing to compute trust along complex paths without taking care of the sub-paths evaluation order. Similarly, $\odot$ must be, in addition, also commutative, being the aggregation among two paths not sensitive to operand order. Another desirable feature is that aggregation should be idempotent. Some authors (*e.g.,* see [19]) noticed that semirings, *i.e.,* a triple $(M, \odot, \otimes)$, where $\odot$ and $\otimes$ enjoy at least the previous properties plus additional ones (as the distributivity of $\otimes$ over $\odot$), could be used as a starting point to model trust operators. This is not a surprise, since these structures are commonly used for calculating weighted paths on graphs[3]. However, other features may be required that do not however change the overall kind of reasoning. For instance, an additional requirement could be that trust degrades with links, thus requiring that $m_1 \otimes m_2 \leq m_1, m_2$ for a suitable $\leq$. In the following, we will use some instances of these structures.

Example 4.2 LINK 1. Assume the interval $[0, 1]$ of real numbers with the usual multiplication operator, here denoted as $\otimes$, and the operator $\odot$ that returns the maximum between two real numbers. Suppose that A trusts B's opinion for recommending someone able to perform f, with a certain degree v_1=0.8. Suppose also that B trusts D as a recommender for f with degree v_2=0.7. Then, conclusion is that A can directly trusts D as a recommender with a measure $v_3 = v_1 \otimes v_2$, where $\otimes$ can be, *e.g.,* the product operator. In this case, v_3 =0.56, according to the intuition that trust is transitive but decreasing.

Example 4.3 LINK 2. Suppose that A trusts B's opinion for recommending someone able to perform f, with a certain degree v. Suppose also that B trusts D for performing f with degree w. Conclusion must take into account both the level of trust towards the recommender and the level of functional trust. Thus, A can trusts D for performing f via B's recommendation, with trust measure $v \otimes w$, where $\otimes$ is, again, the product operator. Also in this case, indeed, a direct trust is greater than trust derived trough a recommendation (assuming that a trust measure is a positive value $v \; |0 \leq v \leq 1$).

Example 4.4 AGGREGATION. Suppose that A trusts B as a good cook, with two possible measures, v_1 and v_2. This could be possible since A could have collected two possible opinions about the goodness of B as a cook, *e.g.,* by means of recommenders (not explicitly highlighted in the rule). Thus, A could choose to maintain a single credential regarding B, by combining the two measures according to some operator $\odot$. For example, A could simply consider the maximum value between v_1 and v_2.

4.3 Extending the Simplified Model with Trust Measures

In this subsection, we show the simplified model of Subsection 4.1 enriched with trust measures and rules for combining them. Rules (1), (2), (3) and the general rule (6) are the same as in the above subsection, thus we omit to explicitly put them. Clearly, rules (4) and (5) do not exist in the simplified model. Here, we define the *link* operator over rule (4*).

$$\frac{A \xrightarrow{rf, v_1} B \quad B \xrightarrow{f, v_2} D}{A \xrightarrow{f, v_1 \otimes v_2} D} \; (4^*) \; \text{Link 3}$$

[3] See, *e.g.,* [2] for other applications of semiring based techniques to security analysis.

5 Encoding the Simplified Transitive Trust Model into (Part of) RT_0

Here, we show how to encode the simplified version of the transitive trust model into part of RT_0.

	SIMPLIFIED TRUST MODEL	RT_0
(1)	$A \xrightarrow{f} D$	$A.f \leftarrow D$
(2)	$A \xrightarrow{rf} D$	$A.rf \leftarrow D$
(3)	$\dfrac{A \xrightarrow{rf} B \quad B \xrightarrow{rf} D}{A \xrightarrow{rf} D}$	$A.rf \leftarrow A.rf.rf$
(4∗)	$\dfrac{A \xrightarrow{rf} B \quad B \xrightarrow{f} C}{A \xrightarrow{f} C}$	$A.f \leftarrow A.rf.f$

(i) The functional trust step of the model can be encoded into the simple member role definition of RT_0. The role term f represents the capability of performing a service or a functionality that D is able to perform according to A. Thus, in $A.f \leftarrow D$, A defines D to have role f, *i.e.,* that D is able to perform f.

