Contextual Integrity and Privacy Enforcing Norms for Virtual Communities *

Yann Krupa, Laurent Vercouter

Laboratory for Information Science and Technology (LIST),
ISCOD team
École des Mines de Saint-Étienne
Saint-Étienne, France
{krupa,vercouter}@emse.fr

Abstract. Contextual Integrity has been proposed to define privacy in an unusual way. Most approaches take into account a sensitivity level or a "privacy circle": the information is said to be private or public and to be constrained to a given group of agents, *e.g.* "my friends". In the opposite, Contextual Integrity states that any information transmission can trigger a privacy violation depending on the context of the transmission. We use this theory to describe a framework that one can use in an open and decentralised virtual community to handle privacy in a socially enforced way. This paper describes a framework, in which we can formally describe privacy constraints, that are used to detect privacy violations according to the Contextual Integrity theory. This framework is part of an ongoing work aiming at applying social control to agents that handle the information, so that malicious agents are excluded from the system.

1 Introduction

Most of the works on privacy focus on security as a means of preserving privacy, either by using a central authority that controls the information access[2, 3], cryptography[11], or by using trusted computing techniques[8, 6]. Some other views[9, 4] aim at designing some preferences that users can attach to the data they "own" without taking into account the possibility of deception by other agents.

While a central authority may be a good solution for a range of applications, it is not possible when working in a decentralized and open system with autonomous agents. Therefore, solutions like Purpose Based Access Control[3] cannot be applied.

One of the problems with Digital Right Management and Trusted Computing measures in general, is that they are very constraining. They impose the use of heavy infrastructure or limit the possibilities of information exchange. These constraints, if they are unacceptable for the users, lead them to interact outside the system that is provided, making every implemented security feature inefficient.

Social regulation is another approach where it is physically possible that violations occur in the system. However, users are observed by the society (usually

*position paper

their neighbours) that can spot them and socially exclude them by ostracism if they commit violations.

So far, very few works consider privacy preserving under the social angle. Yet, it is a prominent problem in applications such as social networks, virtual communities and multi-agent systems, where a social framework will cope naturally with all the social components already working in these systems like trust, reputation, roles for exemple.

Our work tackles the problem of privacy by using social control in decentralized and open multiagent systems. It is based on Nissenbaum's "Contextual Integrity" theory[7] which defines privacy in a socially relevant way. Therefore it is possible to assess privacy violations from the agent point of view, and apply a social control relying on available social mechanisms, such as the use of trust management techniques, to prevent further violations. Privacy violations will be reduced without requiring a central authority or invasive security measures.

Contextual Integrity states that any information, as inoffensive as it could seem, can potentially harm a set of agents. It means that Contextual Integrity does not make assessment towards the degree of sensitivity of a given information. All information is regarded as evenly sensitive/insensitive. We call the set of agents that can be harmed by an information, its *Targets*. We say that an agent is *harmed* by an information if it makes the agent lose any kind of resource *e.g.* time, reputation, acquaintances, role. An agent sending information is called *Propagator* and the agent receiving the information is called *Receiver*. During the different moments of the process and depending on the information, those attributions may change from one agent to another.

The goal of our work is to provide means to a propagator to use logical reasoning and trust mechanisms to make assessments about a further transmission: "will the transmission of information i to agent z be a violation of contextual integrity?". A receiver should also be able to do the same process when receiving an information: "was the reception of information i from agent a a violation?". If a violation is detected, social sanctions are thrown against the violating agents.

This paper describes this ongoing work, it proposes a framework in which we can formally describe privacy constraints according to the Contextual Integrity theory and norms in order to enforce these constraints. This framework is then used to detect the occurrence of privacy violations. The sequel of this article is organized as follows. Section 2 presents Nissenbaum's Contextual Integrity theory and how we interpret it to build appropriateness laws. The characteristics of the application that we consider, virtual communities, is described in section 3 and it is formally described in order to be able to detect automatically privacy violations. Then, a set of privacy enforcing norms are defined in order to give a roadmap for agent's behavior in section 4. Finally, section 5 shows how these social mechanisms are used to prevent and punish privacy violations on a sample application, and we conclude the paper in section 6.

