Towards an integrated formal analysis for security and trust*

Fabio Martinelli

Istituto di Informatica e Telematica - C.N.R., Pisa, Italy.
e-mail: Fabio.Martinelli@iit.cnr.it

Abstract. We aim at defining an integrated framework for the specification and (automated) analysis for security and trust in complex and dynamic scenarios. In particular, we show how the same machinery used for the formal verification of security protocols may be used to analyze access control policies based on trust management.

1 Introduction

Computer security is a research area that is increasingly receiving the attention of researchers. In particular, consider some security issues in ubiquitous computing systems: these consist of different entities that have to cooperate and share resources to achieve a certain goal. Cooperation is often enabled by trust relationships between entities. There is a tight connection between the security mechanisms used to guarantee the confidentiality and integrity of information and mechanisms used to establish, manage and negotiate trust, reputation and recommendation among the different entities.

In this paper we focus on the integrated formal modeling and analysis of security and trust. In particular, we uniformly model security protocols and some form of access control based on trust management.

Formal languages for modeling distributed systems have been applied in the last decade to the analysis of cryptographic protocols. In this framework, cryptography is usually modeled by representing encryptions as terms of an algebra, e.g., $E(m, k)$ may represent the encryption of a message m with a key k. Usually, the so-called perfect encryption abstraction is adopted: encryptions are considered as injective functions which can be inverted only by knowing the correct information, i.e. the decryption key. For instance, common inference rules for modeling the behavior of the encryption and decryption (in a shared-key schema) are the followings:

$$\frac{m \quad k}{E(m,k)} \qquad \frac{E(m,k) \quad k}{m} \tag{1}$$

* Work partially supported by CNR project "Trusted e-services for dynamic coalitions" and by a CREATE-NET grant for the project "Quality of Protection (QoP)". A preliminary version appeared as [18].

which should be read as: from a message m and a key k we can build the encryption $E(m,k)$; from an encryption $E(m,k)$ and a decryption key k we can obtain the encrypted message m.

The long standing tradition of modeling the specific features of cryptographic functions as term-rewriting rules met the powerful verification techniques developed for process algebras. As a matter of fact, several formal languages for describing communication protocols, for instance CSP [17], have been exploited for representing cryptographic protocols without changes in syntax or semantics: the inference rules have been given at the meta-level of the verification. Instead others, like the π–calculus [1] and the CCS [19, 21], have been effectively refined: the π–calculus have been equipped with two pattern matching constructs for modeling message splitting and shared-key decryption, respectively; the CCS has been equipped with an inference construct that permits to infer new messages from others, i.e.:

$$[m_1 \quad m_n \vdash_r x].P$$

which denotes a process that tries to deduce a message m from the messages in $m_1, \ldots, m_n$ and when it succeeds it substitutes this message for x in the process specification P. The language is called CryptoCCS ([19]).

The inference relation could be defined in many ways. Often, we will consider the transitive closure of the entailment relations used in each process. This would give a complex inference system. Such inference systems allow us to cope with the variety of different crypto-systems that can be found in the literature.

However, 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 access control policies may change and typically may also depend on credentials presented by users.

Similarly, when one wishes to formally analyze (e.g., see [2]) access control systems, the authentication mechanisms (usually a security protocol) are considered a priori "secure", without further specification.

While separation of concerns is often desirable, this is not always possible. The interplay between security protocols and access control mechanisms/policies is crucial. Moreover, a good specification and analysis framework should take an holistic point of view.

As a matter of fact, we show that the idea proposed by CryptoCCS of using inference constructs is also useful to model access control mechanisms based on credentials in distributed systems.

Example 1. Indeed, consider a set of credentials, i.e. (signed) messages containing information about access rights. Assume that $\{A, ob_1, +\}_{pr(C)}$ means that the user C (via the signature with its private key $pr(C)$) asserts A has the right to access the object ob_1 and may grant this access to other users (this is denoted through the symbol $+$). A rule like:

$$\frac{\{A, ob_1, +\}_{pr(C)} \quad pr(C) \quad \{grant \quad B, ob_1\}_{pr(A)}}{\{B, ob_1, +\}_{pr(C)}}(acc_C)$$

may be used by the controller C to issue other access right credentials, after receiving an indication by A, i.e. the signed message $\{grant \quad B, ob_1\}_{pr(A)}$.