(ii) The recommendation step is again a simple member role definition in RT_0. Here, the role term rf represents the recommendation for doing a certain service or functionality. In $A.rf \leftarrow D$, A defines D to have role rf, *i.e.,* that D is trusted for giving a recommendation to perform f.

(iii) The transitivity step is encoded into RT_0 by a linking containment of the form $A.rf \leftarrow A.rf.rf$. This statement says that if A defines B to have role $A.rf$, and B defines D to have role $B.rf$, then A defines D to have role $A.rf$, *i.e.,* D is trusted to act as a recommender according to A. Thus, the role term rf represents the recommendation.

(iv) The indirect functional trust step is encoded into RT_0 by a linking containment of the form $A.f \leftarrow A.rf.f$. $A.f \leftarrow A.rf.f$ says that if B has role $A.rf$ and C has role $B.f$ then C has role $A.f$. B who has role $A.rf$ is the recommender, *i.e.,* A trusts B for choosing someone else that is trusted for performing f. C who has role $B.f$ is trusted to perform f according to B. It follows that C is *indirectly* trusted to perform f according to A.

It is worthwhile noticing that the other kinds of credentials are not meaningful. Indeed, in the simplified model we loose the information on the recommender. Thus, RT_0 becomes suitable to express interactions between the parties.

Remark 5.1 On a completely different perspective, we may note that for the encoding in RT_0, we just need the "simple member" and "linked containment" kinds of credentials. This suggested us to investigate whether the set of credentials of RT_0 is minimal or not. As a matter of fact, we noticed that simple containment may be obtained by means of simple member and linked containment credentials, at the cost of adding more roles. To express credential $A.r \leftarrow B.r_1$, we may just add a special role r_B and consider the two credentials $A.r_B \leftarrow B$ and $A.r \leftarrow A.r_B.r_1$.

5.1 *Encoding the Simplified Transitive Trust Model with Trust Measures into (Part of) RT_1^C*

The simplified transitive trust model with trust measures can be encoded into RT, by exploiting the parameterized (and constrained) roles of RT_1^C. Indeed, trust measures can be inserted as parameters:

	SIMPLIFIED TRUST MODEL WITH TRUST MEASURES	RT_1
(1)	$A \xrightarrow{f,v} D$	$A.f(v) \leftarrow D$
(2)	$A \xrightarrow{rf,v} D$	$A.rf(v) \leftarrow D$
(3)	$\dfrac{A \xrightarrow{rf,v_1} B \quad B \xrightarrow{rf,v_2} D}{A \xrightarrow{rf,v_1 \otimes v_2} D}$	$A.rf(v_1 \otimes v_2) \leftarrow A.rf(v_1).rf(v_2)$
(4∗)	$\dfrac{A \xrightarrow{rf,v_1} B \quad B \xrightarrow{f,v_2} C}{A \xrightarrow{f,v_1 \otimes v_2} C}$	$A.f(v_1 \otimes v_2) \leftarrow A.rf(v_1).f(v_2)$
(6)	$\dfrac{A \xrightarrow{f,v_1} B \quad A \xrightarrow{f,v_2} B}{A \xrightarrow{f,v_1 \odot v_2} B}$	$A.f(v_1 \odot v_2) \leftarrow A.f(v_1) \cap A.f(v_2)$

6 Crypto-CCS as a *Trait d'Union*

We now show how the inference construct of Crypto-CCS can naturally model the encoding given in the previous section. For the sake of simplicity, we refrain from reporting the correspondent credentials in RT and in the simplified trust model, and we just report the correspondent labels.

The basic credentials of the transitive trust model (*i.e.,* the functional trsut and recommendation credential) and of RT_0 (the simple member credential) can be represented into Crypto-CCS as self-signed certificates of the following form:

$$\{A, D, f\}_{A^{-1}}$$

In this formalization, A, A^{-1} are, respectively, the public and private key of A. Here, with a little abuse of notation, we overlap an identity and a public key. In a self signed certificate, the person that created the certificate also signed off on its legitimacy.