2 Contextual Integrity

In this section we present the theory of Contextual Integrity[7] and introduce the concept of appropriateness extracted from this theory.

2.1 Original Works

In some approaches[8] privacy is viewed as binary (either the information is private or not). Other models consider different levels of privacy[2] or circles[4], whereas contextual integrity focuses on the appropriateness of a transmission, or use of information. Every information is potentially sensitive.

In order to have a complete description of the foundations of the theory, the reader should refer to the original article[7]. Here we will only focus our work on the concept of "violation". Nissenbaum says that "whether a particular action is determined a violation of privacy is a function of :

1. the nature of the situation/context
2. nature of the information with regard to the context
3. roles of agents receiving the information
4. relation of agents to information subject
5. terms of dissemination defined by the subject"

Those ideas are way too vague to be used as-is in a software application, we define hereafter a more precise definition of the concepts.

2.2 Appropriateness

We use the term appropriateness to define the set of laws that makes a transmission inappropriate (*i.e.* will trigger a violation of privacy) if one of these laws is violated. The term "Appropriateness" is inspired by[1].

We use the term "target" instead of Nissenbaum's term "subject". A subject is directly related to the information, while a target may not even appear in the information. For example, if the information is a picture of Mr X's dog doing bad things on X's neighbour's grass, the subject is the dog but the target, the one that can be harmed by the disclosure of the picture, is X. Therefore, we think that considering the target instead of the subject is more versatile.

We can then define a flow as appropriate if all of the following conditions hold, and inappropriate if one of the conditions does not hold (numbers in parenthesis refers to the corresponding statement in Nissenbaum's definition above):

1. Transmission context must correspond to the information nature (1+2),
2. Agent must have a role within the transmission context (3),
3. Agents must not have incompatible relations with target[1] (4),
4. The target's preferences must be respected (5)

If a flow is inappropriate then there is a privacy violation. Here we can see the point of this approach: an information is not "public" or "private", every information can trigger a privacy violation if it is used inappropriately.

Thereafter, we illustrate the 4 statements of appropriateness with examples:

[1] This item is a work in progress and is not taken into account in the following parts.

1. In the large sense, the context of a transmission can be seen as the environment where and when the transmission takes place. In our framework, for simplification means, we will say that the context of a transmission is declared by the propagator. A context corresponds to the information if it reflects the nature of the information, *e.g.*: personal health information corresponds to medical context.
2. Agents participating in the transaction should have a role associated to this context[1]. For example, a medical doctor has a role belonging to the medical context.
3. Sometimes, it is not sufficient that the agent has some roles belonging to the context of the transmission. Because some relations between the target of the information and the agent receiving the information may be inappropriate. For example, consider the case of an agent A who has an illness, and an agent B who is both a medical doctor and A's boss. It may be inappropriate for B to know A's disease because those agents are having an "out of context" relationship (hierarchical relationship).
4. If one of the targets of the information specifies preferences regarding the propagation of the information, it is inappropriate to violate those preferences.

As appropriateness has been defined it is now necessary to define the kind of application we consider, information transmission in virtual communities. Afterwards, we propose a formalism of appropriateness to be used in this kind of application.

3 Framework

This section presents the application domain and all the components needed to handle contextual integrity as defined in the previous section, as well as a formalism of appropriateness.

3.1 Privacy Preservation in Virtual Communities

In several types of virtual communities, such as social networks or virtual enterprises[2], users communicate and share information using software systems that support the community. These applications raise a difficult problem of privacy preservation. On the one hand, it is the main goal of these communities to enable communication so that users can easily send information to their contacts. On the other hand, as it is stated by the contextual integrity theory, each piece of communicated information may result in a privacy violation. Indeed, if we consider the case of a virtual enterprise, the community includes users with different hierarchical roles, belonging to different services but also different enterprises. It is obvious that all information should not be sent to other users without analysing the nature of information and of the concerned users. The same case

[2] A virtual enterprise is a temporary collaborative network of enterprises made in order to share resources and competences.

occurs in professional or personal social networks in which users' contacts can be her colleagues, siblings, friends.

