Paper

Two Semantics of Trust Management Language with Negation

Anna Felkner

Research and Academic Computer Network (NASK), Warsaw, Poland

Abstract—The family of Role-based Trust management languages is used for representing security policies by defining a formalism, which uses credentials to handle trust in decentralized, distributed access control systems. A credential provides information about the privileges of users and the security policies issued by one or more trusted authorities. The main topic of this paper is $RT_{\ominus}$, a language which provides a carefully controlled form of non-monotonicity. The core part of the paper defines two different semantics of $RT_{\ominus}$ language – a relational, set-theoretic semantics for the language, and an inference system, which is a kind of operational semantics. The set-theoretic semantics maps roles to a set of entity names. In the operational semantics credentials can be derived from an initial set of credentials using a set of inference rules. The soundness and the completeness of the inference system with respect to the set-theoretic semantics of $RT_{\ominus}$ will be proven.

Keywords—access control, inference system, monotonicity, Role-based Trust management, set-theoretic semantics.

1. Introduction

Confidential data, whether in electronic, paper or other form must be properly protected. Guaranteeing that the data and services offered by a computer system are not made available to unauthorized users is an increasingly significant and challenging issue, which must be solved by reliable software technologies. This problem is usually solved by implementing access control techniques. In a typical access control scenario there are two entities, one is the requester that wants to access a protected resource, while the other is an entity that is the resource owner or provider. Usually these are the only entities involved in making the authorization decision. This approach fits well into closed, centralized environments, in which the identity of users is known in advance. Unfortunately, this simple scenario does not apply to highly distributed and decentralized networks. Quite different challenges arise in such decentralized and open systems, where the identity of users is not known in advance and the set of users can change. In decentralized environments, the resource owner and the requester often are unknown to one another, making access control based on identity ineffective. To be able to deal with different requesters coming from different security domains, we need a more flexible solution, named *trust management*.

Trust management is an approach to access control in decentralized distributed systems, where access control decisions are based on policy statements made by multiple principals. The potential and flexibility of trust management approach stems from the possibility of *delegation*: a principal may transfer limited authority over a resource to other principals. Such a delegation is implemented by means of an appropriate credential. This way, a set of credentials defines the access control strategy and allows deciding on who is authorized to access a resource, and who is not.

Despite the fact that the credentials-based models do, to a large degree, solve the access control problem in open systems, they still have some shortcomings. Most trust management languages are *monotonic*: adding new assertion to a query can never result in canceling an action, which was accepted before [1]. It is a problem, because each policy statement or credential added to the system can only increase the capabilities and privileges granted to others. The monotonicity property can simplify the design and analysis of complex network-based security protocols. It is a good property for researching, analyzing and proving, but causes limited usability, because revocation of privileges is not possible to assert. However, we can find in a literature various extensions of basic languages and models that create negation on different levels. And thus, in this way we achieve a non-monotonicity. One of the extensions is Role-based Trust management language with negation.

The rest of the paper is organized as follows. An overview of the work related to trust management systems and languages is given in Section 2. Section 3 shows the overview of the family of Role-based Trust management languages, Section 4 describes set-theoretic semantics of *RT* languages, including an example, and inference system over *RT* credentials with example is shown in Section 5. Final remarks are given in Conclusions.

2. Related Work

Traditional access control systems often rely on Role-Based Access Control (RBAC) model [2], which groups the access rights by the role name and limits the access to a resource to those users, who are assigned to a particular role. It is the most flexible type of access control policy.

The first trust management application described in the literature was PolicyMaker [3], which defined a special assertion language capable of expressing policy statements, which were locally trusted, and credentials, which

had to be signed using a private key. The next generation of trust management languages were KeyNote [4], which was an enhanced version of PolicyMaker, SPKI/SDSI [5] and a few other languages [6]. All those languages allowed assigning privileges to entities and used credentials to delegate permissions from its issuer to its subject. What was missing in those languages was the possibility of delegation based on attributes of the entities and not on their identity. Responding to this need, a family of Role-based Trust management languages has been introduced in [7]–[9]. These languages have a well-defined syntax and semantics, which made them easy to extend in order to apply them to different needs.

A set-theoretic semantics, which defines the meaning of a set of credentials as a function from the set of roles into the power set of entities, has been defined for RT_0 [10], [9] and relational semantics, which apply to RT^T in [11].

