Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science 186 (2007) 85–99
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

# A Uniform Framework for Security and Trust Modeling and Analysis with Crypto-CCS[1]

Fabio Martinelli[2] and Marinella Petrocchi[3]

*Istituto di Informatica e Telematica - C.N.R., Pisa, Italy*

**Abstract**

In this paper, we present our line of research for defining an integrated framework for the specification and analysis of security and trust in complex and dynamic scenarios. We aim at showing how the same machinery applied for the formal verification of security protocols may be useful to model and analyze trust management procedures.

*Keywords:* Security and trust analysis, formal verification, security protocols

## 1 Introduction

In pervasive and autonomous computing systems, security issues are in deep contact with cooperation aspects. Indeed, these systems consist of different entities that have to cooperate and share resources to achieve a certain goal. On its turn, cooperation is often related to trust, being the entities at stake more inclined to collaborate with parties that they trust.

In this paper we focus on the integration of formal modeling and analysis of security and trust. In particular, we present our framework to uniformly model security protocols and some form of trust and reputation management procedures, like Automated Trust Negotiation (ATN, [18]). We thus show that a strong connection exists between methodologies used to establish, manage and negotiate trust and the security mechanisms used to guarantee the confidentiality and integrity of information.

Formal theories, languages and tools have been successfully applied for the analysis of network security. Exploiting formal methods, the protocol under investigation

[1] Invited talk. Extended and revised version of [12,13].

[2] Email: Fabio.Martinelli@iit.cnr.it

[3] Email: Marinella.Petrocchi@iit.cnr.it

doi:10.1016/j.entcs.2007.03.024

is described in a given language, then a formal specification of the security property to be analyzed is defined.

In this framework, cryptography is usually modeled by representing encryptions as terms of an algebra, *e.g.,* $m_k$ may represent the encryption of a message $m$ with a key $k$. Usually, the inverse operation, decryption, can be performed only by knowing the correct information, *i.e.,* the decryption key.

Process algebras have been traditionally apply for modeling cryptographic functions and powerful verification techniques based on process algebras have been extensively used for analysis purposes. As an example, CCS, [14], has been equipped with an inference construct that permits to infer new messages from others, *i.*e.: $[m_1 \ldots m_n \vdash_r x].P$ which denotes a process that tries to deduce a message $m$ from the messages in $m_1, \ldots, m_n$. The language is called Crypto-CCS ([9,11]).

Trying to unify trust features in a pre-existent framework already dealing with security is particularly appealing for our aims. Indeed, when one analyzes a security protocol, usually assumes that public keys, digital certificates, and generally speaking credentials are already given, and does not usually check how these are formated, negotiated and managed. Such a limited view seems not completely appropriate for dynamic, fully interconnected systems, where entering some resources may depend on credentials presented by users.

Also, 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.

Thus, inference constructs may be suitably used not only to cope with the variety of different crypto-systems that can be found in the literature, but also to express trust towards others, by means of direct experience, or it can be used to infer trust relationships through recommendations of third services.

Indeed, consider a set of credentials, *i.e.,* (signed) messages containing some information regarding roles, capabilities or rights that someone has. Assume that $\{B, rf\}_{A^{-1}}$ means that the user $A$ (via the signature with its private key $pk_A^{-1}$) asserts $B$ has capability to recommend a third party for doing a certain functionality $f$. Also, assume that $\{D, f\}_{B^{-1}}$ means that $B$ asserts (again, through its signature) $D$ has capability to do $f$. A rule like:

$$\frac{\{B, rf\}_{A^{-1}} \quad \{D, f\}_{B^{-1}}}{\{D, f\}_{A^{-1}}}(rec)$$

may be used by $A$ to find some people (*e.g.,* $D$) that is able to do $f$. Also we could consider that each credential has a weight expressing the degree of trust of one entity on the other for performing the function $f$. How trust values are calculated depend on the specific inference rules.

