URL: http://www.elsevier.nl/locate/entcs/volume63.html 19 pages

Enforcing policies with privacy guardians

Radu Serban [1] Reind van de Riet [2]

Information Management and Software Engineering Department
Vrije Universiteit
Amsterdam, The Netherlands

Abstract

Reasoning about privacy in electronic environments populated with privacy-concerned agents that exchange personal data requires control of ownership and proving the right of possession of a piece of data. The privacy policy expressed by an individual for his personal data can be enforced by a context-aware mobile agent, called alter-ego, accompanying the personal data disclosed.

We discuss the first steps towards a formal framework for expressing policies on information disclosure and their integration in the behavior specification of the alter-egos, that enables characterization of an environment manipulating personal data from the privacy perspective.

1 Introduction

In systems that manipulate personal data, the transfer of data between individuals is based on control of ownership and jurisdiction. An environment may have several active components, each executing a program, but it can be considered as a special agent running its own program and having its own security policy (including obligations and restrictions regarding the programs run by the principals that act in that environment). Privacy policies defined by an individual to restrict access of unauthorized agents to his personal profile must be enforced by mobile agents termed alter-egos, which control the exchange of private information with other, possibly untrusted, agents, and check whether the security policy of a particular environment is compatible with the privacy policies of the agents working in that environment.

In this agent-oriented framework, policies and commitment are expressed mainly using deontic operators (cf. [9,5]), and agents need to assess the *privacy compliance* of the security policy of that environment with respect to the

[1] Email: serbanr@cs.vu.nl
[2] Email: vdriet@cs.vu.nl

privacy policies expressed by its potential users, to verify whether an agent would be able to work in that environment without violating the security constraints and without facing the risk of a privacy violation.

The context: mobile agents in Cyberspace. In our previous work ([14]), we have proposed the architecture of a privacy assistant, a context-aware program that helps an individual to control his personal data and keeps him informed with respect to his privacy status. It monitors online transactions, ensures identity and trust management, investigates privacy violations and suggests corrective measures.

To protect personal information, the privacy assistant creates and coordinates several alter-egos, termed fireballs, encapsulating personal information together with their policies for specific applications, with the goal of enforcing control by the owner of the data. The fireball (denoted FB) is a privacy guardian agent, a filtering mechanism that ensures content protection for personal data transferred during an electronic transaction. The FBs are primitive context-aware mobile agents that function in special environments in which commitments for informedness and for performing privacy compliance checks hold. These environments ([14]) mediate interaction between agents, keep the fireballs informed with respect to their structure and the roles played by other participants, and are trusted not to tamper with the agents and to ensure that the exchanges of data are correctly formatted, thus preventing unauthorized use of personal information encapsulated in agents. Based on knowledge communicated by credible sources, the FB is able to derive some risk parameters of the environment and notifies its owner if privacy exceptions are detected. It also takes preventive and corrective measures for protecting sensitive data: using pseudonyms, changing the privacy policy or making the embedded data unusable (erasing, altering, or scrambling the data). Changes in the knowledge of one FB result in updates sent to known fixed locations of other FBs of the same individual or to the privacy assistant.

When entering a MMAP (Mobile Multi-Agent Platform) the FB is acknowledged of the program that must be run in that environment (including its security constraints), and a set of commited actions from the other participants, specific for the roles involved in the protocol. Its ability to assess the privacy-compliance of that protocol depends on the accuracy of the information about the other participants, on the perceived commitments to enforce the MMAP security policy, and on the trust in the other participants.

The FB then negotiates and commits to an interface offered to that MMAP. The committed interface can be changed if privacy threats are detected, in agreement with MMAP's security policy.

Our paper is structured as follows: section 2 discusses the enforcement of privacy and the form of privacy rules; section 3 proposes a formal framework for modeling and reasoning about privacy assistants, whose use is illustrated with an example in section 4.

2 Expressing privacy policies

Privacy is the right of individuals or groups to establish and control the accessibility of their information, properties or actions, that would allow others to learn about them or to contact them. It entails the right to secrecy and to non-interference from outsiders with the domains of ownership and jurisdiction of one individual (e.g., protection from unwanted e-mail).

An individual specifies a set of sensitive information (structure of his profile, roles played, knowledge acquired, preferences), and controlling privacy basically amounts to limiting the visibility of other agents to a particular piece of data, expressed by the representative of an individual (an alter-ego) who has jurisdiction over that piece of data in a given evaluation context.
The objectives of protection for an alter-ego may be: the connection between an attribute and a value, a tuple of attribute-value pairs, or an *aggregate* (delimited by [ ]) - a combination of attribute names, attribute values, attribute-value pairs and tuples, that can describe a stream of data collected by eavesdropping a line or a user profile. Protection of personal information is done at the *object level* - protection of the value of a personal attribute (a secret), or at the *meta-knowledge level* - protection of the existence of a strategy, membership, policy or belief (a secrecy, according to definitions in [6,2]).

Enforcing privacy principles. Commitment of the principals collecting or exchanging personal data has been acknowledged ([8,10]) as essential for enforcing privacy. Wilhelm ([21,20]) argues that a system that ensures privacy protection must guarantee **notification** and **control** of the principal about the flow and usage of data related to him.

The EU Directive on Data Protection and the work for an Open Profiling Standard suggest three privacy protection principles: 1. **purpose-binding and informed consent**: any request for personal data must state clearly the intended use of that data, and the data can be used solely for that purpose and only after the owner of that data has accepted this use; 2. **need-to-know and control by source**: any requester of personal data must prove that the requested data is needed for a meaningful processing and the data can be controlled by its source (owner); 3. **appropriate value exchange**: no personal data should be requested without offering something valuable (service, money, information) in exchange.

An individual has to decide whether he wants to disclose some data at all or, once his personal data is disclosed, an alter-ego accompanies it, filters the notifications that must be sent to the individual, and enforces the **control by source** and the **purpose binding** principles. When issuing a request for a piece of personal data, an agent must present an authenticable identity, an acceptable motivation and a meaningful purpose for the future use of that data; the personal data transferred must be accompanied by proofs of ownership (a certificate) released by the source of the personal data, that can lead to its originator. These well-formedness checks allow the detection

of illegal disclosures. To enforce them, we need a Public Key Infrastructure (PKI) and a special delegation, purpose-binded delegation of ownership, i.e. a non-disclosure agreement between two agents A and B that exchange personal information: A agrees to disclose fact φ to B if B specifies the intended use for the requested information, and B commits to not disclose it to third parties for other purposes than the one specified.