One of the extensions of RT languages is the use of time validity constraints of the credentials, which made the languages of the RT family more realistic, because in the real world permissions are usually given just for a limited period of time. Time-dependant credentials were introduced in [10] (for RT_0) and in [12] (for RT^T). This type of time constraints can eliminate the need of non-monotonic system in some cases. Another approach to the monotonicity feature is an extension of RT_0 language, which was created to manage trust in P2P applications and access control in virtual communities described in [1].

3. Role-based Trust Management Languages

The term of *trust management* was introduced in year 1996 by Blaze *et al*. in [13], who defined it as a unified approach to specify and interpret security policies, credentials and trust relationships. In a trust management system an entity's privilege is based on its attributes instead of its identities. An entity's attributes are demonstrated through digitally signed credentials issued by multiple principals. A *credential* is an attestation of qualification, competence or authority issued to an individual by a third party. Examples of credentials in real life include identification documents, driver's licenses, membership cards, keys, etc. A credential in a computer system can be a digitally signed document.

RT is a family of Role-based Trust management languages, which combines trust management and RBAC features. To define a trust management system, a language is needed for describing entities (principals and requesters), credentials and roles, which the entities play in the system. **RT_0** is a simple yet powerful trust management language. It is the core language of RT family, described in detail in [9]. It allows describing localized authorities for roles, role hierarchies, delegation of authority over roles and role intersections. All the subsequent languages add new features to RT_0, they are progressively increasing in expressive power and complexity. **RT_1** introduces *parameterized roles*, which can represent relationships between entities. **RT_2** extends RT_1 with *logical objects*, which can be used to represent permissions given to entities with respect to a group of logically related objects (resources). Those extensions can help in keeping the notation concise, but do not increase the expressive power of the language, because each combination of parameters in RT_1 and each permission to a logical object in RT_2 can be defined alternatively as a separate role in RT_0. **RT^T** language has been introduced to support threshold and separation of duties policies. Similar to a role, which defines a set of principals, a manifold role defines a set of principal sets, each of which is a set of principals whose cooperation satisfies the manifold role. **RT^D** provides mechanism to describe delegation of rights and role activations, which can express selective use of capacities and delegation of these capacities, which are useful when one wants to delegate authority temporarily. In many scenarios, an entity prefers not to use all his privileges, all the more, to delegate all his rights. **$RT_\ominus$** provides a carefully controlled form of non-monotonicity. The members of the RT family presented so far are monotonic: adding a credential to the system can only result in granting additional privileges, it cannot result in canceling an action, which was accepted before. It is a problem, because each policy statement or credential added to the system can only increase the capabilities and privileges granted to others. In [1], Czenko *et al*. argue that many access control decisions in complex distributed systems, like virtual communities, are hard to model in a purely monotonic language. They propose $RT_\ominus$, which adds to RT a restricted form of negation called *negation in context*. $RT_\ominus$ introduces a new operator $\ominus$ and the so called *exclusion* credential. It was created to manage trust in P2P applications and access control in virtual communities. The features of RT^T and RT^D can be combined together with the features of RT_0, RT_1 or RT_2. A more detailed treatment of RT family can be found in [8].

3.1. The Syntax of RT Family Languages

Basic elements of RT languages are *entities*, *role names*, *roles* and *credentials*. Entities represent principals that can define roles and issue credentials, and requesters that can make requests to access resources. An entity can, e.g., be a person or program identified by a user account in a computer system or a public key. We denote an entity by a name starting with an uppercase letter (or just an uppercase letter), e.g., *A*, *Alice*, *University*. Role names represent permissions that can be issued by entities to other entities or groups of entities. A role name is denoted by a string starting with a lowercase letter (or just a lowercase letter), like *r* or *student*. Roles denote sets of entities that have particular permissions granted according to the access control policy. Roles have the form of entity followed by a role name, separated by a dot, like *A.r* or *University.student*. Credentials define roles by appointing a new member of the role or by delegating authority

to the members of other roles. There are four types of credentials in RT_0, which are interpreted in the following way:

- $A.r \leftarrow B$ – *simple membership*: entity B is a member of role $A.r$;
- $A.r \leftarrow B.s$ – *simple inclusion*: role $A.r$ includes (all members of) role $B.s$. This is a delegation of authority over r from A to B;
- $A.r \leftarrow B.s.t$ – *linking inclusion*: role $A.r$ includes role $C.t$ for each C, which is a member of role $B.s$. This is a delegation of authority from A to all the members of the role $B.s$;
- $A.r \leftarrow B.s \cap C.t$ – *intersection inclusion*: role $A.r$ includes all the entities who are members of both roles $B.s$ and $C.t$. This is a partial delegation from A to B and C;
- $A.r \leftarrow B.s \ominus C.t$ – *exclusion*: all members of $B.s$ which are not members of $C.t$ are members of $A.r$.