Once we are able to model both security protocols and trust management policies in a unique framework, we are also able to easily relate different concepts. For instance, we will formally show how the notion of privacy in trust negotiation is related to a well known concept in security, *i.e.,* non interference. Similarly, once could apply analysis methods developed for security protocols for trust management problems and vice-versa.

The structure of the paper is as follows. Next section recalls the Crypto-CCS language. In Section 3, we rephrase, in a formalization more suitable for our purposes, the transitive trust model proposed by [5,6]. We show an extension by introducing levels of trust and operators to combine them, according to the original intention of [5]. Section 4 briefly recalls the simplest language of the $RT$ family of languages for describing role-based credentials extended with measures as well as an encoding of the simplified Josang's model in the $RT_0$. Section 5 gives hints to an implementation of the inference system with levels of trust for $RT_0$ language. Section 6 shows how Crypto-CCS can model the systems presented in the above sections and it gives an application example. Then, in Section 7, applications and results in the field of Automated Trust Negotiation are shown. Finally, Section 8 gives some final remarks.

## 2 Crypto-CCS

Here, we recall syntax and semantics of the formal language Crypto-CCS ([8,11]). 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:

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

where $m$ ranges over *Msgs*, $F^i$ (for $1 \le i \le l$) 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 $1 \le i \le l$, 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 as:

$$rule = \frac{m_1 \quad \dots \quad m_n}{m_0}$$

where $m_1, \dots, m_n$ are premises (possibly empty) and $m_0$ is the conclusion. An instance of the application of the rule $r$ to closed messages $m_1, \dots, m_n$ is denoted as $m_1 \quad \dots \quad m_n \vdash_{rule} m_0$. For each rule $r$ and set of closed messages $\{m_1, \dots m_n\}$, we assume that the set $\{m \mid m_1, .., m_n \vdash_{rule} m\}$ 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:

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

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

where $m, m_1, \ldots, m_r$ are 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 [14]. We call $\mathcal{P}$ the set of all the Crypto-CCS closed and guarded terms. The set of actions is $Act = \{c(m) \mid c \in I\} \cup \{\overline{c}m \mid \overline{c} \in O\} \cup \{\tau\}$ ($\tau$ 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 $P[m/x]$.
- $\overline{c}m.P$ is the process that can send $m$ on channel $c$, and then behaves like $P$.
- $\tau.P$ is the process that executes the invisible $\tau$ 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 $\tau$.
- $P\backslash L$ is the process that cannot send and receive messages on channels in $L$; for all the other channels, it behaves exactly like $P$;
- $A(m_1, \ldots, m_r)$ behaves like the respective defining term $P$ where all the variables $x_1, \ldots, x_r$ are replaced by the closed messages $m_1, \ldots, m_r$;
- $[\langle m_1, \ldots, m_r\rangle \vdash_{rule} x]P; Q$ is the process used to model message manipulation as cryptographic operations. Indeed, the process $[\langle m_1, \ldots, m_r\rangle \vdash_{rule} x]P; Q$ tries to deduce an information $z$ from the tuple $\langle m_1, \ldots, m_r\rangle$ 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:

$$\frac{x \quad x}{Equal(x, x)}\, equal$$

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.

### 2.1 Security protocol analysis

The security protocol analysis proposed in [9,11] is based on the checking of following property:

$$\forall X \text{ s.t. } S \,|\, X \text{ satisfies } F$$

where $F$ is a logical formula expressing the desired property. Often, when secrecy properties are considered, $F$ models the fact that a given message, *i.e.,* the secret to be verified, is not deducible from a given set of messages, *i.e.,* the knowledge of the intruder $X$ acquired during the computation with $S$. The verification of such property requires the ability of computing the closure of an inference system, *i.e.,* the possibility to iteratively apply the inference rules. Given a set $\mathcal{R}$ of inference rules, we consider the deduction relation $\mathcal{D}^{\mathcal{R}} \subseteq \mathcal{P}^{fin}(\mathcal{M}sgs) \times \mathcal{M}sgs$. Given a finite set of closed messages, say $\phi$, then $(\phi, M) \in \mathcal{D}^{\mathcal{R}}$ if $M$ can be derived by iteratively applying the rules in $\mathcal{R}$. Under certain sets of assumptions on the form of the rules, we may have that $\mathcal{D}^{\mathcal{R}}(\phi)$ is decidable (see, *e.g.,* [10]).