Three checks are performed by the recipient of a request for personal data: 1. whether the requester has presented a valid public key; 2. whether the request contains a valid signature of the requester on the purpose binding specification; 3. whether the purpose specified is acceptable for the data owner (can be motivated by the role played by the requester). These checks may represent a threat to the privacy of the requester of the personal data, who is not always willing to disclose his purpose, because it is private. Ideally, the proofs of privacy-compliant requests can be done without revealing explicitly the purposes of one's actions, but only their compliance with the purpose declared to an authorized owner of personal data.

To enforce purpose-binding without threatening privacy, a request sent by A to B must be structured as follows: $\{motivation{+}purpose, \{TS, h(purpose), ?data?\}_{k_A^-}\}_{k_B}$ where TS is a timestamp produced when assembling the message; *purpose* describes the intended processing (action(s) to be performed) on the requested *data*; *motivation* describes the context in which this request occurs and why it is necessary for the disclosure to take place; $\{M\}_{k_B}$ denotes message M encrypted with key k_B, and $\{M\}_{k_A^-}$ denotes message M signed using key k_A^-. The answer must contain $\{\{TS_1, h(purpose), data, \{TS, h(purpose), ?data?\}_{k_A^-}, proof\}_{k_B^-}\}_{k_A}$ where *proof* reveals legal ownership - a chain of certificates leading to an owner of *data* certified by a trusted party. In this way, B cannot repudiate its request, A cannot repudiate its disclosure, and B can prove to any independent verifier that he posesses legal data (since its purpose specified for request is present in the certificate released), without the verifier being required to learn the explicit purpose or the data exchanged.

Example: A purpose specification for the online bookshop B to use the credit card number of client C for paying two copies of book b_1 and one copy of book b_2 is: $purpose = \{< 2 > use_B([data = CCN_C; target = b_1]) \wedge < 1 > use_B([data = CCN_C; target = b_2])\}$.

Expressing security policies. A modal operator 'right-to-know' has been proposed (cf. [1,5]) to express the confidentiality requirement in security policies: $K_A\varphi \rightarrow R_A\varphi$, is read "if A knows φ then A should have the permission to know φ". System S is secure w.r.t. a subject A iff for all times τ and all formulae φ expressing facts exchanged in the system, the formula $K_A^\tau(\varphi) \rightarrow R_A^\tau(\varphi)$ is valid.
In our approach, this operator would denote the permission to learn a proposition about personal data for a given purpose, with the informed consent of a legal owner of that data, such as in the sentence 'Agent A is legally enti-

tled(permitted) to learn(know) and use data D for purpose P, because it was authorized by X'.

A **security policy** is a set of rules specifying what actions the agents are permitted, obliged or prohibited to do, on which objects, aimed at protecting the principals in their interactions with the system and at achieving separation of duties between the various system components. Security rules can be expressed as suggested in [16,15,4]: **'(When regulation R applies and is enforced) If condition C is satisfied, then agent A is obliged, permitted or prohibited to do action α'.**

A part of the security policy deals with disclosures - the **disclosure policy**; it prescribes completely the intended disclosures of data which was provided by third parties to the system components.
A **privacy policy** is a special type of security policy, in which the rules specify allowed and forbidden inferences that regard personal data.

The statements in a security policy (cf. [7]): 1. restrict the operations allowed for principals to execute on the objects (*access control rules*); 2. restrict what principals can infer about objects by observing the system behavior (*information flow rules*); 3. restrict principals from denying others the use of a resource (*availability rules*). A special class of availability rules consists of *aggregation-prevention rules*, of the form: 'Either information i_1 or information i_2 can be learnt by agent A, but A must not learn both of them'.
These types of rules can be expressed uniformly with the notion of visibility rule.

A **visibility rule** is the most frequent type of privacy rules, expressed as an access control rule using deontic operators (permitted, forbidden, obliged), and includes:

- the type of restriction (permission, interdiction or obligation);
- the piece of information whose visibility is restricted;
- the issuer of the visibility restriction, i.e. an entity who has permanent or temporary jurisdiction of the piece of information envisaged by the rule;
- the context of restriction (properties of the environment that must hold) and the allowed types of access (the actions allowed and the purpose of those actions);
- the target of restriction (a category of entities, specified by name or by properties).

The general form of a visibility rule is:
$DeMod^{Time/Ctx}_{Issuer}([Info = I; Actor = A; Context = C; Action = \alpha])$
where $DeMod$ can be $Perm$, $Forbid$ or Obl, interpreted as '$Issuer$ permits, forbids or considers obligatory, respectively, the action(s) denoted by α performed by actor A, in context C, upon the object I'. $Issuer$, α, A, C, I may be expressed as logical formulae. The traditional deontic operators, P, F, O, correspond in our model to the statements $Permitted_A(\alpha)$, $Forbidden_A(\alpha)$,

$Obliged_A(\alpha)$, i.e. 'A is permitted, forbidden, or obliged to α'. The connection between the deontic modalities and the traditional deontic operators is: $DeMod_{Issuer}([Info = I; Actor = A; Context = C; Action = \alpha]) \rightarrow DeOp_A(\alpha)$, where DeOp is one of $Permitted$, $Forbidden$ or $Obliged$ corresponding to the $DeMod$ used.

The distinction between security and privacy rules expressed as visibility constraints is the role played by the issuer of the rule, the type of the data which is protected and the jurisdiction of the issuer over the data: the issuer of a privacy rule is the individual who possesses it or a principal acting on his behalf, while the issuer of a security rule is an authority. Also, security policies contain quite often access control rules, while disclosure policies and security policies contain more availability and information flow rules.

Privacy rules are statements issued by an individual, by an alter-ego delegated by him, or by an entity which is able to enforce them. They express restrictions of the visibility or access to the data, properties or actions of an individual or group. They may refer to conditions from the environment in which the data is used or depend on the history of interactions or on the satisfaction of certain external events. The principals who issue privacy policies must have an authenticated identity and proof of ownership of the data to which the rule is linked.

A privacy policy is a set of visibility rules, expressed as a declaration of permission or interdiction:
$Decl_{X \rightarrow \star}(Perm_X([Inf_1; Actor_1; Act_1; Ctx_1], \ldots, [Inf_n; Actor_n; Act_n; Ctx_n]))$

Examples of visibility rules.

1. 'Principal A_1 gives permission to A_2 to access A_1's record (name and address) in any context, for reading or transfering it to a third party':
$Perm_{A_1}([Info = [name_{A_1}; address_{A_1}]; Actor = \{A_2\}; Context = *; Action = \{read, transfer\}])$.