For the sake of readability, in the following we omit the identity of the authority in the body of the signature, *e.g.,* we write $\{D, f\}_{A^{-1}}$ instead of $\{A, D, f\}_{A^{-1}}$.

First, we show the inference system that models the encoding given in Section 5, without trust measures.

$$
\begin{array}{ll}
(1) & \{D, f\}_{A^{-1}} \\
(2) & \{D, rf\}_{A^{-1}} \\
(3) & \dfrac{\{B, rf\}_{A^{-1}} \quad \{D, rf\}_{B^{-1}}}{\{D, rf\}_{A^{-1}}} \\
(4*) & \dfrac{\{B, rf\}_{A^{-1}} \quad \{C, f\}_{B^{-1}}}{\{C, f\}_{A^{-1}}}
\end{array}
$$

Rule (1) and (2) can be expressed into Crypto-CCS as self-signed certificates where the private key of principal A speaks for A itself. For example, the term $\{D, f\}_{A^{-1}}$ says that A claims that D is trusted for performing f. The way through which A guarantees for that assertion is the signature that is affixed. Indeed, as it is common in security protocols modeling and analysis, it is assumed that the private key of a principal is never disclosed. Thus, A was the unique able to generate such a signature.

Based on (1) and (2), one can express also transitivity of recommendation (3) and indirect functional trust (4*).

Then, we show the inference system for the encoding of Subsection 5.1, with trust measures.

$$
\begin{array}{ll}
(1) & \{D, f, v\}_{A^{-1}} \\
(2) & \{D, rf, v\}_{A^{-1}} \\
(3) & \dfrac{\{B, rf, v_1\}_{A^{-1}} \quad \{D, rf, v_2\}_{B^{-1}}}{\{D, rf, v_1 \otimes v_2\}_{A^{-1}}} \\
(4*) & \dfrac{\{B, rf, v_1\}_{A^{-1}} \quad \{C, f, v_2\}_{B^{-1}}}{\{C, f, v_1 \otimes v_2\}_{A^{-1}}} \\
(6) & \dfrac{\{B, f, v_1\}_{A^{-1}} \quad \{B, f, v_2\}_{A^{-1}}}{\{B, f, v_1 \odot v_2\}_{A^{-1}}}
\end{array}
$$

7 Conclusions and Future Work

In this paper, we investigated the possibility to use Crypto-CCS as a suitable modeling language, not only for the standard capability of its inference rules to model message exchange and manipulation, but also, and hopefully, for its expressiveness in defining, along with its native security features, credential chains, trust and recommendation relationships.

Along with Crypto-CCS, we considered part of two languages for defining trust and recommendation relationships, *RT* and the transitive trust model. By simplifying the latter, we have shown a comparison between them, leading to the result that, to some extent, part of one framework can be mapped into part of the other.

Also, the Crypto-CCS inference systems can express the resulting framework, with new peculiarities representing levels of trust too.

The current investigation is an initial one, and further research is needed in order to provide a more solid framework. In particular, we plan to extend our work providing encoding correctness guarantees.

Moreover, here a simplified version of the trust model has been considered, in order to make it compliant with RT. This has the clear drawback that some information is lost (in particular, information on the recommenders, that in some application can be relevant), with respect to the original intention of [6,7]. Thus, refinements to the current work are possible through, *e.g.,* the study of an extension to RT dealing with recommenders. Furthermore, an encoding has been presented from the simplified trust model to RT. The viceversa could also be considered, by studying how to deal with aspects like intersection of roles.

Upon providing refinements, we plan to extend the framework for enforcing security policies (whose formal semantics is based on a variant of Crypto-CCS) in GRID systems (*i.e.,* see [8]) with mechanisms for managing also reputation and recommendation information.

Acknowledgement

We would like to thank the anonymous reviewers for their helpful comments.

The work presented here was partly supported by EU project IST-3-016004-IP-09 Sensoria (*Software Engineering for Service Oriented Overlay Computers*) and by EU network of excellence IST-004527 ARTIST2 on Embedded Systems Design.