## 3 The (simplified) transitive trust model (with measures)

In [12,13] we have given a formalization based on inference rules of the transitive trust model proposed by Jøsang *et al.* (*e.g.,* see [5,6]). The transitive trust model in [5,6] has been introduced to model trust relationships towards performing a certain function/task or recommending someone else for performing a certain function/task. As an example, one can denote $A \xrightarrow{f} D$ the situation in which $A$ trusts $D$ for performing function $f$, or $A \xrightarrow{rf} D$ the situation in which $A$ trusts principal $D$ for suggesting/recommending a third one for given task $f$. Here we adopt a *simplified* version of this model where the recommender information is not crucial.

In addition, the model can be naturally enriched with trust measures. 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

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

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

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

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

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

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

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

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

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

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 $v, v_1, v_2, v_3$.

In [5,6], the authors suggest that there must be explicit rules for combining trust measures either when dealing with transitivity of trust (*e.g.,* when dealing with chains of recommendation), or in presence of multiple paths (*e.g.,* when dealing with more recommenders recommending the same target), or when dealing with a chain of recommendation followed by a credential recommending the functionality itself. 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.

Below, we show a proposal for a model enriched with trust measures and rules for combining them.

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) AGGREGATION and, more generally, in case of multiple trust paths.

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

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

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

$$\frac{A \xrightarrow{rf,v_1} B \quad B \xrightarrow{f,v_2} D}{A \xrightarrow{f,v_1 \otimes v_2} D} \qquad (4) \text{ FUNCTIONAL TRUST VIA RECCOMENDATION}$$

$$\frac{A \xrightarrow{f,v_1} B \quad A \xrightarrow{f,v_2} B}{A \xrightarrow{f,v_1 \odot v_2} B} \qquad (5) \text{ AGGREGATION}$$

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 with measure $v_1$ and $B$ trusts $D$ analogously, with measure $v_2$, then $A$ can directly trust $D$ for recommending for that purpose, with measure $v_1 \otimes v_2$. 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.

Rule (4) says that if $A$ trusts $B$ for recommendating a third one for doing $f$ (with measure $v_1$) and $B$ trusts $D$ for performing $f$ (with measure $v_2$), then $A$ trusts $D$ for performing $f$, with resulting measure[4] .

Finally, rule (5) gives a resulting measure combining two measures belonging to the same target. As an example, suppose that $A$ trusts $B$ as a good cook, with two possible measures, $v_1$ and $v_2$. Thus, $A$ can maintain a single credential regarding $B$, by combining the two measures according to some operator $\odot$, *e.g.,* $A$ could simply consider the maximum value between $v_1$ and $v_2$.

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 [16]) 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

[4] This is a simplified version of the original rule that does not keep the information on the recommender, thus resulting very similar to the rule 3.

calculating weighted paths on graphs. Also 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 so-called c-semirings[5] that enjoy our desired properties.

**Example 3.1** We give some specific examples of semiring-based trust measures. Assume that we want to consider the path with maximal trust, then:

- $\otimes$ is the multiplication on real numbers (for instance between 0 and 1);
- $\odot$ is the maximum between two real numbers.

We want to consider the path with the minimal weak steps in the trust paths:

- $\otimes$ is the minimum between two real numbers (for instance between 0 and 1);
- $\odot$ is the maximum between two real numbers.

We want to consider the path with minimal number of steps in the trust paths:

- $\otimes$ is the sum on natural numbers;
- $\odot$ is the minimum between two natural numbers.