2. A_1 forbids any operation with his credit card number, performed by a company, if no order has been placed by A_1: $Forbid_{A_1}([Info = CCN_{A_1}; Actor = \{X : \neg Has_X(type = company)\}; Context = \neg Did_{A_1}(Order_{A_1 \rightarrow X}(*))\}; Action = \{*\}])$.

3. 'All users subscribed to the current location system will be tracked, and their position available to all subscribers, but archieved data older than one week is removed'. This rule, combining availability of location and reciprocity, may be part of the disclosure policy offered by a location tracking service.

The goal of enforcing the privacy policy of principal P_i is to prevent that the set of local knowledge hold by any of the other principals $P_j, j \neq i$ at any point in time conflicts with the privacy policy of P_i.

Certain types of privacy rules are enforceable without difficulty, while others depend on an awareness mechanism, and others cannot be enforced without additional commitments of the participants or control measures. The failure to meet a privacy rule is defined as a **privacy exception**, which may be of two types: **privacy conflict** (exception caused by an attempt to gain access to protected data) or **privacy violation** (exception caused by the use of data obtained illegally). Ignoring privacy conflicts can lead to privacy violations.

3 A formal approach

Our formal framework for modelling mobile multiagent systems, regulations and alter-egos' properties builds upon a multi-agent system model (such as the framework for executing discrete temporal logics presented by Wooldridge ([22])). It includes an execution model for alter-egos, a meta-language for modeling agent protocols, properties and actions, and some inference rules for the special predicates that substitute the epistemic operators from modal logics. In this paper, we focus on the execution model of the alter-egos, and only briefly describe the language for modelling agents interactions and the inference rules.

Meta-language for modeling agent properties and actions

For modeling and reasoning about agent systems we use a logical language FBML (FireBall Modelling Language), essentially a many-sorted first order language with special predicates corresponding to traditional epistemic and deontic operators (as presented by Cuppens et al, Syverson et al [11,4,3])), but also capturing the notion of time and action. Its semantics is defined in terms of a possible-worlds models. We distinguish between agents (forming a set Ag), actions (forming a set Ac), data items (forming a set Data), and formulae (making up the set $pow(Form(FBML))$).

Syntax. The set of formulae in FBML (denoted $Form(FBML)$) is generated from a combination of predicates, variables, constants and operators. The predicates represent the names of events, actions or agent properties. The variables denote agents and data items, as uppercase letters (A, B, C, ...), or generic variables (?X?, *Any*, *Unknown*). The constants represent agents (me, all, $a_1, a_2, \ldots, a_n$) and data items (attribute-value pairs, or aggregates). $\star$ and λ are special generic variables, corresponding to 'Anything' and 'None'.

The set of well-formed formulae (WFF) in FBML is defined by the usual construction rules of the propositional logic and by the following rule: if $\varphi \in WFF(FBML)$ is a well-formed formula in FBML language, $\tau \in \mathbf{N}$ is a time point, $Ctx \in pow(Form(FBML)) \cup \{\star, \lambda\}$ describes specific conditions for the context in which the formulae are evaluated, and $A \in Ag$ is an agent, then $\exists A : \varphi, \forall A : \varphi \in WFF(FBML)$ and $DeMod^{\tau/Ctx}_{ActorExpr}(D), Pred^{\tau/Ctx}_{ActorExpr}(\varphi) \in WFF(FBML)$, where Pred is a predicate denoting actions, relations, or domain-specific properties; *ActorExpr* is an expression denoting the agents involved in an action or the agent whose property is expressed; *DeMod* is a deontic modality, and $D \in Data$ is the expression of a data item - an aggregate consisting of a sequence of attribute-value pairs or values, separated by ';' and delimited by '[ ]'.

Examples of WFFs in FBML:

1. 'A believes B authorizes A to store B's public key': $B_A(Authorizes_{B \to A}(Has_A([k_B])))$
2. 'A declares to all agents that he forbids access to his data as specified by

aggregate Acc_1'. $_{Decl_{A\rightarrow all}(Perm_A(Acc_1))}$. Note that privacy policies can also be expressed as WFFs in FBML.

An alter-ego execution model

Each principal maintains a private execution context, containing its policy, beliefs, observations and history of interactions. We assume that agents manipulate only their private context and cannot influence others' beliefs other than by performing communicative actions, which observed by other agents may trigger their reactions.

Each agent (including the environment, modeled explicitly) executes a program (or workflow) and is able to perform the following operations:

Communicative acts such as $Send_{A\rightarrow B}(Msg)$ and $Rcv_{B\leftarrow A}(Msg)$, or predicates capturing more semantics than $Send$, such as $Ask_{A\rightarrow B}(\alpha)$ ('Ag. A asks ag. B to perform action α'). An event (such as a communication) triggers inferences and knowledge updates. Predicates $Said_A(Msg)$ and $Received_B(Msg)$ reflect the knowledge updates of the agents as consequence of the communicative acts. Another coomunicative act, used to publish views about the state of the world, is $Decl_{X\rightarrow Y}(Formula)$, read as 'Ag. X declares to ag. Y that $Formula$ holds'.

Encryption of message M with key k is denoted $\{M\}_k$. k^- is the private key corresponding to the public key k, and $\{M\}_{k^-}$ is a signed message.

Delegation is the process of agent A giving to agent B the authority to act on A's behalf with respect to fulfilling some partial goal. $Delegates_{A\rightarrow B}(\alpha, Ctx, D)$ is read: 'A delegates B to execute α in the context Ctx, on behalf of A, and to use data item D'.

Authorization. $Authorize_{A\rightarrow B}(\alpha)$ corresponds to the action of A consenting to B to α. $Authorized_B(\alpha)$ means that B has been given consent for α.

Manipulating data. Agents are able to aggregate data and to split a piece of data into components: $Aggr/Split_A(D, [D_1, \ldots, D_n])$, where $D_1, \ldots, D_n, D \in Data$ are data items, and D is made of $D_1, \ldots, D_n$. The agent can also perform more intricate internal operations (such as evaluation of formulae, deriving consequences, observing/recording events), that change its initial set of beliefs.

By performing an action, an agent causes an event. Agents perceive events as the activation of some triggers (sensors) for those events. We use the Event-Condition-Action (E-C-A) paradigm to specify the behavior of an alter-ego, using three data structures, as suggested in [12]: **meta-knowledge** about agents, such as: agents A and B share certain information; ag. A trusts ag. B; ag. A was created by ag. B; ag. A belongs to the administrative domain D; **historical information** about past interactions: ag. A asked ag. B data D at time t_1; ag. A has performed action α at time t_2; a **stimulus-response table** that contains the program to be performed by the agent to achieve its objective, and that specifies the reaction to external events (such as requests from other agents). The stimulus-response table is specified declar-

atively as a knowledge program ([17]), with alternative execution branches, termed triggers, guarding events in different local states of the agent, of the form: $\langle R \rangle \quad Evt \ : \ S(PreC) \ : \ \alpha \ : \ PostC$, read as 'When regulation R is enforced, if event Evt occurs when agent is in state S, if condition $PreC$ is satisfied, then action α must be performed (which makes $PostC$ hold)'.