A policy is a finite set of credentials.

The languages discussed in this paper can be used, in general, in very complex systems. Therefore, we present here only a simplified example, with the intention to illustrate the basic notions and the notation, with a focus on $RT_{\ominus}$ credentials.

Example 1. Suppose that John wants to share his pictures and movies using file sharing system. John restricts the access to his pictures to those of his friends, who are a members of Picture Club and he gave similar restriction to his movie, but he requires that his friends should be members of Movie Club instead of Picture Club.
John can also create a list of people who are forbidden to see the gallery of his private pictures, which means that people, who are on the list can see the general gallery of his pictures but not the private one.
The entire policy can be expressed as follows:

$$John.accessPic \leftarrow John.friend \cap John.pictureClub \quad (1)$$

$$John.accessMov \leftarrow John.friend \cap John.movieClub \quad (2)$$

$$John.privatePic \leftarrow John.accessPic \ominus John.blackList \quad (3)$$

Now, assume that the following credentials have been added:

$$John.friend \leftarrow Bob \quad (4)$$

$$John.friend \leftarrow Lily \quad (5)$$

$$John.friend \leftarrow Maria \quad (6)$$

$$John.friend \leftarrow Sofia \quad (7)$$

$$John.pictureClub \leftarrow Bob \quad (8)$$

$$John.pictureClub \leftarrow Etan \quad (9)$$

$$John.pictureClub \leftarrow Lily \quad (10)$$

$$John.movieClub \leftarrow Alice \quad (11)$$

$$John.movieClub \leftarrow Maria \quad (12)$$

$$John.movieClub \leftarrow Sofia \quad (13)$$

$$John.blackList \leftarrow Bob \quad (14)$$

Then one can conclude that, according to the policy, people who have access to John's pictures are *Bob* and *Lily*, but only *Lily* has access to his private gallery, and *Maria* and *Sofia* have access to John's movies.

4. The Set-Theoretic Semantics of *RT* Languages

A set-theoretic semantics of RT_0, which defines the meaning of a set of credentials as (monotone) function from the set of roles into the power set of entities, has been originally defined in [9]. A slightly different approach, closely related to the semantics of RT_0 language, was shown in [14], where various semantic readings of Simple Distributed Security Infrastructure (SDSI) were provided. A definition quoted in this section is a modified version of the same semantics, which has been introduced in [10], with addition of $\ominus$ operator.

Definition 1. The semantics of a set $\mathscr{P}$ of *RT* credentials, denoted by $\mathscr{S}_{\mathscr{P}}$, is the smallest relation S_i, such that:

1. $\mathscr{S}_0 = \emptyset$
2. $\mathscr{S}_{i+1} = \bigcup_{c \in \mathscr{P}} f(\mathscr{S}_i, c)$ for $i = 0, 1, \ldots$

which is closed with respect to function f, which describes the meaning of credentials in the following way (A, B, C, X, Y are entities):

$$\begin{aligned}
&f(\mathscr{S}_i, A.r \leftarrow X) = \{(A, r, X)\} \\
&f(\mathscr{S}_i, A.r \leftarrow B.s) = \{(A, r, X) : (B, s, X) \in \mathscr{S}_i\} \\
&f(\mathscr{S}_i, A.r \leftarrow B.s.t) = \bigcup_{C:(B,s,C)\in\mathscr{S}_i} \{(A, r, X) : (C, t, X) \in \mathscr{S}_i\} \\
&f(\mathscr{S}_i, A.r \leftarrow B.s \cap C.t) = \{(A, r, X) : (B, s, X) \in \mathscr{S}_i \\
&\qquad\qquad \wedge (C, t, X) \in \mathscr{S}_i\} \\
&f(\mathscr{S}_i, A.r \leftarrow B.s \ominus C.t) = \{(A, r, X \setminus Y) : (B, s, X) \in \mathscr{S}_i \\
&\qquad\qquad \wedge (C, t, Y) \in \mathscr{S}_i\}
\end{aligned}$$

Example 2. Set-theoretic semantics for Example 1

We use Example 1 from Section 3 to illustrate the definition of RT semantics.
The sequence of steps to compute consecutive relations $\mathscr{S}_i$ is shown in Table 1. Consecutive sections of the table describe relations $\mathscr{S}_0$ through $\mathscr{S}_3$. Each section of Table 1 has exactly one row, which corresponds to the issuer of the role, *John*. The columns of the table correspond to role names. This way, a cell of the table shows the set of entities, which are members of the respective role issued by *John*.
The starting relation $\mathscr{S}_0$ is, by definition, empty. According to Definition 1, only credentials (4)–(14) are