## 4 $RT_0$ with trust measures

In [7], Li et al. introduced the *RT* family of languages for managing and expressing trust relationships among entities. In the *RT* family of languages, credentials carry information on policies to define attributes of principals by starting from assertions of other principals. *RT* combines the strength of Role-based Access Control (RBAC), [15], 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 $A$ defines who are the members of each role of the form $A.r$ by using specific credentials.

We give an extension of the simplest language of the *RT* family, *i.e.,* $RT_0$, with trust measures (weights). We recall here the credentials and their meaning[6]. We will also use the trust operators described in the previous section.

- $A.r(v) \leftarrow D$ (**simple member with measure**)
  The statement defines that $D$ is member of role $A.r$ with weight $v$.
- $A.r \leftarrow_{v_2} A_1.r_1$ (**simple containment**)
  $A$ and $A_1$ are possibly the same entities and $r$ and $r_1$ are possibly the same role terms. With this statement, $A$ defines all members of role $A_1.r_1$ with weight $v_1$ to be members of role $A.r$ with weight $v = v_1 \otimes v_2$.

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

[6] The notation is similar to the usual one for $RT_0$ credentials, but the actual semantics should be clear from the context.

- $A.r \leftarrow A.r_1.r_2$ (**linking containment**)
  This statement defines a linked role. If $B$ has role $A_1.r_1$ with weight $v_1$ and $D$ has role $B.r_2$ with weight $v_2$, then $D$ has role $A.r$ with weight $v = v_1 \otimes v_2$. Note that this works as a sort of role-based delegation.
- $A.r \leftarrow A_1.r_1 \cap A_2.r_2$ (**Intersection containment**)
  This statement defines that if $D$ has both roles $A_1.r_1$ with weight $p_1$ and $A_2.r_2$ with weight $v_2$, then $D$ has role $A.r$ with weight $v = v_1 \odot v_2$.

### 4.1 *Encoding the simplified transitive trust model with trust measures into (part of) $RT_0$ with measures*

The simplified transitive trust model with trust measures can be encoded into $RT_0$ with measures as follows:

| | SIMPLIFIED TRUST MODEL WITH TRUST MEASURES | $RT_0$ WITH TRUST MEASURES |
|---|---|---|
| (1) | $A \xrightarrow{f,v} D$ | $A.f(v) \leftarrow D$ |
| (2) | $A \xrightarrow{rf,v} D$ | $A.rf(v) \leftarrow D$ |
| (3) | $\dfrac{A \xrightarrow{rf,v_1} B \quad B \xrightarrow{rf,v_2} D}{A \xrightarrow{rf,v_1 \otimes v_2} D}$ | $A.rf \leftarrow A.rf.rf$ |
| (4) | $\dfrac{A \xrightarrow{rf,v_1} B \quad B \xrightarrow{f,v_2} C}{A \xrightarrow{f,v_1 \otimes v_2} C}$ | $A.f \leftarrow A.rf.f$ |
| (5) | $\dfrac{A \xrightarrow{f,v_1} B \quad A \xrightarrow{f,v_2} B}{A \xrightarrow{f,v_1 \odot v_2} B}$ | $A.f \leftarrow A.f \cap A.f$ |

## 5 An implementation of $RT_0$ with trust measures

We present an algorithm for calculating a set of simple member credentials with trust measures on $RT_0$. It takes as input the credentials available, split in two sets, the set of basic credentials and the others. If one does not consider trust measures, the algorithm basically builds the minimal set of simple member credentials by iteratively applying the inference rules for each kind of credential. If the inferred credential does not belong yet to the set of computed basic credentials, then it is added to this set. The procedure is iterated until no new credentials are found. When the algorithm is applied to a finite set of credentials, it correctly terminates. Adding weights is possible. As a matter of fact, due to the specific nature of c-semiring we are going to apply, it can be also seen as a variant of the Floyd algorithm for calculating minimal/maximal weighted paths among all the nodes in a graph. Indeed, a simple member credential, say $A.r(v) \leftarrow C$, states that between the node $A$ and the node $C$ there is an arc labeled $r$ and with measure $v$. If we assume the order $\leq_w$ among weights defined as $v_1 \leq_w v_2$ iff $v_1 \odot v_2 = v_2$, then the algorithm computes the greatest weighted path (w.r.t. $\leq_w$) in the graph. We remind that in c-semiring $\otimes$ is an inclusive operation.