Starting from some initial state (containing a non-empty set of initial beliefs), and using its meta-knowledge, history and the stimulus-response table describing its behavior, an agent selects an action to perform and moves into its next state, on the basis of the messages it has received, the events it has observed and the action(s) it has performed.

We assume a discrete time parameter associated with each formula. All alter-egos of the same individual have the same clock, which allows them to refer to past events uniformly. A partial ordering of events is possible using temporal predicates: $Before/After(\alpha_1, \alpha_2)$, where α_1, α_2 are events.

The execution of an agent with identifier i is characterized by a trace, capturing its knowledge and actions performed, represented as a tuple: $\langle i, MK_i, Obj_i, Prog_i, Constr_i, SRT_i, Hist_i \rangle$, where $MK_i \in pow(Form(FBML))$ captures the initial knowledge of the agent and its subsequent updates; Obj_i includes the goal (objective) of the agent; $Prog_i \in pow(Form(FBML))$ is the agent program (capturing the agent's strategy to achieve its objective); $Constr_i$ captures the constraints on the agent behavior (such as regulations, preferences and the privacy policy PP, obligations, etc); SRT_i defines the reactive behavior of the agent to external events and passing from one state to another; $Hist_i$ is the history of local states.

The privacy policy PP of agent A is a sequence of declarations of the form:

$$Decl_{A \to all}(Perm_A([Info_1; Actor_1; Ctx_1; Action_1]; \ldots))$$
$$Decl_{A \to all}(Forbid_A([Info_i; Actor_i; Ctx_i; Action_i]; \ldots))$$
$$Decl_{A \to all}(Obl_A([Info_j; Actor_j; Ctx_j; Action_j]; \ldots))$$

The history of agent i is a set of local states $Hist_i = \{ls_i^t\}, t = \overline{0, n}$, where n is the current time point in agent's execution.

Each local state of agent i at time t is defined by a tuple of the form: $ls_i^t = \langle state\_id, OES_i^t, AS_i^t, US_i^t \rangle$; OES_i^t is the set of events observed by ag. i at time t, AS_i^t is the set of actions taken by i in response to the events, and US_i^t is the set of updates of the local knowledge at time t (changes to constraints, initial knowledge, stimulus-response table, etc).

An agent platform with n agents is represented by an n+1 set of tuples of the form $\{A_i\}$, $i = \overline{0, n}$, where $A_0 = Env$ is a special agent representing the environment. As its agents execute, a system traces out an execution history. The set of all possible execution traces is denoted Σ.

Describing properties of objects is done using the notation $Has_X^\tau(P)$, meaning 'At time τ, entity X has/ possesses P'. P is a property of the form 'attribute = value', or a formula describing a goal or a policy.

Our language allows reasoning about the following types of properties:

Provability. The "need-to-know" principle requires that personal data is

only disclosed to agents authorized to access it in a certain protocol execution framework. Agents must be able to prove that they should be authorized to perform certain actions. We write $Prove_A(\phi, \gamma, B)$ for "Agent A is able to prove proposition ϕ in the context γ to the audience B".

Essentially, provability derives from the possession of a valid piece of evidence. In computer security protocols, it is connected with verifying signatures and proofs; a certificate is usually regarded as a proof of identity, consisting of a signed message which says that some user U is associated with a piece of data D, usually his public key k, confirmed by an authority CA trusted by the potential verifier of the signature.

Jurisdiction is important in establishing the validity of a proof. $J_X(\varphi, Ctx)$ is to be read as 'X has jurisdiction over φ in context Ctx. Statements made by entities are fully trusted when their jurisdiction is recognized: $B_Y(Said_X(\varphi)) \wedge B_Y(J_X(\varphi)) \rightarrow B_Y(\varphi)$.
A TTP with jurisdiction over a MMAP is defined as a party whose all declarations are common knowledge: $Decl_{TTP \rightarrow MMAP}(\varphi) \rightarrow CK_{\forall A: A \in MMAP}(\varphi)$.

Evidence. We use the "count as" operator $\Longrightarrow_S$ (introduced by Jones & Sergot ([18], and used to express evidence rules in [19]) to express consequence in a given framework; "$A \Longrightarrow_S B$" reads "fact A counts, in the context where policy S applies, as fact B". We can express interpretation of formulae and conversions between data and formulae:
'If C_1 shows to C_2 a message M signed by C_3, this is interpreted by C_2 as evidence from C_1 that C_3 has issued M': $Sent_{C_1 \rightarrow C_2}(\{M\}_{k_{C_3}^-}) \Longrightarrow_{C_2} Prove_{C_1}(Said_{C_3}(M))$.

Ownership is understood by us as a special relation between a principal and a piece of data, as in [13]: a unique initial owner is defined for any object; the list of actual owners of the object changes during various transactions in which the object is involved; a subject who starts an application owns all the initial objects defined within the application unless they have predefined owners; a subject who creates a new object owns that object.

We make distinction between having full jurisdiction over a piece of data (having authorship), having control and partial jurisdiction over it (having ownership), having direct access to a piece of data (seeing), and comprehending a piece of knowledge (understanding). We denote these notions with *Authors*, *Owns*, *Sees* and *Understands*.

To illustrate the distinction, imagine an encrypted message $m = \{\{\{M\}_{k_O^-}\}_{k_D}\}_{k_F}$ originated at entity O, destined to entity D, sent to a temporary recipient F, who must forward it to D, but the message is also intercepted by an intruder I. Initially, $Auth_O(M), Owns_O(M), Sees_O(M)$ and supposedly $Understands_O(M)$, and upon receiving the message m, $Sees_F(m), Understands_F(m_1 = \{\{M\}_{k_O^-}\}_{k_D})$ (assuming that F is able to recognize and decrypt any message encrypted with its public key). Also, it holds that $Sees_I(m), Sees_I(m_1)$. After D receives m_1 from F
$Understands_D(M), Sees_D(m_1), Owns_D(m_1)$, but not $Understands_I(M)$nor $Understands_I(\{M\}_{k_O^-})$.