Table 1
The relations $\mathscr{S}_0$ through $\mathscr{S}_4$

John	friend	pictureClub	movieClub	blackList	accessPic	accessMov	privatePic
$\mathscr{S}_0$	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
$\mathscr{S}_1$	*Bob* *Lily* *Maria* *Sofia*	*Bob* *Etan* *Lily*	*Alice* *Maria* *Sofia*	*Bob*	ϕ	ϕ	ϕ
$\mathscr{S}_2$	*Bob* *Lily* *Maria* *Sofia*	*Bob* *Etan* *Lily*	*Alice* *Maria* *Sofia*	*Bob*	*Bob* *Lily*	*Maria* *Sofia*	ϕ
$\mathscr{S}_3$	*Bob* *Lily* *Maria* *Sofia*	*Bob* *Etan* *Lily*	*Alice* *Maria* *Sofia*	*Bob*	*Bob* *Lily*	*Maria* *Sofia*	*Lily*

mapped in $\mathscr{S}_0$ into nonempty sets by function f. These sets are shown in relation $\mathscr{S}_1$ in Table 1. In $\mathscr{S}_1$, credentials (1) and (2) are mapped into instances $(John, accessPic, Bob)$, $(John, accessPic, Lily)$, $(John, accessMov, Maria)$, and $(John, accessMov, Sofia)$ of relation $\mathscr{S}_2$, and in $\mathscr{S}_2$, credential (3) is mapped into instances $(John, privatePic, Lily)$. The resulting relation $\mathscr{S}_3$ cannot be changed using the given set of credentials, hence

$$\mathscr{S}_{\mathscr{P}} = \mathscr{S}_3$$

and it is the end of the set-theoretic semantics for Example 1.

5. The Inference System over *RT* Credentials

The member sets of roles can also be calculated in a more convenient way (than set-theoretic semantics) using an inference system, which defines an operational semantics of *RT* languages. An inference system consists of an initial set of formulae that are considered to be true, and a set of inference rules, that can be used to derive new formulae from the known ones.

Let $\mathscr{P}$ be a given set of *RT* credentials. The application of inference rules of the inference system will create new credentials, derived from credentials of the set $\mathscr{P}$. A derived credential c will be denoted using a formula $\mathscr{P} \succ c$, which should be read: credential c can be derived from a set of credentials $\mathscr{P}$.

Definition 2. The initial set of formulae of an inference system over a set $\mathscr{P}$ of *RT* credentials are all the formulae

$$c \in \mathscr{P},$$

for each credential c in $\mathscr{P}$.

The inference rules of the system are the following:

$$\frac{c \in \mathscr{P}}{\mathscr{P} \succ c} \quad (W_1)$$

$$\frac{\mathscr{P} \succ A.r \leftarrow B.s \qquad \mathscr{P} \succ B.s \leftarrow X}{\mathscr{P} \succ A.r \leftarrow X} \quad (W_2)$$

$$\frac{\mathscr{P} \succ A.r \leftarrow B.s.t \qquad \mathscr{P} \succ B.s \leftarrow C \qquad \mathscr{P} \succ C.t \leftarrow X}{\mathscr{P} \succ A.r \leftarrow X} \quad (W_3)$$

$$\frac{\mathscr{P} \succ A.r \leftarrow B.s \cap C.t \qquad \mathscr{P} \succ B.s \leftarrow X \qquad \mathscr{P} \succ C.t \leftarrow X}{\mathscr{P} \succ A.r \leftarrow X} \quad (W_4)$$

$$\frac{\mathscr{P} \succ A.r \leftarrow B.s \ominus C.t \qquad \mathscr{P} \succ B.s \leftarrow X \qquad \mathscr{P} \succ C.t \leftarrow Y}{\mathscr{P} \succ A.r \leftarrow X \setminus Y} \quad (W_5)$$

The five kinds of credentials described in Section 4 are handled by the rules above. The rules should be self-explanatory.

5.1. Soundness and Completeness of Inference System over RT^T Credentials

There could be a number of inference systems defined over a given language. To be useful for practical purposes an inference system must exhibit two properties. First, it should be sound, which means that the inference rules could derive only formulae that are valid with respect to the semantics of the language. Second, it should be complete, which means that each formula, which is valid according to the semantics, should be derivable in the system.