The algorithm is as follows.

```
Trust Calculations (basic creds, rules)= {
   Results:=basic creds; Changed := true;
   While(Changed) {
     Changed:=false;
     For each credential $A.r \leftarrow_{v_2} A_1.r_1$ in rules and for each credential
     $A_1.r_1(v_1) \leftarrow C$ in basic creds
       if $A.r \leftarrow C$ not in basic creds, or $A.r(v) \leftarrow C$ in basic creds with
       not $v_1 \otimes v_2 \leq_w v$
          then {remove from basic creds all the creds like $A.r \leftarrow C$;
          insert $A.r(v_1 \otimes v_2) \leftarrow C$ in basic creds; Changed:=true};
     For each credential $A.r \leftarrow A.r_1.r_2$ in rules and for each credential
     $A.r_1(v_1) \leftarrow B$, $B.r_2(v_2) \leftarrow C$ in basic creds
       if $A.r \leftarrow C$ not in basic creds, or $A.r(v) \leftarrow C$ in basic creds with
       not $v_1 \otimes v_2 \leq_w v$
          then {remove from basic creds all the creds like $A.r \leftarrow C$;
          insert $A.r(v_1 \otimes v_2) \leftarrow C$ in basic creds; Changed:=true};
     For each credential $A.r \leftarrow A_1.r_1 \cap A_2.r_2$ in rules and for each credential
     $A_1.r_1(v_1) \leftarrow C$, $A_2.r_2(v_2) \leftarrow C$ in basic creds
       if $A.r \leftarrow C$ not in basic creds, or $A.r(v) \leftarrow C$ in basic creds with
       not $v_1 \odot v_2 \leq_w v$
          then {remove from basic creds all the creds like $A.r \leftarrow C$;
          insert $A.r(v_1 \odot v_2) \leftarrow C$ in basic creds; Changed:=true};
}
```

## 6 Embedding in Crypto-CCS

The inference construct of Crypto-CCS can express the transitive trust model (and clearly also the $RT$ one), *e.g.,* see [12,13]. We present an example of a Crypto-CCS process that uses such rules.

Indeed, a basic credential of the transitive trust model and of $RT_0$ (*i.e.,* the simple member credential) can be represented into Crypto-CCS as certificates of the following form:

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

In this formalization, $A^{-1}$ is the private key of $A$.

We show the inference system that models the transitive trust model given in Section 3.

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

$$(2)\ \{D, rf, v\}_{A^{-1}}$$

$$(3)\ \frac{\{B, rf, v_1\}_{A^{-1}} \quad \{D, rf, v_2\}_{B^{-1}}}{\{D, rf, v_1 \otimes v_2\}_{A^{-1}}}$$

$$(4)\ \frac{\{B, rf, v_1\}_{A^{-1}} \quad \{C, f, v_2\}_{B^{-1}}}{\{C, f, v_1 \otimes v_2\}_{A^{-1}}}$$

$$(5)\ \frac{\{B, f, v_1\}_{A^{-1}} \quad \{B, f, v_2\}_{A^{-1}}}{\{B, f, v_1 \odot v_2\}_{A^{-1}}}$$

In all the credentials, the private key of principal $x$ speaks for $x$ itself. For example, the term $\{D, f, v\}_{A^{-1}}$ says that $A$ claims that $D$ is trusted for performing $f$ with measure $v$. 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 rules (4) and (5).

**Example 6.1** Consider the following scenario. An employer $E$ starts the interviews for engaging a new clerk. The criterion for the evaluation is as follows.

A candidate $C$ must pass through an interview through which the employer evaluates $C$'s capabilities in doing the work $f$. Thus, at the end of the interview, $E$ will trust $C$ for performing $f$ with a certain measure $v_E$. This direct opinion of $E$ is expressed by the following credential: $\{C, f, v_E\}_{E^{-1}}$.