The goal of our work is to specify a software assistant agent that is able to help a user to preserve privacy in a virtual community. The assistance is both to preserve the user's privacy by providing advices when an information is communicated (should he send this information or not to a given contact?), and to preserve the other users' privacy by detecting when a privacy violation occurred and should be punished. This paper describes the first steps of this ongoing work by defining a language to express privacy constraints and means to detect privacy violations.

The virtual community that we consider has the following characteristics. It works as a peer-to-peer network, meaning that information is exchanged by a communication between one sender and one receiver. Moreover, it is a decentralized and open system. It is thus impossible to define a centralized control that relies on a global and complete perception of communications. We have chosen a system with these features to be as general as possible. By proposing a local assistance to users, the assistant agent can be used both in centralized and decentralized systems and it does not constrain the system scalability. The choice of peer-to-peer communication is also general enough to be able to represent other kinds of communications. For instance, if we want to consider a social network in which information is exchanged by publishing it on a page or a "wall" readable by the user's contacts, it can be represented by several one-to-one communications.

In order to be able to define privacy preservation according to contextual integrity, we need to introduce two concepts in the virtual community: **context** and **role**. The context describes the situation in which an information is exchanged. Examples of context are "Dave's work", "John's family", "health". Roles are defined within a context and attached to users. Examples of roles in the three contexts mentioned above are respectively "Dave's boss", "John's father", "medical doctor". There can multiple roles per context. In this paper, we assume that users' roles and their corresponding contexts are provided by organisational entities that act as repositories. These entities are able to return the role associated to a specific user and the context associated with a specific role. For this purpose, it is possible to use organisational multiagent infrastructures[5].

These concepts are useful to be able to express rather fine rules for Contextual Integrity. We use them in the next subsections to allow the assistant agent to reason on privacy violations.

3.2 Message Structure

Users exchange **information** encapsulated in a **message**. Information is raw data. We don't make assessment about the structure of the information and leave it free. A message encapsulates information plus meta-information described below.

First, from a given information, can be computed a unique reference that allows to refer unambiguously to the information without carrying itself the information (Hash algorithms like Message Digest[10] can be used).

Then, the message adds the following meta-information:

- Context Tags: tags referring to the context of the information
- Target Tags: tags referring to the targets of the information
- Privacy Policies: policies expressing preferences regarding further distribution of information
- Transmission Chain: a chain of transmissions that allows to keep track of the message path in the system

Each of these components may be digitally signed by agents that wish to support the meta-information accountability. When signing a meta-information an agent engages his responsibility. The semantics that relies behind the signature is a certification: *i.e.* the agent that signs the context tag "medical context" certifies that the information is about medical context. Therefore, it is very important that a meta-information, even if it can be detached from the information (which is possible), cannot be reattached to another information. We prevent that from happening by including the information hash before signing. Signatures are formed by a name and a signature (RSA signature for example[11]). The transmission chain allows to keep track of the message path among the agents. Every agent is required to sign the chain before propagating a message, an agent adds his signature including his own name and the name of the receiver of the message.

3.3 Primitives

To allow the agent to recover data regarding the concepts described earlier, like the meta-information or the roles of agents, we need to provide the agents a set of logical primitives. These primitives can then be used to express constraints about transmission of information.

1. Primitives based on meta-information:
   - `information(M,I).` Means that I is the information[3] encapsulated in message M.
   - `contexttag(C,A,M).` Means that C is the context tag for message M signed by agent A.
   - `targettag(T,A,M).` T is the target tag for message M, signed by A.
   - `policy(P,A,I).` There is a policy P signed by agent A for information I.
2. Primitives based on transmission roles:
   - `receiver(X,M).` Agent X is receiving the message M.
   - `propagator(X,M).` Agent X is sending the message M.
3. Primitives based on agent beliefs:
   - `target(X,I).` The agent believes that agent X is targeted by the information I.