To detail this distinction:

$Auth_X(D)$: X is the originator of D; X has created or assembled data D, has full jurisdiction and all access rights to it.
$Owns_X(D)$: X controls and legally owns D; X could have received full or delegated access to D from a legal owner of D or from D's author (D's author must have been informed and must have expressed consent for that delegation). X cannot remove existing restrictions on D, but only add new restrictions, corresponding to his own privacy preferences.
$Sees_X(D)$: X controls data item D or is able to perceive D, but has not been legally delegated by a legitimate owner of it (or was delegated by an entity without jurisdiction over D). Example: a router can see all packets traversing it (their headers), but does not understand or own their content.
$Understands_X(D)$: X is able to access the meaning (the structure and the semantics) of D. This is useful for deciding whether a party who was in the possession of a message stating a norm is responsible for violating that norm or not, since the norm could be expressed in another language, using unknown terms, or encrypted with an unavailable key.

Note that the rights given by authorship of a data item in privacy or copyright protection scenarios are different from the authorship rights in some e-commerce protocols. The author of a piece of personal data maintains full jurisdiction over that data, no matter who owns it, and can even decide who may own it, while in some e-commerce protocols, the author of a sold electronic good loses the right to re-sell it, to destroy it, or to decide who may own it.

Violation. The security policy contains rules dealing with situations of violation. $Violates_A^{Ctx}(Policy)$ reads 'In context Ctx, A violates $Policy$'.

A privacy violation is regarded as a belief inconsistency of the owner of a piece of information: the agent believes that a party has information of or about it that, according to its privacy policy and the history of its interactions, should not be the case. Violation is expressed as:
$Decl_{X \rightarrow all}(Violation_Y([Formula; Policy/Context; Info; Cred]))$
standing for 'X declares that Y violates $Formula$, according to the terms expressed by $Context$ or $Policy$'. The claimed target of violation is $Info$, and the claimant specifies his credentials $Cred$ (e.g., the quality in which he makes the claim). $Context$ describes the circumstances in which the violation is possible, the entities affected by this violation, etc.

A privacy violation by X w.r.t. data contained in D occurs when
$Understands_X(D) \wedge Sees_X(D) \wedge \neg Owns_X(D) \wedge \neg Auth_X(D)$ (if X is forbidden access to D, $Forbidden_X(D)$). A privacy violation investigation involves proving that the privacy has been violated and establishing the responsibility.

Semantics

Agents in our model communicate by exchanging formatted envelopes of the form $Msg = \langle Type, Sender, Receiver, Data, Context \rangle$, where $Type$ is a label indicating the category of data transferred (whether the message is a request, reply, notification, whether it is encrypted and how, and what knowledge is required

to handle its content); $Sender, Receiver \in Ag$ are the originator agent and the destination of the envelope, $Data \in pow(Data) \times pow(Form(FBML))$ contains facts and data items transmitted, $Context \in pow(Form(FBML))$ is a set of formulae from the language FBML describing the circumstances in which $Data$ is to be evaluated. The set of all messages that can be exchanged is denoted $Mess$.

A model for our language FBML is a structure:
$\mathcal{M} = \langle \sigma, Ag, Ac, Data, Mess, Events, Epst, Acts, Comms, Poss \rangle$, where $\sigma \in \Sigma$ is a trace out of the possible set of execution traces Σ; Ag is a set of agents; Ac is the set of allowed actions; $Data$ is the set of personal data items; $Mess$ is the set of messages that can be exchanged in the system; $Events$ returns the set of (external) events occurred in a trace at a certain time; $Epst : \Sigma \times Ag \times \mathbf{N} \rightarrow pow(Form(FBML))$is a function returning the set of knowledge and beliefs hold by an agent in a particular trace; $Acts : \Sigma \times Ag \times \mathbf{N} \rightarrow Ac$returns the actions performed by the agents (knowledge updates and data manipulations); $Comms : \Sigma \times Ag \times Ag \times Data \times \mathbf{N} \rightarrow pow(Mess)$returns the set of messages exchanged by the agents of the system; $Poss : \Sigma \times Ag \times Data \times \mathbf{N} \rightarrow pow(Form(FBML))$returns the set of keys and personal data items possessed by agents in a trace.

Model $\mathcal{M} = \langle \sigma, Ag, Ac, Data, Mess, Events, Epst, Acts, Comms, Poss \rangle$ represents the execution of a multi-agent system $MMAP = \{\langle i, MK_i, Obj_i, Prog_i, Constr_i, SRT_i, Hist_i \rangle, i = \overline{0,n}\}$ iff: $\forall t \in \mathbf{N}, \forall i \in Ag : \quad Epst(\sigma, i, t) = MK_i^t \cup Constr_i^t \cup Obj_i^t \wedge Acts(\sigma, i, t) = AS(ls_i^t) \wedge Comms(\sigma, i, t) = \bigcup_{u=\overline{0,t}} \{M \in OES(ls_i^t) \mid M \in Mess\}$.

Inference Rules

We use two special predicates as substitutes for traditional **epistemic operators**: CK (common knowledge) describes norms and knowledge of policies and B (belief) is the standard epistemic operator for expressing formulae assumed to hold in the current context. If φ is a formula, A an agent, and G a group of agents then $B_A(\varphi)$ is read 'A believes that φ holds in the current context', and $CK_G(\varphi)$ means 'Fact φ is common knowledge in the group G'. The security policy of an agent platform is, for example, considered by us common knowledge for all its agents, and trusted third parties are considered disseminators of such knowledge.

We use the following inference rules:
P. All instances of propositional calculus tautologies.
Modus Ponens: From φ and $\varphi \rightarrow \psi$ any agent infers ψ.
Necessitation: From $\vdash \varphi$ infer $\vdash B_\star(\varphi)$, where φ is a theorem.
K-axiom applies for both B and CK operators.
Persistence of common knowledge. $CK_A^\tau(\varphi) \rightarrow CK_A^{\tau'}(\varphi)$ where $\tau, \tau' \in \mathbf{N}$ are time points such that $\tau \leq \tau'$.