Also, $E$ requires a letter of reference coming from a previous employer $PE$. The letter could be sent by e-mail directly from $PE$. It is digitally signed, in order to prove its authenticity of origin, and it is expressed by the following formal form: $\{C, f, v_{PE}\}_{PE^{-1}}$. The credential says that $PE$ trusts $C$ for doing $f$ with measure $v_{PE}$.

We assume that the $PE$'s public key is publicly known. By accepting a letter of reference from $PE$, $E$ is making the following assertion: $\{PE, rf, 1\}_{E-1}$, *i.e.,* $E$ completely trusts $PE$ for recommending someone able to perform $f$.

Thus, by applying rule 4 to premises $\{PE, rf, 1\}_{E-1}$ and $\{C, f, v_{PE}\}_{PE^{-1}}$, conclusion is $\{C, f, v_{PE}\}_{E^{-1}}$, *i.e.,* $E$ has an indirect opinion about trusting $C$ for doing $f$ via $PE$. Here, $\otimes$ is the product operator, and 1 is the neutral element, thus $v_{PE} \otimes 1$ is $v_{PE}$.

Currently, $E$ has two possible measures of trust towards $C$, *i.e.,* $v_{PE}$ and $v_E$. Through rule (5), $E$ can compare the two measures by obtaining a final level of trust $v_{PE} \odot v_E$. Note that the criterion for the comparison is not explicitly defined, since it is up to $E$'s belief instantiating some particular operator for $\odot$.

Finally, $E$ decides to engage $C$ if $v_{PE} \odot v_E$ is greater (or equal) than a certain threshold $v_t$.

When there are many candidates, and more than one candidate goes over $v_t$, one may think to ask for another letter of reference. The one who has the highest value in this second letter wins the competition.

Now, we show the formalization of the procedure by using Crypto-CCS. The entities at stake are represented as Crypto-CCS processes. For the sake of simplicity, we consider the case with a single candidate.

The employer process is parameterized by the threshold value $v_t$, by $\{PE, rf, 1\}_{E^{-1}}$ and by the credential obtained after the interview, *i.e.,* $\{C, f, v_E\}_{E^{-1}}$. This last assumption is a modeling trick that totally bypasses the candidate $C$ in the model we are going to present. We justify this choice for the sake of readability, since we are mainly interested in giving the flavor of a Crypto-CCS formalization and we simplify as much as possible the procedure.

Each conclusion $x_m$ of an inference construct is a message variable and it means: *variable x should contain message m.*

$$
\begin{array}{ll}
E(v_t, \{PE, rf, 1\}_{E^{-1}}, \{C, f, v_E\}_{E^{-1}}) \doteq & \\
c?(x_l). & \mathit{Receive\ the\ reference} \\
[x_l \quad PE \vdash_{ver} x_{(C,f,v_{PE})}] & \mathit{Verify\ the\ signature} \\
[\{PE, rf, 1\}_{E^{-1}} \quad x_l \vdash_4 x_{\{C,f,v_{PE}\}_{E^{-1}}}] & \mathit{Apply\ rule\ 4} \\
[x_{\{C,f,v_{PE}\}_{E^{-1}}} \quad \{C, f, v_E\}_{E^{-1}} \vdash_5 x_{\{C,f,v_{PE} \odot v_E\}_{E^{-1}}}] & \mathit{Apply\ rule\ 5} \\
[x_{\{C,f,v_{PE} \odot v_E\}_{E^{-1}}} \quad E \vdash_{ver} x_{\{C,f,v_{PE} \odot v_E\}}] & \mathit{Verify\ the\ signature} \\
[x_{\{C,f,v_{PE} \odot v_E\}} \vdash_{snd} x_{v_{PE} \odot v_E\}}] & \mathit{Extract}\ x_{v_{PE} \odot v_E} \\
[x_{v_{PE} \odot v_E\}} \quad v_t \vdash_{\geq} y] & \mathit{If}\ v_{PE} \odot v_E \geq v_t\ \mathit{then} \\
c_{out}!yes.\mathbf{0} & \mathit{success};\ \mathit{else\ abort}
\end{array}
$$