Due to space constraints, we only present sketches of proofs and proofs for (W_5) formula (introduced in the $RT_{\ominus}$ language), full proofs for the rest formulae can be found in [15]. The semantics of a set $\mathscr{P}$ of *RT* credentials, defined by the inference system, is given by a set of all the formulae of the type: $\mathscr{P} \succ A.r \leftarrow X$.

To prove the soundness of such a formula, one must prove that the triple (A,r,X) belongs to the semantics $\mathscr{S}_{\mathscr{P}}$ of the set of credentials $\mathscr{P}$. Let us first note that all the formulae $\mathscr{P} \succ A.r \leftarrow X$, such that $A.r \leftarrow X \in \mathscr{P}$ are sound. This is proved in Lemma 1.

Lemma 1. If $A.r \leftarrow X \in \mathscr{P}$ then $(A,r,X) \in \mathscr{S}_{\mathscr{P}}$.

Proof. The relation $\mathscr{S}_{\mathscr{P}}$, which defines the semantics of $\mathscr{P}$, is a limit of a monotonically increasing sequence of sets $\mathscr{S}_0, \mathscr{S}_1, \ldots$, such that $\mathscr{S}_0 = \emptyset$. According to Definition 1: $f(\mathscr{S}_0, A.r \leftarrow X) = (A,r,X)$. Hence,$(A,r,X) \in \mathscr{S}_1$ and because $\mathscr{S}_1 \subseteq \mathscr{S}_{\mathscr{P}}$ then $(A,r,X) \in \mathscr{S}_{\mathscr{P}}$.

■

To prove the soundness of the inference system over $\mathscr{P}$, we must prove the soundness of each formula $\mathscr{P} \succ A.r \leftarrow X$, which can be derived from the set $\mathscr{P}$. This is proven in Theorem 1.

Theorem 1. If $\mathscr{P} \succ A.r \leftarrow X$ then $(A,r,X) \in \mathscr{S}_{\mathscr{P}}$.

Proof. By induction with respect to the number n of inference steps, which are needed to derive a formula $\mathscr{P} \succ A.r \leftarrow X$. If $n = 1$ then the formula $\mathscr{P} \succ A.r \leftarrow X$ could be derived only using rule (W_1), because the premises of only this rule are axioms. Hence, the thesis is true according to Lemma 1. For the inductive step, assume that the thesis is true if the number of inference steps was not greater than n. Then, it is possible to show that it is true also in a case when the number of inference steps equals $n+1$. Since any one of the rules (W_2) through (W_5) could be used in the last $(n+1)$ step of inference, all those four cases should be discussed separately, analyzing the premises and using Definition 1 to show that the thesis holds. As it was mentioned before, a proof for the rule (W_5) is provided, see [15] for rules (W_2)–(W_4).

($\mathbf{W_5}$) The fist premise of (W_5) cannot be derived otherwise than using (W_1). Hence, $A.r \leftarrow B.s \ominus C.t \in \mathscr{P}$. The second premise of (W_5): $\mathscr{P} \succ B.s \leftarrow X$ was derived from $\mathscr{P}$ using at most n steps of inference, hence, $(B,s,X) \in \mathscr{S}_{\mathscr{P}}$ according to the inductive hypothesis. By Definition 1, there exists such $\mathscr{S}_i$ that $(B,s,X) \in \mathscr{S}_i$. Similarly, in the case of the third premise of (W_5): $\mathscr{P} \succ C.t \leftarrow Y$, there exists such $\mathscr{S}_j$ that $(C,t,Y) \in \mathscr{S}_j$. Let k be the maximum of (i,j). Then $(B,s,X) \in \mathscr{S}_k$, $(C,t,Y) \in \mathscr{S}_k$ and $(A,r,X \setminus Y) \in f(\mathscr{S}_k, A.r \leftarrow B.s \ominus C.t)$ according to $f(\mathscr{S}_i, A.r \leftarrow B.s \ominus C.t) = \{(A,r,X \setminus Y) : (B,s,X) \in \mathscr{S}_i \wedge (C,t,Y) \in \mathscr{S}_i\}$. Because $f(\mathscr{S}_k, A.r \leftarrow B.s \ominus C.t) \subseteq \mathscr{S}_{k+1} \subseteq \mathscr{S}_{\mathscr{P}}$ then $(A,r,X \setminus Y) \in \mathscr{S}_{\mathscr{P}}$.