Thus, we may consider the inference rules as an abstract mechanism to express security policies usually defined using other mathematical models and logics (e.g., see [10, 24]).

In this paper, we deal with the RT trust management system [16]. However, our approach is very general. In particular, we will show also how to encode with inference systems the mechanisms for reasoning about trust proposed in [12] and modeled with different approaches.

Having a unique language will allow us to model the interplay between security protocols that use the trust relationships among different users, and the ways in which these relationships are created (that often rely on security/interaction protocols).

The fact that we can both model cryptography and some form of credential/trust management with the inference construct of CryptoCCS allows us to use the software tools and methodologies already developed for security protocols analysis to the more general case where credentials are explicitly managed. In particular, in [22] a software tool for automated security protocols analysis has been defined in [20] has been extended to cope with a huge class of inference systems.

It is worthy noticing that the CryptoCCS has been previously defined to set up a uniform framework for the analysis of security properties and information flow (non-interference) with the same machinery (e.g., see [7, 5, 6]). This helped us quite a lot in establishing a precise correspondence of properties of trust negotiation protocols with non-interference ones (as hinted in [28]).

To sum up, the main contribution of this paper is to present an effective framework, based on the flexibility of the CryptoCCS inference construct, for uniformly specifying and analyzing several aspects of network/system security and trust management.

There are few attempts to analyze security protocols and trust management altogether. A notable example is the recent work in [9]. There the trust is expressed at a meta-level by decorating protocol specifications with formulas of a trust logic and by ensuring that such formulas hold at certain points. Our approach is thus different and is based on modeling trust (in different flavors) inside the protocol specifications.

The paper is organized as follows. Section 2 presents the CryptoCCS language and recalls some of its analysis techniques. Section 3 shows how the CryptoCCS may be naturally used to model trust management languages. Section 4 investigates the relationships between notions of safety in Automated Trust Negotiation and non-interference. Section 5 concludes the paper.

2 CryptoCCS

CryptoCCS [21, 19] is a slight modification of CCS process algebra [23], adopted for the description of cryptographic protocols.

The CryptoCCS model consists of a set of sequential agents able to communicate by exchanging messages.

The data handling part of the language consists of a set of inference rules used to deduce messages from other messages. We consider a set of relations among messages as: $\vdash_r \subseteq \mathcal{M}^{i_r+1}$, where r is the name of the rule and i_r the number of premises. For the sake of simplicity, we assume that $\vdash_r$ (for each $r \in \mathcal{R}$) is decidable.

2.1 The Language Syntax

CryptoCCS syntax is based on the following elements:

- A set Ch of channels, partitioned into a set I of input channels (ranged over by c) and a set O of output channels (ranged over by $\overline{c}$, the output corresponding to the input c);
- A set Var of variables, ranged over by x;
- A set $\mathcal{M}$ of messages, defined over a certain signature, ranged over by M, N, m, n ....

The set $\mathcal{L}$ of CryptoCCS terms (or processes) is defined as follows:

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

$$A(M_1, \ldots, M_n) \mid [\langle M_1, \ldots, M_r \rangle \vdash_{rule} x]P; Q$$

where $M, M', M_1, \ldots, M_r$ are messages or variables 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 [23]. We call $\mathcal{P}$ the set of all the *CryptoCCS* closed and guarded terms. The set of actions is $Act = \{c(M) \mid c \in I\} \cup \{\overline{c}M \mid \overline{c} \in O\} \cup \{\tau\}$ (τ is the internal, invisible action), ranged over by a. We define $sort(P)$ to be the set of all the channels syntactically occurring in the term P. Moreover, for the sake of readability, we always omit the termination $\mathbf{0}$ at the end of process specifications, e.g. we write a in place of $a.\mathbf{0}$. We give an informal overview of *CryptoCCS* operators:

- $\mathbf{0}$ is a process that does nothing.
- $c(x).P$ represents the process that can get an input 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 τ and then behaves like P.
- $P_1 \,|\, P_2$ (*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;
- $A(M_1, \ldots, M_n)$ behaves like the respective defining term P where all the variables $x_1, \ldots, x_n$ are replaced by the messages $M_1, \ldots, M_n$;

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

$$\frac{m \quad k}{\{m\}_k}(\vdash_{enc}) \qquad \frac{\{m\}_k \quad k}{m}(\vdash_{dec})$$

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

– $[\langle M_1, \ldots, M_r\rangle \vdash_{rule} x]P; Q$ 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).