To exemplify a few of the axioms used in our logical framework:

1. $B_A(J_B(\varphi)) \wedge B_A(Said_B(\varphi)) \rightarrow B_A(\varphi)$
2. $Sees_A(\varphi) \wedge Understands_A(\varphi) \rightarrow B_A(\varphi)$
3. $B_A(Has_B(X)) \rightarrow Aggr_A([B, X])$
4. $Has_A(\{M\}_{k_B^{-1}}) \wedge Prove_A(Auth_B(k_B)) \rightarrow Prove_A(Said_B(M))$
5. $Auth_A(D) \rightarrow Own_A(D)$
6. $Own_A(D) \rightarrow Has_A(D)$
7. $Has_A(D) \rightarrow Sees_A(D)$
8. $Said_A(M) \rightarrow Sees_A(M)$
9. $Has_A(M) \wedge Has_A(f) \rightarrow Has_A(f(M))$
10. $Rcv_{A \leftarrow X}(\{M\}_{k_B^{-1}}) \wedge B_A(Auth_B(k_B)) \rightarrow B_A(Said_B(M))$
11. $Rcv_{A \leftarrow X}(M) \rightarrow Has_A(M)$
12. $Rcv_{A \leftarrow X}([D_1, \ldots, D_n]) \rightarrow Rcv_{A \leftarrow X}(D_1) \wedge \ldots \wedge Rcv_{A \leftarrow X}(D_n)$
13. $B_A(Said_B(\varphi)) \wedge Understands_A(\varphi) \wedge Trust_A(B) \rightarrow B_A(\varphi)$
14. $Has_A(\{D\}_{k_A}) \wedge Has_A(k_A^{-1}) \rightarrow Sees_A(D)$
15. $Sees_A(h(D)) \wedge Has_A(h) \wedge Has_A(D) \rightarrow Understands_A(h(D))$

Privacy-compliance

Privacy-compliance is a safety property of an agent platform, which depends on several parameters: the privacy requirements of the actors, the class of traces (representing possible executions of a protocol) analyzed, the trust relations between actors, the reliability of the communication medium, etc. To enforce it, we propose purpose-binded delegation of ownership, which relies on privacy-compliant possession.

Definition 3.1 Privacy-compliant possession is defined as the possession of data together with a "compatible" purpose (an operation allowed for the given data), signed by the owner of that data.

Definition 3.2 Given a logical framework $\mathcal{L}^{FBML}$ for analysing epistemically the actors in combination with their programs and a particular trace of events, we term the protocol composed of those programs **(privacy-)compliant** with respect to the class of traces and the classes of actors analyzed, if no inference can be made in this logic that reveals a contradiction between the state of affairs and the stated privacy policy of a participant.

A protocol (i.e., the interactions dictated by the program of a multi-agent platform) is privacy-compliant iff what is communicated to the agents in its possible execution traces does not conflict with the privacy policies of the agents whose personal data is exchanged in the protocol. We state two necessary conditions under which a protocol can be privacy-compliant.

Definition 3.3 A protocol is *privacy-compliant* only if during its execution:

1. $\forall A \in Ag, \delta \in Data$ such that $Has_A(\delta) \rightarrow Auth_A(\delta) \vee Owns_A(\delta)$
2. $\forall A, B \in Ag$ s.t. $Auth_A(\delta) \wedge Owns_B(\delta) \rightarrow B_A(Owns_B(\delta))$

Expressing privacy requirements for privacy guardians

In a privacy-enforcing system S, one would like to express rules for enforcing privacy-compliance at particular moments in time such as:

$Send_{X \rightarrow Y}(D) \Longrightarrow_S Obliged_Y(Valid(Owns_X(D)))$

The security and privacy rules defining violations of privacy can be defined using formulae in FBML. A rule for preventing profile building by aggregating

employees' addresses with salary records may be represented as:
$Has_X([name_A, income_A]) \wedge Sees_X(addr_{C_1}) \Longrightarrow_A Violates_X(privacy_A)$

Possession of an aggregate composed of name and address of an agent X may be defined as a privacy violation: $Has_Y([name_X, addr_X]) \Longrightarrow_X Violates_Y(privacy_X)$

The privacy preferences of the individual, expressed as visibility rules, can be transformed into triggers in the stimulus-response table. For example, implementing purpose-binding checks can be expressed as:
$Rcv_{A \leftarrow B}(msg) : Has_{msg}(purpose = P) : Check(Valid(P))$

To reflect that the notifications sent by alter-ego B are credible to A: $Send_{B \rightarrow A}(\varphi) : B_A(J_B(\varphi)) : B_A(\varphi)$.

4 Example: An information flow scenario

We study the rightful ownership problem in the context of enforcing the purpose-binding principle in a multi-agent environment. We want to ensure that, during all stages of an electronic protocol executed on an agent platform the alter-egos are able to establish the legality of a possession or of a claim of ownership. If this simple privacy rule can be enforced, without all agents being assumed honest, the protocol using certificates of authorship for data items and purpose-binded transfer of ownership is privacy-compliant w.r.t. the privacy rule above, i.e. it guarantees that a piece of personal data cannot be used by an agent unauthorized to learn it, at the price of imposing additional controls (namely, for purpose-binding)

The information flow in figure 1 depicts a privacy violation scenario: information obtained by B from C (that is certified by a trusted third party CA) gets to flow from B to A without C being notified. In case this information contains personal details of C, this results in a violation of C's privacy, since C has not authorized and is not aware of the presence of his personal details at A's site. Example: a bookshop B selling user profiles (including credit card details) of its clients (C included) to company A without notifying their clients.

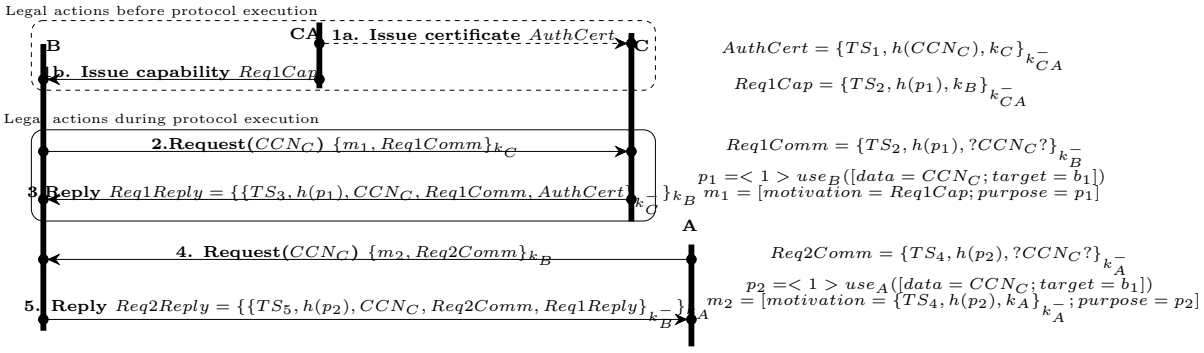

Fig. 1. Information flow scenario using cryptographic primitives and purpose-binding allowing detection of dishonest parties.

Before starting the analysis of the scenario in figure 1, we make a few

remarks. The analysis is based on some assumptions about the PKI:
1. PKI ensures confidentiality: a message encrypted with the public key of B can only be read by B: $Has_B(\{m\}_{k_B}) \wedge Auth_B(k_B) \Longrightarrow_{PKI} Sees_B(m); Has_A(\{m\}_{k_B}) \wedge \neg Auth_A(k_B) \wedge \neg Has_A(m) \Longrightarrow_{PKI} \neg Sees_A(m)$
2. A message signed with the private key of an entity A denotes (in PKI) possession and visibility of that message content by the signer A: $\{m\}_{k_A^-} \Rightarrow_{PKI} Has_A(m) \wedge Sees_A(m)$.