■

To prove the completeness of the inference system over a set $\mathscr{P}$ of RT credentials, one must prove that a formula $\mathscr{P} \succ A.r \leftarrow X$ can be derived using inference rules for each element $(A,r,X) \in \mathscr{S}_{\mathscr{P}}$. This is proven in Theorem 2.

Theorem 2. If $(A,r,X) \in \mathscr{S}_{\mathscr{P}}$ then $\mathscr{P} \succ A.r \leftarrow X$.

Proof. Assume $(A,r,X) \in \mathscr{S}_{\mathscr{P}}$. By Definition 1, there exists such $i \geq 0$ and such $c \in \mathscr{P}$ that $(A,r,X) \in f(\mathscr{S}_i, c)$. The proof of the thesis is by induction with respect to the value of index i. If $i = 0$ then credential c must take the form of $A.r \leftarrow X$. This is because $\mathscr{S}_0 = \emptyset$ and $f(\mathscr{S}_0, d) = \emptyset$ for each credential d other than $A.r \leftarrow X$. Hence, $A.r \leftarrow X \in \mathscr{P}$ and the formula $\mathscr{P} \succ A.r \leftarrow X$ can be derived using rule (W_1). For the inductive step, assume that the thesis is true, if the value of index i in the expression $(A,r,X) \in f(\mathscr{S}_i, c)$ was not greater than n. Then it suffices to show that it is true also in a case when the value of index i equals $(n+1)$. Assume $(A,r,X) \in \mathscr{S}_{\mathscr{P}}$ and $(A,r,X) \in f(\mathscr{S}_{n+1}, c)$ for a certain $c \in \mathscr{P}$. The credential c can take one of the five forms allowed in RT_0 and $RT_{\ominus}$. Each of these types of credentials should be discussed separately, showing that it can be derived using one of the rules (W_1)–(W_5). Definition 1 is used in all cases except $\mathbf{c} = \mathbf{A.r} \leftarrow \mathbf{B}$, which trivially results from (W_1). As it was mentioned before, a proof for $c = A.r \leftarrow B.s \ominus C.t$ is provided, see [15] for rules (W_2)–(W_4).

$\mathbf{c} = \mathbf{A.r} \leftarrow \mathbf{B.s} \ominus \mathbf{C.t}$: If $(A,r,X) \in f(\mathscr{S}_{n+1}, A.r \leftarrow B.s \ominus C.t)$, then according to Definition 1, $f(\mathscr{S}_i, A.r \leftarrow B.s \ominus C.t) = \{(A,r,X \setminus Y) : (B,s,X) \in \mathscr{S}_i \wedge (C,t,Y) \in \mathscr{S}_i\}$, there exist two sets of entities Z, Y such that $Z \setminus Y = X$ and $(B,s,Z) \in \mathscr{S}_{n+1}$ and $(C,t,Y) \in \mathscr{S}_{n+1}$. Hence, there exist credentials c_1, c_2 such that $(B,s,Z) \in f(\mathscr{S}_n, c_1)$ and $(C,t,Y) \in f(\mathscr{S}_n, c_2)$. This implies that $(B,s,Z) \in \mathscr{S}_{\mathscr{P}}$ and $(C,t,Y) \in \mathscr{S}_{\mathscr{P}}$, hence, $\mathscr{P} \succ B.s \leftarrow Z$ and $\mathscr{P} \succ C.t \leftarrow Y$ according to the inductive hypothesis. $\mathscr{P} \succ A.r \leftarrow X$ is a conclusion of rule (W_5).

■

A conclusion from Theorem 1 and Theorem 2 is such that the inference system of Definition 1 is sound and complete with respect to the semantics of RT credentials. This way, the inference system gives an operational definition of RT semantics (for RT_0 and $RT_{\ominus}$) and it proves that the inference system provides an alternative way of presenting the semantics of RT.

Example 3. (Inference system for Example 1):