2.2 The Operational Semantics of CryptoCCS

In order to model message handling (and so cryptography in an abstract way) we use a set of inference rules. Note that *CryptoCCS* syntax, its semantics and the results obtained are completely parametric with respect to the inference system used. We present in Figure 1 an instance inference system, with rules: to combine two messages obtaining a pair (rule $\vdash_{pair}$); to extract one message from a pair (rules $\vdash_{fst}$ and $\vdash_{snd}$); to encrypt a message m with a key k obtaining $\{m\}_k$ and, finally, to decrypt a message of the form $\{m\}_k$ only if it has the same key k (rules $\vdash_{enc}$ and $\vdash_{dec}$, 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. Natural numbers may be encoded by assuming a single value 0 and a function $S(y)$, with the following rule: $\frac{x}{S(x)}$ *inc*. Similarly, we can define summations and other operations on natural numbers. ■

Example 3. We do not explicitly define equality check among messages in the syntax. However, this can be implemented through the usage of the inference construct. E.g., consider rule $\frac{x \quad x}{Equal(x,x)}$ *equal*. Then $[m = m']A$ (with the expected semantics) may be equivalently expressed as $[m \quad m' \vdash_{equal} y]A$ where y does not occur in A. Similarly, we can define inequalities, e.g., $\leq$, among natural numbers. ■

The operational semantics of a *CryptoCCS* term is described by means of labeled transition relations, $P \xrightarrow{a} P'$, with the informal meaning that the process P may perform an action a evolving in the process P'. More formally, we consider a *labelled transition system* (*lts*, for short) $\langle \mathcal{P}, Act, \{\xrightarrow{a}\}_{a\in Act}\rangle$, where

$$(input)\frac{m \in \mathcal{M}}{c(x).P \xrightarrow{c(m)} P[m/x]} \qquad (output)\frac{}{\overline{c}m.P \xrightarrow{\overline{c}m} P} \qquad (internal)\frac{}{\tau.P \xrightarrow{\tau} P}$$

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

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

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

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

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

$\{\xrightarrow{a}\}_{a \in Act}$ is the least relation between *CryptoCCS* processes induced by the axioms and inference rules of Figure 2. The expression $P \stackrel{a}{\Rightarrow} P'$ is a shorthand for $P(\xrightarrow{\tau})^* P_1 \xrightarrow{a} P_2(\xrightarrow{\tau})^* P'$ where $(\xrightarrow{\tau})^*$ denotes a (possibly empty) sequence of transitions labeled τ. The expression $P \Rightarrow P'$ is a shorthand for $P(\xrightarrow{\tau})^* P'$.

2.3 Security protocol analysis

The security protocol analysis proposed in [19, 21] is based on the checking of following property:

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

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 a inference systems, i.e. the possibility to iteratively apply the inference rules. Given a set $\mathcal{R}$ of inference rules, we consider the deduction relation $\mathcal{D}^{\mathcal{R}} \subseteq \mathcal{P}^{fin}(\mathcal{M}) \times \mathcal{M}$. Given a finite set of closed messages, say ϕ, 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. Below, we present an example useful in our case (e.g., see also [20]).

2.4 Some assumptions on the inference system

Given a well-founded measure on messages, we say that a rule

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

is a S-rule (*shrinking* rule), whenever the conclusion is a proper subterm of one of the premises (call such premise *main*). The rule r is a G-rule (*growing* rule) whenever the conclusion is strictly larger than each of the premises, and all the variables in the conclusion must be in the premises.

Definition 1. *We say that an inference system enjoys a G/S property if it consists only of G-rules and S-rules, moreover whenever a message can be deduced through a S-rule, where one of the main premises is derived by means of a G-rule, then the same message may be deduced from the premises of the G-rule, by using only G-rules.*

Several of the inference systems used in the literature for describing cryptographic systems enjoy this restriction[1].