References

[1] Basagni, S., M. Conti, S. Giordano and I. Stojmenovic, *Mobile Ad Hoc Networking*, Chapter 12 of Ad Hoc Networks Security (2004), pp. 329–354.

[2] Bella, G., S. Bistarelli and S. N. Foley, *Soft Constraints for Security*, in: M. H. ter Beek and F. Gadducci, editors, *Proceedings VODCA 2004*, ENTCS **142**, 2006, pp. 11–29.

[3] Blaze, M., J. Feigenbaum and J. Lacy, *Decentralized Trust Management*, in: *IEEE Symposium on Security and Privacy*, 1996, pp. 164–173.

[4] Camara, J., C. Canal and J. Cubo, *Issues in the formalization of web service orchestration*, in: *WCAT*, Tech. Rep. TR ITI-05-07. Dept. of Computer Science. School of Informatics, Malaga University, 2005, pp. 17–24.

[5] Focardi, R., R. Gorrieri and F. Martinelli, *Classification of Security Properties—Part II: Network Security*, in: *FOSAD 2001/2002—Tutorial Lectures*, LNCS **2946** (2004), pp. 139–185.

[6] Jøsang, A., L. Gray and M. Kinateder, *Analysing Topologies of Transitive Trust*, in: *Proceedings FAST, Tech. Rep. IIT TR-10/2003*, 2003, pp. 9–22.

[7] Jøsang, A., L. Gray and M. Kinateder, *Simplification and analysis of transitive trust networks*, Web Intelligence and Agent Systems **4** (2006), pp. 139–161.

[8] Koshutanski, H., F. Martinelli, P. Mori and A. Vaccarelli, *Fine-grained and History-based Access Control with Trust Management for Autonomic Grid Services*, in: *Proceedings ICAS* (2006), pp. 34–44.

[9] Li, N. and J. C. Mitchell, *DATALOG with Constraints: A Foundation for Trust Management Languages*, in: *Proceedings PADL*, LNCS **2562**, 2003, pp. 58–73.

[10] Li, N., J. C. Mitchell and W. H. Winsborough, *Design of a role-based trust management framework*, in: *Proceedings S&P* (2002), pp. 114–130.

[11] Li, N., W. H. Winsborough and J. C. Mitchell, *Distributed credential chain discovery in trust management*, Journal of Computer Security **1** (2003), pp. 35–86.

[12] Marchignoli, D. and F. Martinelli, *Automatic Verification of Cryptographic Protocols through Compositional Analysis Techniques*, in: *Proceedings TACAS*, LNCS **1579** (1999), pp. 148–162.

[13] Martinelli, F., *Analysis of security protocols as open systems*, Theoretical Computer Science **290** (2003), pp. 1057–1106.

[14] Martinelli, F., *Towards an Integrated Formal Analysis for Security and Trust*, in: *Proceedings FMOODS*, LNCS **3535** (2005), pp. 115–130.

[15] Milner, R., "Communication and Concurrency," Prentice Hall, 1989.

[16] Olmedilla, D., O. F. Rana, B. Matthews and W. Nejdl, *Security and Trust Issues in Semantic Grids*, in: *Semantic Grid: The Convergence of Technologies*, Dagstuhl Seminar Proceedings **05271** (2005).

[17] Salaün, G., L. Bordeaux and M. Schaerf, *Describing and Reasoning on Web Services using Process Algebra*, in: *Proceedings ICWS* (2004), pp. 43–51.

[18] Sandhu, R. S., E. J. Coyne, H. L. Feinstein and C. E. Youman, *Role-based access control models*, IEEE Computer **2** (1996), pp. 38–47.

[19] Theodorakopoulos, G. and J. S. Baras, *Trust evaluation in ad-hoc networks*, in: *Proceedings WiSe* (2004), pp. 1–10.

[20] Winslett, M., *An Introduction to Trust Negotiation*, in: *Proceedings iTrust*, LNCS **2692**, 2003, pp. 275–283.