[3] The primitives are referring to an information I or a message M. This is because some primitives will be specific to a given message M, and some others will be common to all messages containing the same piece of information I.

- `policyvalid(P,I).` The agent believes that the preferences expressed by policy P are respected for the information I.
- `context(C,I).` Means that the agent believes that C is the context of information I
- `role(A,R).` The agent believes that Agent A has the role R.
- `rolecontext(R,C).` The agent believes that role R belongs to context C (role "surgeon" belongs to Medical context).
- `link(X,Y).` The agent believes that agent X is capable of communicating with Y.

Now, based on this primitives, we are able to express preferences or norms.

3.4 Appropriateness Laws

Our goal is to obtain some simple laws that agents can rely on to be able to decide if a given transmission of information should be seen as a violation or not.
These appropriateness laws are thereafter abbreviated as A-laws.
This is the definition of the A-laws we propose in Prolog-like code:

- Context declared by the propagator must be equal to the information context (Fig. 1).
- Receiver must have a role within the transmission context (Fig. 2).

```
fitcontext(C,M):-
   information(M,I),
   propagator(P,M),
   context(C,I),
   contexttag(C,P,M).
```

Fig. 1. fitcontext

```
fitrole(C,M):-
   receiver(Rc,M),
   role(Rc,R),
   rolecontext(R,C).
```

Fig. 2. fitrole

- The target's preferences must be respected:
  - In the case there is no[4] policy defined by a target then `fitpolicy(M)` holds (Fig. 3).
  - If there is a policy defined by the target, the agent must respect it (Fig. 4).

Therefore, a transmission is defined as appropriate for a message M if the following formula holds:

```
appropriate(M):-
   fitcontext(C,M),
   fitrole(C,M),
   fitpolicy(M).
```

If the definition above does not hold, then we can say that the transmission is inappropriate, there is a violation of the contextual integrity.

[4] `\+` is the negation-as-failure in Prolog.

```
fitpolicy(M):-
   information(M,I),
   \+ (
      policy(P,T,I),
      target(T,I)
      ).
```

Fig. 3. fitpolicy (when no policy is defined)

```
fitpolicy(M):-
   information(M,I),
   policy(P,T,I),
   target(T,I),
   policyvalid(P,I).
```

Fig. 4. fitpolicy (when a policy exists)

3.5 Policies

The A-laws define what is appropriate or not in a general point of view, but targets can define policies (preferences) in order to constrain the information. These preferences are defined for a given information by a given agent who signs the policy. In the system, it is not possible to insure that a policy cannot be detached from the information it is referring to, *i.e.* an agent may erase the policy at some point. But it is ot possible to reattach a policy to another information, because the policy is signed, and contains a pointer to the information it refers to.

A **policy** is composed by several **statements**. A **statement** is composed by several **primitives** from the ones described in section 3.3 and by a type of statement that can be:

- `forbidden(I):-`
  Declares a situation that should not occur within a transmission of information I.
- `mandatory(I):-`
  Declares a situation that has to occur within a transmission of information I.

A given policy is fulfilled if none of its forbidden statements holds (if one holds, then it is unfulfilled) and one of its mandatory statements holds[1][5].

An example of policy for a given information identified by 'info99' is given below. It is composed by two forbidden statements (do not send data to an agent who has a common contact with the target AND don't send data to the target) and one empty mandatory statement.

```
forbidden(info99):-
   information(M,info99),
   receiver(X,M),
   target(T,info99),
   link(X,Z),
   link(Z,T).
```

```
forbidden(info99):-
   information(M,info99),
   receiver(X,M),
   target(X).

mandatory(info99).
```

[5] A statement is composed by a conjunction of primitives, therefore the disjunction is expressed by defining multiple statements of the same kind. This is why only one mandatory statement is required to validate the policy and one forbidden to invalidate it.