Indeed, using G-rules for inferring the main premises of an S-rules, is unuseful. Thus, shrinking rules may be significantly applied only to messages in ϕ and to messages obtained by S-rules. However, since the measure for classifying the S-rules is well-founded then such a shrinking phase would eventually terminate when applied to a closed set of messages ϕ. Then, only growing rules are possible. Thus, if the inference system enjoys the G/S restriction then $\mathcal{D}^R(\phi)$ is decidable when ϕ is finite. We may note that the inference system in page 1 enjoys the G/S restriction and so its deduction relation is indeed decidable.

In the case the inference system has no growing rules, we have decidability even in the presence of a weaker form of *shrinking rules*. We say that a rule is *eq-shrink* whenever the conclusion has an equal or smaller size than one of the premises; moreover all the variables occurring in the conclusion must occur in at least one of the premises. In such a case the decision procedure simply consists of building the transitive closure of the inference rules.

3 Modeling several trust management languages

Through process algebras, one can formally specify communicating protocols and complex distributed systems. For instance, one could use CryptoCCS to describe the components and the communication interface of an access control mechanism as the Policy Enforcement Point (PEP), the Policy Decision Point (PDP) and the resource to be protected (see [26]).

In Figure 3 we may see the components of a common access control framework. A request is performed by the user to the PEP. The PEP often applies a communication with the user, often performing an authentication protocol. Then, using the information acquired by PEP is sent to the PDP. Eventually the access is granted to the resource.

[1] It is worthy noticing that in [13] a similar terminology has been used, and a restriction, called S/G, has been defined. However, this is rather different from ours and it is not well suited to model cryptographic systems.

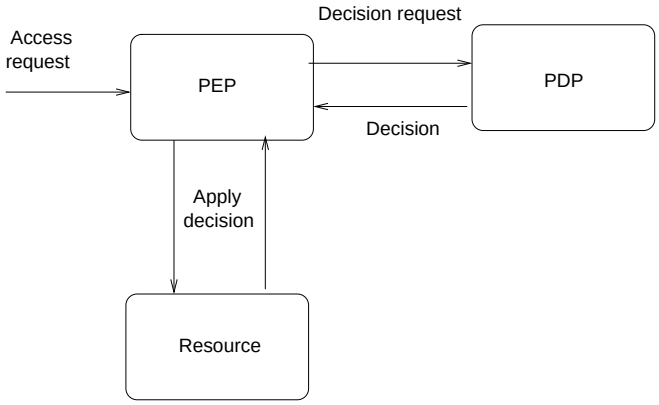

Fig. 3. An access control system

In particular, in trust management systems, where policies are given through credentials, this allow one to use the inference system of CryptoCCS to model also the trust engine used in these frameworks. Let us see how it works with two well known models.

3.1 RT_0: Role-based Trust Management

We show how inference rules can be conveniently used to model RT languages for trust management [16, 28, 15, 14]. In these languages, credentials carry information on policies to define attributes of principals by starting from assertions of other principals. The notion of attribute is general enough to permit to use RT languages to model Role-based Access Control Mechanisms (RBAC), e.g. see [25]. As a matter of fact, an attribute could be considered as a role. Then one could use RT credential to express how principals are related to roles[2]. More precisely, we denote principals with $A, B, C...$; we denote role names with $r, u, z....$ A role takes the form of a principal followed by a role name, separated by a dot, e.g. $A.r$.

RT assumes four kind of credentials that express possible policy statements.

- $A.r \leftarrow D$ (**simple member**)
  This statement defines that D has role $A.r$.
- $A.r \leftarrow A_1.r_1$ (**simple containment**)
  This statement asserts that if D has role $A_1.r_1$ then it has role $A.r$. This kind of credential can be used to delegate the authentication of attributes from A to A_1.
- $A.r \leftarrow A_1.r_1.r_2$ (**linking containment**)
  This statement asserts that E has role $A_1.r_1$ and D has role $E.r_2$ then D

[2] Similarly, credentials and attributes could be used to assign permissions to roles

has role $A.r$. This kind of credential may be used to delegate the assignment of $A.r$ role not to specific entities but to entities of a given role.

– $A.r \leftarrow A_1.r_1 \cap A_2.r_2$ (**Intersection containment**)
This statement asserts that D has role $A_1.r_1$ and $A_2.r_2$ then D has role $A.r$.