The scenario in figure 1 has 4 actors: agents A, B, and C, and the trusted third party CA. A and B are two bookshops; C is a client that orders a book at B. Steps 2 and 3 in this figure are part of a protocol for payment of books ordered, in which client C gives bookshop B his credit card number to pay for his order.

Before this protocol is started, the following operations take place (steps 1a and 1b in the dotted rectangle):
1. In step 1a, client C obtains a proof of authorship from CA, associating his credit card number (CCN_C) with his public key. CA issues the authorship certificate for CCN_C only if CCN_C is not already in his database of previously certified items. Being certified by CA, who is a TTP with jurisdiction of the agent platform in which C acts, CCN_C is a personal data item for which C can claim authorship.
2. In step 1b, in the phase of establishing the roles of the agents involved in the protocol, the TTP that guarantees the well development of electronic transactions, CA in our scenario, delivers capabilities to agents. These capabilities are certificates specifying the actions the alter-egos are authorized to perform on the personal data presented by the participants to the protocol, of the form $\{TS, h(p), k\}_{k_{CA}^-}$, where TS is a timestamp, $h(p)$ is a standard one-way function applied to the authorized action (purpose), k represents the public key of the agent authorized to perform the operation, and k_{CA}^- is the private key of CA. An alter-ego carries several such capabilities, which must be presented as motivation for his requests, whenever it asks for personal data. Function h hides the specified intended purpose for which ownership is transferred, to avoid privacy infringement during purpose-binding checks. A verifier can check purpose-binding only for purposes he knows of.

Posession of valid capabilities provides non-repudiable evidence that the requester is entitled to ask data, according to a protocol guaranteed by a TTP. Possession of valid proofs of ownership guarantees the right to possess data for a restricted purpose. Any request for personal data must be signed by the requester, to ensure non-repudiation: the queried agent is later able to prove that the request for disclosure came from an agent authorized to ask for it.

We analyze the situation depicted in figure 1 using our language for reasoning about privacy. In this scenario, B is dishonest and leaks to A the legally obtained credit card number of C, breaking the purpose-binding principle (commitment to use CCN_C solely for the payment for a book ordered). (p_1, m_1) and (p_2, m_2) in figure 1 represent the purpose-motivation pairs speci-

fying the right of B and C, respectively, to use CCN_C for payment of a book ordered by C. Note that m_1 contains a valid capability for purpose p_1, since it has been issued by a TTP, while m_2 does not contain a valid authorization, since it is signed by A, and not by a TTP.

Upon receiving the authorship certificate $AuthCert = \{TS_1, h(CCN_C), k_C\}_{k_{CA}^-}$ in step 1a, C is able to prove (based on the definition of jurisdiction) to any agent X that trusts CA that he is a legal author of CCN_C: $Has_C(AuthCert) \rightarrow Prove_C(Auth_C(CCN_C), \star, \forall X : Trust_X(CA))$.
Actually, any agent Y who understands PKI and purpose-binded delegation of ownership (PPDO) and possesses this certificate is able to prove to anyone who trusts CA that C is legal owner of a piece of data that can generate $h(CCN_C)$:
$Has_Y(AuthCert) \wedge Understands_Y([PKI, PPDO]) \rightarrow Prove_Y(Auth_C(CCN_C), \star, \forall X : Trust_X(CA))$.

The pairs of steps 2, 3, and 4, 5 in figure 1 correspond to purpose-binding delegations of ownership for the data item CCN_C. Steps 2 and 3 are legal operations, part of a protocol for an online payment, while steps 4 and 5 represent actions performed outside of the legal framework of any protocol.

After performing step 2, C is able to prove that the request was authorized as part of a protocol guaranteed by CA. If C trusts CA, C trusts B to be a legitimate requester. After a lookup in his privacy policy, in which no rule $Forbid_C([Info = CCN_C; Actor = B; \ldots])$is detected, C performs step 3, issuing a purpose-binded delegation of ownership to B.

After decrypting the message received in step 3,
$Sees_B(Req1Reply) \wedge B_B(Auth_C(k_C)) \Longrightarrow_B B_B(Said_C(M)); M = [TS_3, h(p_1), CCN_C, Req1Comm, AuthCert]$
$Understands_B(M) \rightarrow Sees_B(CCN_C) \wedge B_B(Authorize_{C \rightarrow B}([CCN_C, h(p_1)]))$. That is, B understands that he has received CCN_C and that C authorizes an aggregate $[p_1, CCN_C]$ (which describes the right of B to pay a book b_1 using CCN_C), and this authorization comes as a reply to a non-repudiable request ($Req1Comm$) of B. The second part of M contains the authorship certificate of C over CCN_C. Since B is able to prove that he was authorized by C to use CCN_C for purpose p_1, and can prove that C is the legal author of CCN_C, B is now able to prove that he is a legal owner of CCN_C (with respect to p_1):
$Prove_B(Authorize_{C \rightarrow B}([CCN_C, h(p_1)]) \wedge Auth_C(CCN_C), \star, \forall X :$
$Trust_X(CA)) \Rightarrow_{PKI,PPDO} Prove_B(Owns_B(CCN_C))$

In step 4, A asks from B CCN_C to pay for the same book C ordered to B. A has been issued no capability by CA to support his request, but he self-signs his request.

If B were honest, he would discard A's request, because he doesn't have the right to disclose further CCN_C to unauthorized parties. But B is dishonest, and delegates ownership of CCN_C to A, in step 5, authorizing A to use CCN_C for purpose p_2, different than the one for which he has been authorized to, by C. B forges a 'legal' delegation himself, behaving as he is entitled to delegate ownership of CCN_C to A. Therefore, B commits abuse (misappropriation of CCN_C), violating C's privacy (breaking the commitment to purpose-binding).

C cannot detect this misappropriation by B, unless an alter-ego of C accompanies the message and notifies him of this privacy exception.

Anyway, A cannot use the missapropriated CCN_C, because when he is to pay at the bank using CCN_C, he has to present the purpose for which he is paying ($h(p_2)$) (in order to show its privacy-compliant possession of data), and the chain of ownership certificates presented cannot lead back to a certified author of CCN_C. Even if B had replayed the message received in step 3 from C, A would have got an unusable authorization (which authorizes B, but not C, to use CCN_C for one payment).