In order to test the primitive `policyvalid(P,I)`, an agent adds to his memory all the statements contained in policy P (we suppose here that we have a primitive `addpolicy(P)` to do just that):

```
policyvalid(P,I):-
   addpolicy(P),
   \+ forbidden(I),
   mandatory(I).
```

4 Privacy Enforcing Norms

As shown in the previous sections, we need the agents to check the transmissions, to be able to see if there are violations and punish the responsibles. This section propose a set of norms that defines what should be the behavior of a compliant agent in the system. Then it describes the punition mechanisms and finally discusses the inherent problems regarding the subjectivity of beliefs.

4.1 Definition

The basic component of the system is the set of A-laws, that express Contextual Integrity violation. But the keystone of the system are the Privacy Enforcing Norms (PENs), defined in this section, that instruct the agents to respect the A-laws and punish those who do not.
The PENs are the following :

1. Respect the Appropriateness laws
2. Sign the transmission chain before sending
3. Do not send information to untrusted agents
4. Delete information from violating or untrusted agents
5. Punish agents violating these norms (this one included)

The first norm (PEN 1) that we propose is meant to protect the A-laws from being violated : “Respect the Appropriateness laws”.

From our point of view, every agent must take responsibility when doing a transmission. Thus we define a norm stating that every agent has to sign the transmission chain (in order to backtrack the potential violation to its source). We also consider that sending information to an agent while knowing that he will commit a violation, is a violation itself. Two new norms are then defined : “Sign the transmission chain before sending (PEN 2) ; Do not send information to untrusted agents (PEN 3).” The PEN 3 also implements the social punishment, because agents will stop communicating with these untrusted agents.

The fourth norm aims at minimizing the violations by deleting information received from unreliable agents (PEN 4).

Norms that the agents should respect have been defined, but we want to be sure that the agents in the system will punish those who do not respect the norms, henceforth punishing those that do not punish agents not respecting the norms. This last norm (PEN 5) insures consistency of the PENs, because an

agent that decides to violate a norm will be punished, others will stop trusting him and eventually he will become socially isolated.

Therefore norms are not enforced by the system but by the agents themselves and agents refusing to enforce the norms will be punished by other agents. For now, the punishment is implemented as a social punishment: an agent witnessing a violation has to send a message to all of its contacts stating the details of this violation. The following section gives more details about this punishment mechanism.

4.2 Punishment

When an agent detects a violation of the PENs, PEN 5 states that he has to send a punishment message. This message is meant to describe the violation so that other agents can punish the culprit. The message has the same structure than all the messages in the system: information and meta-information. Here the information part contains:

- The meta-information of the original message source of the violation
- A description of the violation usign the primitives of section 3.3

Sending the meta-information of the original message is useful to provide evidence to other agents that may not believe that there was a violation. The advantage of sending only the meta-information is that the agent will not transmit the information itself (which could in turn trigger a violation and so on).

The violation is described using the primitives, and the PEN that has been violated. For instance, if the PEN 3 has been violated by Bob, the following primitives will be sent:

```
pen3violation(Bob,mess45),
receiver(John,mess45),
propagator(Bob,mess45),
untrustworthy(John).
```

These primitives will be handled and verified by the receiving agent. If the agent agrees with every primitive in the argument, then he can propagate the punishment message and punish the culprit by revising its trust. There are situations where the agents may not have the same beliefs, *e.g.* John may or may not be trustworthy depending on the agent making the assesment.

4.3 Discussions on Subjectivity

Some of the PENs are very subjective, because they are based on beliefs. Therefore 2 given agents in the system may not have the same belief and interpret the norms differently. For instance, two agents, A and B may have different beliefs regarding agent X trustworthiness: A trusts X but B does not. Now A sends a message to X who in turn sends the message to B. In the transmission chain, B is able to see that the transmission occured between A and X, which violates norm 3. Going back from A point of view, it would not be fair to be punished for this transmission as X seems trustworthy for him.