Example 4. Consider the following set of credentials.

$$Univ.stud \leftarrow FM$$
$$Shop.discount \leftarrow Univ.stud$$

It follows that FM has role $Shop.discount$. So, the shop offers discounts to the students of the University.

The language for credentials has been equipped with several semantics. In particular, one semantics based on datalog is very similar to our inference rules (that in this case can be seen as datalog rules). So, we define one inference rule for each credential as follows.

$$A.r \leftarrow D \qquad \{D, r\}_A$$
$$A.r \leftarrow A_1.r_1 \qquad \frac{\{y, r_1\}_A}{\{y, r\}_A}$$
$$A.r \leftarrow A_1.r_1.r_2 \qquad \frac{\{z, r_1\}_{A_1} \quad \{y, r_2\}_z}{\{y, r\}_A}$$
$$A.r \leftarrow A_1.r_1 \cap A_2.r_2 \qquad \frac{\{y, r_1\}_{A_1} \quad \{y, r_2\}_{A_2}}{\{y, r\}_A}$$

However, this requires a rule for each credential. We wish to fix from the very beginning the set of inference rules. Thus, we provide a slightly modified version of the inference system where we consider only 3 rules, one for each kind of credential defined in RT_0 (with the exception of the first kind of credentials that are simply messages).

$$A.r \leftarrow D \qquad \{D, r\}_A$$
$$A.r \leftarrow A_1.r_1 \qquad \frac{\{y, r_1\}_A \quad \{r, A_1, r_1\}_A}{\{y, r\}_A}$$
$$A.r \leftarrow A_1.r_1.r_2 \qquad \frac{\{z, r_1\}_{A_1} \quad \{y, r_2\}_z \quad \{r, A_1, r_1, r_2\}_A}{\{y, r\}_A}$$
$$A.r \leftarrow A_1.r_1 \cap A_2.r_2 \qquad \frac{\{y, r_1\}_{A_1} \quad \{y, r_2\}_{A_2} \quad \{r, A_1, A_2, r_1, r_2\}_A}{\{y, r\}_A}$$

Note that, under the common measure of the size of terms, all the previous rules are *eq-shrink* rules and there are no *growing* rules. Thus, establishing whether a given principal, say D, has a certain role in a policy ϕ, i.e. $\{D, r\}_A \in \mathcal{D}(\phi)$ is decidable. This kind of analysis[3] is called *Simple Safety* in [14] and can be performed by our analysis tool PaMoChSA [22].

[3] Actually, that work considers a dynamic set of policies. However, the analysis technique adopted is actually based on a subset of the set of prolog rules that represent the initial problem. Thus, we are also able to manage it.

3.2 Josang *et al.* topologies

We also show how the trust model of Josang *et al.* [12] can be managed in our framework. The authors suggest trust is always linked to a purpose. The most natural situation is when one trusts another for performing a certain function/task. This may be expressed as $A \xrightarrow{f} D$, i.e. A trusts D for performing f. Moreover, it is often common that one, say A, asks another, say D, for suggesting/reccomendating a third one for doing a given task, i.e. f. This could be expressed by the following credential $A \xrightarrow{r,f} D$.

The main idea is that 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 all the others are recommendation steps. Thus, we have another kind of credential like $A \xrightarrow{r} B \xrightarrow{f} D$, , expressing the fact that A trusts D for performing f via the recommendation of B.

$$
\begin{array}{ll}
A \xrightarrow{f} D & \{f, D\}_A \\
A \xrightarrow{r,f} D & \{r, D, f\}_A \\
\dfrac{A \xrightarrow{r,f} B \quad B \xrightarrow{r,f} D}{A \xrightarrow{r,f} D} & \dfrac{\{r, B, f\}_A \quad \{r, D, f\}_B}{\{r, D, f\}_A} \\
\dfrac{A \xrightarrow{r,f} B \quad B \xrightarrow{f} D}{A \xrightarrow{r} B \xrightarrow{f} D} & \dfrac{\{r, B, f\}_A \quad \{f, D\}_B}{\{r, B, f, D\}_A}
\end{array}
$$

As in the previous case, the deduction relation of this set of rules is decidable. This gives us an alternative strategy w.r.t. the one presented in [12].