We use the inference system to formally derive entities which can have access to *John's* galleries. Using credentials (1)–(14) according to rule (W_1) it can infer:

$$\frac{John.accessPic \leftarrow John.friend \cap John.picClub \in \mathscr{P}}{\mathscr{P} \succ John.accessPic \leftarrow John.friend \cap John.picClub}$$

$$\frac{John.accessMov \leftarrow John.friend \cap John.movieClub \in \mathscr{P}}{\mathscr{P} \succ John.accessMov \leftarrow John.friend \cap John.movieClub}$$

$$\frac{John.privatePic \leftarrow John.accessPic \ominus John.blackList \in \mathscr{P}}{\mathscr{P} \succ John.privatePic \leftarrow John.accessPic \ominus John.blackList}$$

$$\frac{John.friend \leftarrow Bob \in \mathscr{P}}{\mathscr{P} \succ John.friend \leftarrow Bob}$$

$$\frac{John.friend \leftarrow Lily \in \mathscr{P}}{\mathscr{P} \succ John.friend \leftarrow Lily}$$

$$\frac{John.friend \leftarrow Maria \in \mathscr{P}}{\mathscr{P} \succ John.friend \leftarrow Maria}$$

$$\frac{John.friend \leftarrow Sofia \in \mathscr{P}}{\mathscr{P} \succ John.friend \leftarrow Sofia}$$

$$\frac{John.pictureClub \leftarrow Bob \in \mathscr{P}}{\mathscr{P} \succ John.pictureClub \leftarrow Bob}$$

$$\frac{John.pictureClub \leftarrow Etan \in \mathscr{P}}{\mathscr{P} \succ John.pictureClub \leftarrow Etan}$$

$$\frac{John.pictureClub \leftarrow Lily \in \mathscr{P}}{\mathscr{P} \succ John.pictureClub \leftarrow Lily}$$

$$\frac{John.movieClub \leftarrow Alice \in \mathscr{P}}{\mathscr{P} \succ John.movieClub \leftarrow Alice}$$

$$\frac{John.movieClub \leftarrow Maria \in \mathscr{P}}{\mathscr{P} \succ John.movieClub \leftarrow Maria}$$

$$\frac{John.movieClub \leftarrow Sofia \in \mathscr{P}}{\mathscr{P} \succ John.movieClub \leftarrow Sofia}$$

$$\frac{John.blackList \leftarrow Bob \in \mathscr{P}}{\mathscr{P} \succ John.blackList \leftarrow Bob}$$

Then, using credentials (1), (4), (5), (8), (10), and rule (W_4) we infer:

$$\frac{\mathscr{P} \succ John.accessPic \leftarrow John.friend \cap John.pictureClub \quad \mathscr{P} \succ John.friend \leftarrow Bob \quad \mathscr{P} \succ John.pictureClub \leftarrow Bob}{\mathscr{P} \succ \mathbf{John.accessPic} \leftarrow \mathbf{Bob}}$$

$$\frac{\mathscr{P} \succ John.accessPic \leftarrow John.friend \cap John.pictureClub \quad \mathscr{P} \succ John.friend \leftarrow Lily \quad \mathscr{P} \succ John.pictureClub \leftarrow Lily}{\mathscr{P} \succ \mathbf{John.accessPic} \leftarrow \mathbf{Lily}}$$

showing that the group of people who can see John's pictures are *Bob* and *Lily*.

In the next step the newly inferred credentials and additionally credentials (3) and (14) with the rule (W_5) is used:

$$\frac{\mathscr{P} \succ John.privatePic \leftarrow John.accessPic \ominus John.blackList \quad \mathscr{P} \succ John.accessPic \leftarrow Bob \quad \mathscr{P} \succ John.accessPic \leftarrow Lily \quad John.blackList \leftarrow Bob}{\mathscr{P} \succ \mathbf{John.privatePic} \leftarrow \mathbf{Lily}}$$

showing that only *Lily* can see John's private collection.

Additionally, if we want to find a group of people, who can see John's movies, we can do this using credentials (2), (6), (7), (12), (13), and rule (W_4). We infer:

$$\frac{\mathscr{P} \succ John.accessMov \leftarrow John.friend \cap John.movieClub \quad \mathscr{P} \succ John.friend \leftarrow Maria \quad \mathscr{P} \succ John.movieClub \leftarrow Maria}{\mathscr{P} \succ \mathbf{John.accessMov} \leftarrow \mathbf{Maria}}$$

$$\frac{\mathscr{P} \succ John.accessMovie \leftarrow John.friend \cap John.movieClub \quad \mathscr{P} \succ John.friend \leftarrow Sofia \quad \mathscr{P} \succ John.movieClub \leftarrow Sofia}{\mathscr{P} \succ \mathbf{John.accessMov} \leftarrow \mathbf{Sofia}}$$

showing that the group of people who can see John's movies are *Maria* and *Sofia*.

6. Conclusions

This paper deals with modeling of trust management systems in decentralized and distributed environments. The modeling framework is a family of Role-based Trust management languages, especially RT_0 and $RT_\ominus$ languages. Two types of semantics for *RT* credentials have been introduced in the paper.