B witnessed a violation so he has to send a punishment message. The punishment message has to argue about the punition. More than just saying "A does not respect the norms", B makes a message stating that "A violated the third norm because B believes that X is untrustworthy and A sent a message to X". The agent receiving this violation message is going to check these statements and if he agrees, he can revise his trust level towards A.

Along with the violation description, the punishment message contains the meta-information of the original message. This allows other agents to check the PENs and violation description. For instance, it will allow agents to check that A did sent the information to X by looking at the transmission chain contained in the meta-information.

4.4 Usage

This section describes how the agents are meant to protect privacy using the tools provided in the previous sections. As it is said in the introduction, our goal here is to handle privacy from the agent perspective to minimise the number of violations in the whole system.

There will be two main situations:

- Receiving: When the agent receives an information: "Does the agent that sent me this information made a violation by sending it to me?"
- Propagating: When the agent is about to send information: "Am I going to make a violation if I send the information to this agent?"

Trust In the framework presented in this article, agents may perceive things differently. If we take a closer look at the primitive `context(C,I)` described earlier, for instance, it is stated that it means that the "agent **believes** that C is the context of information I". Therefore, some agent X may believe for a given information that the context is O, and another agent Y may believe that the context is P. This situation can happen because the agents are autonomous and have beliefs that can be different from one to another. As they have different beliefs, some agent may think that a given transmission is inappropriate, and another may think that it is not. Because of this uncertainty, when an agent detects a violation, he is not able to be sure that the other agent made the violation on purpose, therefore it will be unfair to kick him directly from the system. This is where trust comes in, this kind of "soft security" is able to cope with detection errors while still being able to exclude the ones that make violations. Trust is one of the main components to decide who is reliable or not for handling our information. If someone is untrustworthy, we are not willing to send him any piece of information.

The trust management component of agents is not yet implemented and is being defined in our current ongoing work. We will probabaly use an adaptation of existing computational trust models for multi-agent systems such as LIAR[13] or Repage[12].

Receiving When the agent is receiving a message, he has to check if the transmission that just occured is a PEN violation or not. First, the agent has to check the A-laws to see if the transmission is appropriate (PEN 1), as described in section 2.2. To do that, the agent will have to infer multiple things, for example: who is the target of the message? what is the context of the message? This is possible either by using personal knowledge, by using the context tags and target tags or by analysing the information directly. As the context tags (and target tags) are signed, it is possible to trust the agent that signed the given tag, to come to believe that this context tag corresponds to the context of the information.

If the agent detects a PEN violation, he sends a "punishment message" to other agents.

Finally, the agent readjusts the trust level he has towards the propagator that just made the violation.

Propagating This second situation happens when the agent is about to send information. Before sending, it is necessary to attach to the information all possible meta-information:

- If the agent can identify the target of the information (by using knowledge or information analysis), he adds a target tag for target Z that he signs. This states that the agents confirms that the target of the information is Z.
- If the agent is able to determine the context of the information (by using knowledge or information analysis), he adds and signs a context tag.
- If the agent is the target, he can specify some restrictions regarding further dissemination of the information, in this case, he adds a policy that he signs.
- The agent also signs the transmission chain to insure PEN 2.

Then, the agent should make all PEN assessments towards the receiver:

- Does the agent violates the A-laws (PEN 1) by sending the information to the receiver? An agent never violates A-laws, except if he is malevolent or ignorant, which in both cases, will be punished by other agents.
- Does the agent trust the receiver? (PEN 3) If he is untrustworthy, it means that he has probably made some privacy violations in the past. As the agent aims at protecting the information he holds, he only sends to the ones he trusts.
- And so on with the other PENs.

At the end, the agents send information from one to another, checking before sending and after receiving if some violation has occurred. When violations are detected, agents send "punishment messages" to their contacts, so that others become aware of the violation that occurred. Eventually, agents that make violations will be socially excluded from the system, because no agents communicate with untrustworthy agents.

5 Sample Application

Our aim here is to define a sample application to show how all the framework components instantiate on this application.