As in [12], one could insert further information into the credentials, as measure of trust. For instance, credentials could be enhanced with such information and rules could derived the trust measure of resulting credentials in the appropriate way. For instance, consider the following credential enhanced with a trust measure, i.e.: $A \xrightarrow{r,f,m} B$. Then the transitive composition rule could be the following:

$$
\frac{A \xrightarrow{r,f,m_1} B \quad B \xrightarrow{r,f,m_2} D}{A \xrightarrow{r,f,m_3} D}
$$

where m_3 is a function of m_1, m_2, for instance $m_3 = min\{m_1, m_2\}$.

If the set of possible trust values is finite, then the deduction relation is still decidable. More complex trust measures can be found in [11]. Clearly, one may try to define specific strategies for each set of inference rules in order to obtain decidability. However, we argue that the mechanisms we proposed are general enough to deal with common trust management systems.

4 An application to Automated Trust Negotiation problems

The usage of credentials for policy decision is useful, but as mentioned before is not the unique part of access control. When one user (i.e., a *requester*) tries to access to a resource controlled by another entity (i.e., *access mediator*) there could

be a trust establishment phase where the two entities exchange some credentials in several steps. As a matter of fact, the requester could not know exactly which kind of credential to present. Then, the access mediator could try to help him by prompting the access control policy for its resource. Some user's attributes stated in the credentials used for the negotiation phase could be sensible. Thus, specific procedures for controlling the disclosure of such credentials have been designed. 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 is called in the literature Automated trust negotiation (ATN, for short), e.g. see [3, 27, 29, 28].

Since ATN actually deals with protocols for exchanging credentials, it seems natural that it should be modeled in our framework. This has a very nice consequence to make it formal the intuition expressed in [28] that some properties of ATN resemble non-interference ones. We exactly identify a notion that is very good for describing properties of ATN.

We briefly present a slightly simplified version of the theory developed in [28].

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 (i.e., the participant knows the corresponding private key);
- E is a set of credentials, where we assume that the subject of each credential is K_G;
- $Policy_G$ is a table where to each entry corresponds a positive propositional logic formula expressing a disclosure policy for attributes (such a logic may be easily modeled through a suitable inference system);
- 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 [28] for a deeper discussion).

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.

A negotiation strategy *strat* describes the behavior of each negotiator (in contrast to [28] we do not assume it is deterministic).

- $strat.rstart(G, K_O)$ is used by the requester just after the sending of the access request to the access mediator; it returns the requester's initial local state;

– $strat.start(G, pid, K_O)$ is used to respond to the first message from the requester; the access mediator checks the policy associated with the resource (identified by pid) and then determines the next local state and message to be sent to the requester. (The negotiation proceeds only if the state is different from $success$ and $failure$.)
– $strat.respond(G, st, msg)$ is used to respond to a message from the other negotiator; it returns the new local state and the next message to be sent. (The negotiation proceeds only if the state is different from $success$ and $failure$.)

Using CryptoCCS, 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. Note that states simply record the execution history of a negotiation (and the initial request). Thus, by recording the messages received and sent, one may avoid the usage of states. Moreover, note that having in the term algebra a constructor for pairs one may express sequences of messages using a single one. We assume to have two special messages used to encode the success and failure states, and an inference system $\vdash_{G,strat}$ that suitably mimics $strat$ strategy. Eventually, the definition of the term corresponding, for instance, to the requester is as follows:

$$
\begin{array}{rll}
A_{G,rstart} = & \overline{c}(resource). & \text{outpus the resource requests} \\
& A_{G,respond}(resource, nil) & \text{proceeds to the respond phase}
\end{array}
$$

$$
\begin{array}{rll}
A_{G,respond}(s, r) = & c(y). & \text{receives the message} \\
& [r \quad y \vdash_{pair} r_1] & \text{added to received ones} \\
& [s \quad z \vdash_{G,strat} x] & \text{next message} \\
& [x = success]A_{G,success}; (& \text{success} \\
& [x = failure]A_{G,failure}; (& \text{failure} \\
& \overline{c}(x). & \text{outputs next message} \\
& [s \quad x \vdash_{pair} s_1] & \text{added to sent ones} \\
& A_{G,respond}(s_1, r_1) & \text{continues the negotiation} \\
&)) &
\end{array}
$$

In [28] there is an interesting discussion about the privacy issues in ATN. This is out of the scope of this paper. We are simply interested in presenting the main notion of privacy preserving negotiation defined in that paper, i.e., credential-combination-hiding[4].