The previous employer process is parameterized by the opinion about $C$.

$$
\begin{array}{ll}
PE(\{C, f, v_{PE}\}_{PE^{-1}}) \doteq & \\
c!\{C, f, v_{PE}\}_{PE^{-1}}.\mathbf{0} & \mathit{Send\ the\ letter\ and\ stop}
\end{array}
$$

The whole Crypto-CCS process is defined as

$$P \doteq PE(\{C, f, v_{PE}\}_{PE^{-1}}) || E(v_t, \{PE, rf, 1\}_{E^{-1}}, \{C, f, v_E\}_{E^{-1}}).$$

## 7 Applications to Automated Trust Negotiation

Here we show how our machinery may help on studying security properties on Automated Trust Negotiation. Let the reader consider two users, a *requester*, trying to enter a resource, and an *access mediator*, controlling the resource. There could be a trust establishment phase where the two entities exchange some credentials in several steps. During the exchange, the requester could not know exactly which kind of credential to present, the access mediator could try to help him by prompting the access control policy for its resource, and, also, some user's attributes stated in the credentials used for the negotiation phase could be sensible. Thus, credentials are managed like resources to be protected, and have their own access (disclosure) control policies. This applies to both the requester and the access mediator. This aspect of trust management is an active topic of investigation and it is called Automated Trust Negotiation (ATN, for short), *e.g.,* see [4,17,18,19].

Since ATN actually deals with protocols for exchanging credentials, it seems natural that it should be modeled in our framework.

In [18], a participant in a trust negotiation protocol is described through a finite configuration $G = \langle K_G, E, Policy_G, Ack_G \rangle$, where: $K_G$ is the public key of the participant; $E$ is a set of credentials; $Policy_G$ is a table where to each entry corresponds a positive propositional logic formula expressing a disclosure policy for attributes; $Ack_G$ is a partial function mapping attributes to an entry in $Policy_G$. Basically, a credential proving an attribute may be disclosed only if the attributes presented by the other participants satisfy the corresponding (ack-)policy. The goal is to protect attributes rather than credentials where these attributes are stated (see [18] for a deeper discussion).

A negotiation strategy *strat* describes the behavior of each negotiator. A negotiation starts when the requester sends a request to the access mediator and continues by exchanging of messages. Each participant has a local state that keeps track of the negotiation steps. We have two special states: *failure, success.* The negotiation process fails when one of the two participants enters into the failure state. The negotiation process succeeds when the access mediator enters into the success state.

Using Crypto-CCS, we may model the negotiation steps performed by a negotiator starting in a configuration $G$ and using a strategy *strat* through a term of the process algebra. Along with showing how to model negotiation strategies by process algebras, [12] presented some results in relating a specific notion of non-interference to some privacy issues discussed in [18].

Roughly, a negotiation that preserves privacy is a negotiation in which no adversary, using observations it can make during the negotiation phase with the other participant, may infer something about credentials proving attributes that it is not entitled to know. The definition about privacy issues given in [18] depends on the notion of the adversary strategy *seq* (a certain attack sequence) and on the notion of the behavior of the other participant, given *seq*. A consequence of using our formalism, equipped with a precise operational semantics and an abstract model of cryptography, is that these notions come for free.

Indeed, let $Rel_{G,M}$ be the set of credentials that can be safely disclosed (without revealing non authorized information about attributes to an adversary $M$). Then, an adversary should not be able to tell apart two configurations that are equal but for the set of credentials not in the *Rel* set. We do not consider here the strategy that will be used by $M$, but simply the set of credentials it has at the beginning of the computation (this set is usually called *initial knowledge of the intruder* in security protocol analysis).