5.1 Photo Sharing

The application that we consider here is a photo sharing social network. Basically, users can share pictures with their contacts who can, in turn, share again those pictures with their own contacts and so on. We provide the users with an assistant agent that will do all the assessment described before to inform the user of any violation. The final decisions lies in the hands of the user, the assistant does not take any decision.

In this system, the pictures are the information that is exchanged.

5.2 Primitives Instantiation

Some of the primitives we defined earlier need to be specified for this application. The primitives based on meta-information always remain the same, because the nature of the meta-information does not change. So do the primitives for transmission roles.

We can explain in more detail the primitives based on agent beliefs because the way they are inferred is what is interesting here:

- `context(C,I)` For the agent to believe that C is the context of information I, there are alternative solutions:
  - Look if there is a context tag emitted by a trusted agent
  - Analyse the picture to find its context (using image analysis techniques)
  - Ask the user attached to the agent to determine the context of the picture
- `target(X,I)` The same process can be used for the agent to believe that X is the target of I:
  - Look if there is a target tag emitted by a trusted agent
  - Analyse the picture to find if the target is on the picture
  - Ask the user attached to the agent to determine the target of the picture
- `link(X,Y)` By analysing the transmission chain in the meta-information, the agent can discover links between other agents.
- `knows(X,I)` Using the same technique, the agent can extract from the transmission chain the list of agents that received the information in the past.
- `role(A,R)` The agent asks the organisational infrastructure to know the possible roles of A.
- `rolecontext(R,C)` The agent asks the organisational infrastructure to know the possible roles fitting in context C.
- `policyvalid(P,I)` The agent infers on his belief base to see if the policy is valid as explained in section 3.5.

With the primitives instantiated, it is easy to check the policies, the A-laws and all needed components. In the next section we show an example of what happens in the application.

5.3 Use Case

Alice wants to share a picture with Bob. The target of the information is James, who is in an awkward position on the picture. Some of James' friends already had this information before, therefore, there are tags describing the context as "James friends" and the target as "James". No policy has been attached. The unique identifier of the information is "pic254". The message is identified by "mess412".

When Alice clicks on the button to send the picture to Bob, the assistant agent checks the PENs:

- PEN 1: Does the agent violates the A-laws by sending the information to the receiver? This is the instantiation of the laws described in section 3.4:
  - The declared context is set by the agent, so the declared context fits the context the agent believes to be the real one, the following formula holds:

```
fitcontext('James friends',mess412):-
   information(mess412,pic254),
   propagator('Alice',mess412),
   context('James friends',pic254),
   contexttag('James friends','Alice',mess412).
```

  - The assistant agent is not able to find a role for Bob that fits into the context "James friends", the formula does not hold:

```
fitrole('James friends',mess412):-
   receiver('Bob',mess412),
   role('Bob',?),
   rolecontext(?,'James friends').
```

  - No policies were defined, therefore, the first `fitpolicy(M)` statement holds (no policy exists for none of the target of the information).

  The following Appropriateness formula does not hold, because Bob is not a friend of James (the target):

```
appropriate(mess412):-
   fitcontext('James friends',mess412),
   fitrole('James friends',mess412),
   fitpolicy(mess412).
```

  Beyond this point, the assistant agent knows that the transmission will be inappropriate, and therefore violates the PENs. Anyway, he asks the user (Alice), what to do: continue or abort the transmission?

Alice wants to continue. The message containing the picture and meta-information is sent to Bob.

Bob's agent handles the information by checking the PENs:

- Does the message violates contextual integrity? Bob's agent runs here the same test that Alice's agent did (using his own beliefs). As Bob is not a friend of James, no roles fits in the context "James friends" and a violation is therefore detected.

Bob's agent adjusts his beliefs, he does not trust Alice anymore because it is not the first time that Alice deceives Bob. He sends to all his contacts a "punishment message" containing the meta-information (context tags, target tags, transmission chain) and the following description:

```
pen1violation(Alice,mess412),
information(mess412,pic254),
context(C,mess412),
\+ fitrole(C,mess412).
```

Dave's agent is one among those who receives this message. Dave was about to send a message to Alice, when he clicks the "send" button, his agent checks the PENs. Then PEN 3 forbids to send a message to an untrusted partner. Dave's agent warns him that Alice is untrustworthy and that the transmission will violate the PENs.