A privacy exception is triggered by the bank, as the request from A to use CCN_C is not privacy-compliant. Upon an investigation, the parties across the certification chain are challenged to reveal their proofs of ownership. B cannot prove he is entitled to delegate ownership of CCN_C for $h(p_2)$, therefore he can be accused of misappropriation. A is also responsible for not notifying a disclosure that is not privacy-compliant to a law-enforcement agency.

If A were an authorized bookshop to which B is authorized to disclose CCN_C (in case that book b_1 is no longer in B's stock, but it is in A's stock, when disclosing CCN_C the purposes accepted in step 3 would have been $\{h(p_1), h(p_2)\}$ instead of $h(p_1)$, meaning that C explicitly authorizes other bookshops to take over the order and deliver the book. In that case, the bank could certify that A is a legitimate user of CCN_C.

The scenario presented, which relies on PKI, and in which purpose-binded delegation of ownership and privacy-compliant possession are mandatory, allows detection of dishonest parties (that commit privacy violations), and assignment of blame, under the assumption that some principals are corrupt, but authorities (such as the bank) and the honest principals commit to performing privacy-compliance checks.

Implementing it using alter-egos would guarantee enforcement of purpose-binding checks and notification of the individual and would allow more subtle constraints between the different certificates issued by an individual (such as, that either bookshop B or bookshop A can use CCN_C to pay for book b_1, but if one of them has used it once, the other can no longer use it).

5 Concluding remarks

The participants to Cyberspace protocols need informedness about the other participants, and must assume additional commitments to purpose-binding checks in order to reduce the risk of privacy violations. We discuss a possible representation of privacy rules and some primitives for modeling the agents that enforce these rules, that would enable reasoning about delegation, ownership, and the ability to provide supporting evidence.

A protocol which transfers or manipulates personal data can be analyzed and characterized from the privacy-perspective, as compliance with the privacy principles (such as purpose-binding), as illustrated in section 4. This

would enable the design of a privacy assessment mechanism, in which agents check the compatibilities of privacy policies expressed by individuals with the privacy practices and security restrictions expressed by the service-providers, or even negotiate a compromise solution, such that individual's privacy interests can be protected without violating relevant security requirements of the environment.

References

[1] Bieber, P. and F. Cuppens, *Computer security policies and deontic logic*, in: *Proceedings of the First International Workshop on Deontic Logic in Computer Science, Amsterdam, The Netherlands*, 1991.

[2] Biskup, J., *For Unknown Secrecies Refusal is Better than Lying*, Data & Knowledge Engineering (2000), pp.1-23.

[3] Cuppens, F., *Roles and deontic logic*, in: *Proceedings of the Second International Workshop on Deontic Logic in Computer Science, Oslo, Norway*, 1994.

[4] Cuppens, F. and C. Saurel, *Specifying a security policy: A case study*, in: *Proceedings of the computer security foundations workshop, Kenmare, Co. Kerry, Ireland*, 1996.

[5] Cuppens, P. B. F., *Expression of confidentiality policies with deontic logic*, in: J. M. R. W. (Eds), editor, *Deontic Logic in Computer Science: Normative System Specification*, Wiley, 1993 .

[6] W. de Jonge; Reind van de Riet, G. L. S., *Answering queries without revealing secrets*, ACM Transactions on Database Systems (March 1983), pp.41-59.

[7] Fred B. Schneider, *Enforceable Security Policies* (1999).

[8] Harry Hochheiser, *Principles for privacy protection software*, in: *Proceedings of the 14th Conference on Computers, Freedom and Privacy, CFP2000, Toronto, Canada*, April 2000.

[9] J. Glasgow, G. M. and P. Panangaden, *A logic for reasoning about security*, in: *Proceedings of Computer Security Foundations Workshop, Franconia* (1990), pp. 2–13.

[10] Mark S. Ackerman; Lorrie Faith Cranor; Joseph Reagle, *Privacy in e-commerce: Examining user scenarios and privacy preferences*, in: *Proceedings of the ACM Conference on Electronic Commerce, 1999*, 1999, pp. 1–8.

[11] Paul F Syverson; Stuart G Stubblebine, *Group principals and the formalization of anonymity*, in: J. W. J. W. J. Davies, editor, *Formal Methods, vol. I*, LNCS vol. 1708, Springer-Verlag, Berlin, September 1999 pp. 814–833.

[12] Piero A. Bonatti; Sarit Kraus; Jose Salinas; V. S. Subrahmanian, *Data-Security in Heterogeneous Agent Systems* (1999).

[13] Prasun Dewan; Honghai Shen, *Flexible Meta Access-Control for Collaborative Applications* (2000).

[14] Radu Serban; Reind van de Riet, *Trusted environments for privacy protection in online transactions*, in: C. C. Y.-H. T. R. F. B. S. Firozabadi, editor, *Proceedings of the 3rd workshop on Deception Fraud and Trust in Agent Societies, AA-1999, Seattle, U.S.A.*, May 1999, pp. 107–116.

[15] Rodolphe Ortalo, *Using Deontic Logic for Security Policy Specification*, Technical Report LAAS-96380, LAAS-CNRS Toulouse, France (1996).

[16] Rodolphe Ortalo, *Using Role-based Abstractions for Security Policy Specification with Deontic Logic*, Technical Report LAAS-97216, LAAS-CNRS Toulouse, France (1998).

[17] Ronald Fagin; Joseph Y. Halpern; Yoram Moses; Moshe Y. Vardi, *Knowledge-Based Programs*, Distributed Computing (1997), pp.199-225.

[18] Sergot, A. J. M., *A formal characterisation of institutionalised power*, Journal of the IGPL (1996), pp.429-445.

[19] Thoen, Y.-H. T. W., *Towards a generic model of trust for electronic commerce*, in: C. C. Y.-H. T. R. F. B. S. Firozabadi, editor, *Proceedings of the 3rd workshop on Deception Fraud and Trust in Agent Societies, AA-1999, Seattle, U.S.A.*, May 1999, pp. 117–127.

[20] Uwe G. Wilhelm; Sebastian M. Staamann, *A Pessimistic Approach to Trust in Mobile Agent Platforms*, IEEE Internet Computing (2000), pp.40-48.

[21] Wilhelm, U. G., "A Technical Approach to Privacy based on Mobile Agents Protected by Tamper-resistant Hardware (PhD Thesis)," Ecole Polytechnique Federale de Lausanne, Switzerland, 1999.

[22] Wooldridge, M., *Temporal belief logics for modelling distributed artificial intelligence systems*, in: M. P. O'Hare and N. R. Jennings, editors, *Foundations of DAI*, Wiley Interscience, 1995 .