According to [1], a protocol $S(x)$ keeps secret the message $x$ iff for any message

$M, M'$ there is no attacker able to tell apart $S(M)$ from $S(M')$. This secrecy property has been nicely modeled by exploiting an equivalence notion, called *May-Testing Equivalence*. The notion states that two processes cannot be distinguished by any process (tester). In our framework, it is possible to formally impose the fact that the tester is not able to break cryptography and so to forge credentials. Let us consider as testers only the ones with any credentials able to infer a fixed set of attributes. Eventually, one configuration $G$ and one configuration $G'$ that have the same $Rel$ set and differ on the credentials that cannot be released, may be analyzed by using a single process $A_{G,Rel}(y)$ that has as parameter $y$ the set of credentials that cannot be disclosed. Thus, this amounts to check whether or not $A_{G,Rel}$ keeps secret such credentials.

## 8 Conclusions

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, access control policies based on credentials. This made it very natural for us to model automated trust negotiation problems, as proposed in [18], as non-interference ones.

As future work we plan at least to provide automated tools for security and trust analysis as well as to define a fully integrated semantics with trust measures for the whole $RT$ family.

## Acknowledgement

The work presented here was partly supported by the EU project IST-3-016004-IP-09 Sensoria (*Software Engineering for Service Oriented Overlay Computers*), the EU network of excellence IST-004527 ARTIST2 on Embedded Systems Design, and by the EU project IST-4-27004-STREP S3MS (*Secure Software and Service for Mobile Systems*).

## References

[1] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 1999.

[2] G. Bella, S. Bistarelli, and S. N. Foley. Soft constraints for security. *Electr. Notes Theor. Comput. Sci.*, 142:11–29, 2006. VODCA 2004.

[3] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society, 1996.

[4] R. Dingledine and P. F. Syverson, editors. *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers*, volume 2482 of *Lecture Notes in Computer Science*. Springer, 2003.

[5] A. Jøsang, L. Gray, and M. Kinateder. Analysing topologies of transitive trust. In *FAST, Tech. Rep.IIT TR-10/2003*, pages 9–22, 2003.

[6] A. Jøsang, L. Gray, and M. Kinateder. Simplification and analysis of transitive trust networks. *Web Intelligence and Agent Systems*, 4(2):139–161, 2006.

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

[8] D. Marchignoli and F. Martinelli. Automatic verification of cryptographic protocols through compositional analysis techniques. In *TACAS*, volume LNCS 1579, pages 148–162. Springer, 1999.

[9] F. Martinelli. Languages for description and analysis of authentication protocols. In P. Degano and U. Vaccaro, editors, *Proceedings of 6th Italian Conference on Theoretical Computer Science*, pages 304–315. World Scientific, 1998.

[10] F. Martinelli. Symbolic semantics and analysis for crypto-ccs with (almost) generic inference systems. In *MFCS, LNCS 2420*, pages 519–531. Springer, 2002.

[11] F. Martinelli. Analysis of security protocols as open systems. *Theoretical Computer Science*, 290(1):1057–1106, 2003.

[12] F. Martinelli. Towards an integrated formal analysis for security and trust. In *FMOODS*, volume LNCS 3535, pages 115–130. Springer, 2005.

[13] F. Martinelli and M. Petrocchi. On relating and integrating two trust management frameworks. In *Views On Designing Complex Architectures (VODCA06). To appear in ENTCS.*

[14] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[15] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 2(29):38–47, 1996.

[16] G. Theodorakopoulos and J. S. Baras. Trust evaluation in ad-hoc networks. In *WiSe*, pages 1–10. ACM Press, 2004.

[17] W. H. Winsborough and N. Li. Towards practical automated trust negotiation. In *POLICY*, pages 92–103. IEEE, 2002.

[18] W. H. Winsborough and N. Li. Safety in automated trust negotiation. In *S&P*. IEEE, 2004.

[19] M. Winslett. An introduction to automated trust establishment. In *GI Jahrestagung*, pages 106–113, 2002.