Users stop communicating with Alice because of the violation she made. Alice is now socially excluded, she is yet still in the system but nobody keeps communicating with her.

The example is a little bit hard on Alice in order to show the power of social exclusion. Normally, it will take multiple violations for someone to be excluded from the system and forgiveness could occur after a certain time.

6 Conclusions

The framework we presented in this article allows to protect users privacy in a decentralised and open system when it is not possible to apply computer security approaches. Based on Nissenbaum's Contextual Integrity theory, we propose an approach using appropriateness laws that defines what is an appropriate information transmission (and therefore, what is inappropriate). Primitives are defined to express these laws and to allow agents to define preferences over the transmission of some specific information.

Agents in the system play both roles of actors and judge: they transmit information, and they detect violations. Agents also inform others when they spot a violation, so that the violating agents are excluded of the system. This behavior is directed by the Privacy Enforcing Norms (PEN) described in this article.

There are still some unsolved problems in the system, that may prevent it from working correctly:

- The trust related problems: "what happens if there are too many malevolent agents in the system?"
- "Journalist Problem": "what happens if an agent decides to sacrifice himself to become a relay for information that violates privacy?" (the original source is never punished, only the journalist).
- Reputation Paradox: Information about reputation is libellous in a way, so it can generate privacy violation. But at the same time, it is required for maintaining information regarding agents that make violations.

In our future works, we will integrate the trust mechanisms directly in the decision process, *i.e.* decompose the primitives that rely on trust in predicates. We will, at the same time, investigate the problems related to the subjectivity and related to strategic manipulation (agents sending fake violation messages for example). Then an application will be developped to probe the system in the real world by providing assistant agents to users.

References

1. Barth, A., Datta, A., Mitchell, J., Nissenbaum, H.: Privacy and Contextual Integrity: Framework and Applications. 2006 IEEE Symposium on Security and Privacy (S&P'06) pp. 184–198, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1624011
2. Bell, D.E., LaPadula, L.J.: Secure computer systems: Mathematical foundations. Tech. rep., Technical Report MTR-2547 (1973)
3. Byun, J., Bertino, E., Li, N.: Purpose based access control of complex data for privacy protection. In: Proceedings of the tenth ACM symposium on Access control models and technologies. p. 110. ACM (2005)
4. Crépin, L.: Les Systèmes Multi-Agents Hippocratiques. Ph.D. thesis (2009)
5. Hübner, J.F., Boissier, O., Kitio, R., Ricci, A.: Instrumenting multi-agent organisations with organisational artifacts and agents. Autonomous Agents and Multi-Agent Systems 20(3), 369–400 (mai 2009), http://www.springerlink.com/content/g115t233633v6h16
6. Mont, M.C., Pearson, S., Bramhall, P.: Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. In: Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on. pp. 377–382 (2003)
7. Nissenbaum, H.: Privacy as Contextual Integrity. Washington Law Review pp. 101–139 (2004)
8. Piolle, G.: Agents utilisateurs pour la protection des données personnelles: modélisation logique et outils informatiques (2009)
9. Reagle, J., Cranor, L.F.: The platform for privacy preferences. Communications of the. ACM 42(2), 48–55 (1999)
10. Rivest, R.: The MD5 Message-Digest Algorithm. Distribution (1992)
11. Rivest, R.L., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public- Key Cryptosystems. Communications 21(2) (1978)
12. Sabater, J., Paolucci, M., Conte, R.: Repage: REPutation and ImAGE Among Limited Autonomous Partners. Journal of Artificial Societies and Social Simulation 9(2), 3 (2006)
13. Vercouter, L., Muller, G.: L.i.a.r.: Achieving social control in open and decentralised multi-agent systems. Applied Artificial Intelligence (2010), to appear