Roughly, no adversary, using observations it can make during the negotiation phase with the other participant, may infer something about credentials proving attributes it is not entitled to know (i.e., it does not satisfy the appropriate disclosure policies).

We give below a variant of the notion of indistinguishability for non-deterministic strategies originally presented in [28].

[4] Note also we are not advocating this property; we simply show how it is possible to relate it to a specific notion of non-interference.

Definition 2. *(Indistinguishability) Given an adversary M and a negotiation strategy strat, two configurations $G = \langle K, E, Policy, Ack\rangle$, $G' = \langle K, E', Policy, Ack\rangle$ are indistinguishable under strat by M, if and only if for every attack sequence seq, any possible response sequence induced by seq from G is among the ones induced by seq on G'.*

Clearly, this definition depends on the notion of attack sequence and response. 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. (For instance, in [28] a notion of computationally feasible is referred while dealing with cryptography forgery, without mentioning the difficulties on managing it in an automated manner.)

Then, one identifies the set of credentials that can be safely disclosed without revealing information about attributes the adversary M is not entitled to know (i.e., it cannot present the necessary credentials during the negotiation phase). Call this set $Rel_{G,M}$. 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. Note that here we do not consider the strategy that will be used by M but simply the set of credentials it has at the beginning of the computation (called usually initial knowledge in security protocol analysis, e.g. see [7]).

Definition 3. *(Credential-combination hiding safe) A negotiation strategy strat is* credential-combination hiding safe *if for every pair of configurations $G = \langle K, E, Policy, Ack\rangle$ and $G' = \langle K, E', Policy, Ack\rangle$, and adversary M with $Rel_{G,M} = Rel_{G',M}$ then G and G' are indistinguishable.*

4.1 ATN properties as non-interference properties

To solve the problem of preventing unauthorized information flows, be they direct or indirect, in the last two decades many proposals have been presented, starting from the seminal idea of *non interference* proposed in [8] for deterministic functions. In [4–7], many non interference-like notions in the literature have been uniformly defined in a common process algebraic setting based on CryptoCCS, producing one of the first taxonomies of these properties reported in the literature.

We recall here a notion of secrecy about security protocols defined by Abadi and Gordon [1].

Basically, a protocol $S(x)$ keeps secret the variable 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 testing equivalence.

May-Testing Equivalence 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.

Consider a special action ω available only to testers. We say that two processes P and Q are may testing equivalent iff for any tester T, $P \,|\, T \stackrel{\omega}{\Longrightarrow} P'$ iff

$Q \mid T \stackrel{\omega}{\Longrightarrow} Q'$. Basically, the tester plays the role of the adversary. Thus the notion of indistinguishability is similar to the notion of testing equivalence, when one considers 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 with 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 (again we assume that a set of messages may be encoded as a single one).

Probabilistic notions of this property may be easily given by using suitable modifications of the process algebra and of the corresponding may testing equivalence.

5 Conclusions and future work

We have shown how the same machinery used for the formal specification and verification of security protocols may be used to analyze a variety of access control approaches based on trust management. In addition, in [5, 7, 21] CryptoCCS has been proposed as a uniform specification and verification framework for security protocols properties and non-interference ones, usually managed with different techniques. This made it very natural for us to model automated trust negotiation problems as proposed in [28] as non-interference ones.

The approach presented in this paper may be considered as a step towards the creation of a uniform and automated verification framework for studying security properties of networked systems. As future work we wish to extend our analysis tool called PaMoChSA [22] to fully support our approach. Moreover, we wish to investigate more deeply the relationships of non-interference with ATN properties.

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

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. P. A. Bonatti and P. Samarati. Logics for authorizations and security. In *Logics for Emerging Applications of Databases*, LNCS. Springer-Verlag, 2003.
3. 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.
4. R. Focardi and R. Gorrieri. Classification of security properties (part i: Information flow). In *Foundations of Security Analysis and Design*, volume 2171 of *Lecture Notes in Computer Science*, pages 331–396, 2001.