A set-theoretic semantics of *RT* languages is defined as a relation over a set of roles and a set of entities.

An operational semantics of *RT* languages is defined as an inference system, in which credentials can be derived from an initial set of credentials using a set of inference rules. The semantics is given by the set of resulting credentials of the type $A.r \leftarrow X$, which explicitly show a mapping between roles and sets of entities.

References

[1] M. R. Czenko *et al.*, "Nonmonotonic Trust Management for P2P Applications", in Proc. *1st Int. Worksh. Secur. Trust Manag. STM 2005*, Milan, Italy, 2005.

[2] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models", *IEEE Comp.*, vol. 29, pp. 38–47, 1996.

[3] M. Blaze, J. Feigenbaum, and M. Strauss, "Compliance checking in the PolicyMaker trust management system", in *Proc. 2nd Int. Conf. Financial Cryptogr.*, London, UK, 1998, pp. 254–274.

[4] M. Blaze, J. Feigenbaum, and A. D. Keromytis, "The role of trust management in distributed systems security" in *Secure Internet Programming*, J. Vitek, C. Damsgaard Jensen, Eds. London: Springer, 1999, pp. 185–210.

[5] D. Clarke *et al.*, "Certificate chain discovery in SPKI/SDSI", *J. Comp. Secur.*, vol. 9, pp. 285–322, 2001.

[6] P. Chapin, C. Skalka, and X. S. Wang, "Authorization in trust management: Features and foundations", *ACM Comput. Surv.*, vol. 3, pp. 1–48, 2008.

[7] M. R. Czenko, S. Etalle, D. Li, and W. H. Winsborough, "An Introduction to the Role Based Trust Management Framework RT", Tech. Rep. TR-CTIT-07-34, Centre for Telematics and Information Technology University of Twente, Enschede, The Netherlands, 2007.

[8] N. Li, J. Mitchell, W. Winsborough, "Design of a Role-Based Trust-Management Framework", in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, 2002, pp. 114–130.

[9] N. Li, W. Winsborough, and J. Mitchell, "Distributed credential chain discovery in trust management", *J. Comput. Secur.*, vol. 11, no. 1, pp. 35–86, 2003.

[10] D. Gorla, M. Hennessy, and V. Sassone, "Inferring dynamic credentials for role-based trust management", in *Proc. 8th Conf. Princip. Pract. Declarat. Program. PPDP 2006*, Venice, Italy, 2006. New York: ACM, 2006, pp. 213–224.

[11] A. Felkner and K. Sacha, "The semantics of role-based trust management languages", in *Advances in Software Engineering Techniques*, T. Szmuc, M. Szpyrka, and J. Zendulka, Eds. LNCS, vol. 7054, pp. 179–189. Heidelberg: Springer, 2012.

[12] A. Felkner and A. Kozakiewicz, "RT^T_+ – time validity constraints in RT^T language", *J. Telecom. Inform. Technol.*, no. 2, pp. 74–82, 2012.

[13] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management", in *Proc. 17th IEEE Symp. Secur. Priv. S&P 1996*, Oakland, CA, USA, 1996, pp. 164–173.

[14] N. Li and C. Mitchell, "Understanding SPKI/SDSI using first-order logic", *Int. J. Inf. Secur.*, vol. 5, no. 1, pp. 48–64, 2006.

[15] A. Felkner, "Zarządzanie zaufaniem oparte na rolach" ("Role-based Trust Management"), PhD Thesis, Faculty of Electronics and Information Technology, Warsaw University of Technology, 2009.

[16] A. Felkner and A. Kozakiewicz, "Time validity in role-based trust management inference system", *Sec. and Trust Comput., Data Manag., and Appl. Commun. in Comp. and Inform. Sci.*, vol. 187, pp. 7–15, 2011.

[17] K. Lasota and A. Kozakiewicz, "Model of user access control to virtual machines based on RT – family trust management language with temporal validity constrains – practical application", *J. Telecom. Inform. Technol.*, no. 3, pp. 13–21, 2012.

Anna Felkner graduated from the Faculty of Computer Science of Białystok University of Technology (M.Sc., 2004) and the Faculty of Electronics and Information Technology of Warsaw University of Technology (Ph.D., 2010). At present she is an Assistant Professor at Network and Information Security Methods Team in NASK Research Division. Main scientific interests concern the security of information systems, especially access control and trust management.

E-mail: anna.felkner@nask.pl
Research and Academic Computer Network (NASK)
Wąwozowa st 18
02-796 Warsaw, Poland