5. R. Focardi, R. Gorrieri, and F. Martinelli. Non interference for the analysis of cryptographic protocols. In *Proceedings of 27th International Colloquium in Automata, Languages and Programming*, volume 1853 of *Lectures Notes in Computer Science*, pages 354–372, 2000.

6. R. Focardi, R. Gorrieri, and F. Martinelli. Classification of security properties – Part II: Network security. In *FOSAD school Lectures*, volume 2946 of *LNCS*. Springer-Verlag, 2004.

7. R. Focardi and F. Martinelli. A uniform approach for the definition of security properties. In *Proceedings of World Congress on Formal Methods (FM'99)*, volume 1708 of *Lecture Notes in Computer Science*, pages 794–813, 1999.

8. J. A. Goguen and J. Meseguer. Security policy and security models. In *Proceedings of the 1982 Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, 1982.

9. J. D. Guttman, F. J. Thayer, J. A. Carlson, J. C. Herzog, J. D. Ramsdell, and B. T. Sniffen. Trust management in strand spaces: A rely-gurantee method. In *Proceedings of the European Symposium on Programming (ESOP)*, LNCS, 2004.

10. Halpern and van der Meyden. A logic for SDSI's linked local name spaces. In *PCSFW: Proceedings of The 12th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1999.

11. A. Josang. The consensus operator for combining beliefs. *Artif. Intell.*, 141(1):157–170, 2002.

12. A. Josang, E. Gray, and M. Kinateder. Analysing topologies of transitive trust. In *Proc. of the* 1^{st} *workshop on Formal Aspects in Security and Trust (FAST2003)*, 2003.

13. D. Kindred and J. M. Wing. Fast, automatic checking of security protocols. In *Second USENIX Workshop on Electronic Commerce*, pages 41–52, Oakland, California, 1996.

14. N. Li, J. Mitchell, and W. H. Winsborough. Beyond proof-of-compliance: Safety and availability analysis in trust management. In *IEEE Symposium on Research in Security and Privacy*. 2003.

15. N. Li and M. V. Tripunitara. Security analysis in role-based access control. In *ACM Symposium on Access Control Models and Techniques (SACMAT 2004)*. 2004.

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

17. G. Lowe. Breaking and fixing the Needham Schroeder public-key protocol using FDR. In *Proceedings of Tools and Algorithms for the Construction and the Analisys of Systems*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer Verlag, 1996.

18. F. Martinelli. Towards a uniform framework for the formal analysis for security and trust. Technical Report IIT, 2004. A position paper has been presented at 2nd Workshop on Ubiquitious Networking (Cambridge, UK-Ubinet 2004).

19. 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.

20. F. Martinelli. Symbolic semantics and analysis for crypto-ccs with (almost) generic inference systems. In *Proceedings of the 27th international Symposium in Mathematical Foundations of Computer Sciences(MFCS'02)*, volume 2420 of *LNCS*, pages 519–531, 2002.

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

22. F. Martinelli, M. Petrocchi, and A. Vaccarelli. PaMoChSA: A tool for verification of security protocols based on partial model checking. 2001. Tool Demo at the 1st International School on Formal Methods for the Design of Computer, Communication and Software Systems: Process Algebras.

23. R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.

24. P. Samarati and S. D. C. di Vimercati. Access control: Policies, models, and mechanisms. In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design*, LNCS 2171. Springer-Verlag, 2001.

25. R. Sandhu, V. Bhamidipati, E. Coyne, S. Ganta, and C. Youman. The arbac97 model for role-based administration of roles: preliminary description and outline. In *Proceedings of the second ACM workshop on Role-based access control*, pages 41–50. ACM Press, 1997.

26. J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, B. C. de Laat, M. Holdrege, and D. Spence. RFC 2904 AAA authorization framework. 2000.

27. W. H. Winsborough and N. Li. Towards practical automated trust negotiation. In *IEEE 3rd Intl. Workshop on Policies for Distributed Systems and Networks (Policy)*, 2002.

28. W. H. Winsborough and N. Li. Safety in automated trust negotiation. In *IEEE Symposium on Security and Privacy*. 2004.

29. M. Winslett. An introduction to automated trust negotiation. In *Workshop on Credential-Based Access Control Dortmund, October 2002*.