Privacy is Linking Permission to Purpose*

Fabio Massacci[1] and Nicola Zannone[1]

Department of Information and Communication Technology
University of Trento - Italy
{massacci,zannone} at dit.unitn.it

Abstract. The last years have seen a peak in privacy related research. The focus has been mostly on how to protect the individual from being tracked, with plenty of anonymizing solutions.
We advocate another model that is closer to the "physical" world: we consider our privacy respected when our personal data is used for the purpose for which we gave it in the first place.
Essentially, in any distributed authorization protocol, credentials should mention their purpose beside their powers. For this information to be meaningful we should link it to the functional requirements of the original application.
We sketch how one can modify a requirement engineering methodology to incorporate security concerns so that we explicitly trace back the high-level goals for which a functionality has been delegated by a (human or software) agent to another one. Then one could be directly derive purpose-based trust management solutions from the requirements.

1 Privacy Protection and Cleaning Ladies

Consumer privacy[1] is a growing concern in the marketplace. While the concerns are most prominent for e-commerce, the privacy concerns for traditional transactions are increasing as well. Some enterprises are aware of these problems and of the market share they might loose if they do not implement proper privacy practices. As a consequence enterprises publish privacy statements that promise fair information practices[2].

There are a number of risks to an enterprise if it does not manage its personally identifiable information correctly. Recently, many countries have promulgated a new privacy legislation. Most of these laws incorporate rules governing

* This work has been partially funded by the IST programme of the EU Commission, FET under the IST-2001-37004 WASP project and by the FIRB programme of MIUR under the RBNE0195K5 ASTRO Project. We would like to thank P. Giorgini, M. Pistore, and J. Mylopoulos for many useful discussions on Tropos.

[1] Privacy is the right of individuals to determine for themselves when, how, and to what extent information about them is communicated to others (Alan Westin).

[2] The OECD defined a set of privacy principles in 1980. The document OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data is considered to contain the core requirements for managing privacy today.

collection, use, store and distribution of personally identifiable information. It is up to an organization to ensure that data processing operations respect any legislative requirements. If an enterprise breaches trust, that is, it uses data for other purposes, then it can be sued. Further, business relationships are built on trust. Organizations that demonstrate good privacy practices can build trust. Organizations with no privacy practices will turn away customers.

The last years have seen a substantial increase in privacy-related security research[3]: we have a number of dedicated workshops (e.g. PET, WPES), a number of European and US projects and standard initiatives such as P3P. In the realm of cryptography the work on anonymizing networks and transactions has a long history since Chaum's first proposals. One could even say that trust negotiation's birth itself [11] was spurred by privacy concerns of non-disclosure of sensitive credentials to unknown strangers.

However, the current set of solutions is slight unsatisfactory: we must either struggle on keeping privacy by a complex cryptographic infrastructure or be involved in complex protocols for trust negotiation.

Not long ago, at a Cambridge seminar, Robert Morris, formerly from the US NSA, spoke about the cryptographic role of the cleaning ladies, inviting people to consider the actual perimeter of one's secure systems. We would like to use the example with a slightly different view.

Indeed, from the standpoint of access control and privacy protection, the problem of the cleaning ladies admits no solution. The cleaning lady must have access to the room and even when the room is not occupied to avoid disturbing people while at work. There is no way to prevent her from looking at the papers laying on the desk, even searching through them, doing any possible action on the unattended desktop and leaving unnoticed.

Yet, we do allow cleaning ladies and we are perfectly happy with many others similar problems. One possible explanation may be in the the actual economics of trust in cleaning ladies wrt the risk of privacy losses [12]. However, we believe that the solutions lies in the implicit permission model that we use in the physical world, but so far we have not implemented in trust management protocols.

So it is useful to consider how the legal profession defines the *power of attorney*: with the general power of attorney the individual appointed attorney in fact is vested with unlimited powers for an indefinite amount of time (unless otherwise explicitly specified). The validity of this of power of attorney ceases only with a specific written revocation or with the death of the person that has granted it. This is the way we used as delegation of "identity" works in trust management systems[4], and is also the idea behind usage of data by systems. Once the system has the data, it will be used. We can read privacy policies to guarantee that the data will not be misused but we have no way to know that this policy will actually be enforced by the system. Privacy polices are added *after* the system has been built.

[3] A comprehensive and updated literature survey can be found at http://www.freehaven.net/anonbib/date.html.

[4] This is the A speaks for B paradigm of SDSI/SPKI [6]

What is interesting for our purposes is the *special power of attorney*: the individual appointed attorney is vested solely with the power needed to carry out a specific affair (i.e. the sale or purchase of a real estate or a car). Therefore, in the formal contract it is necessary to indicate exactly the particular power that the principal intends to give to his attorney. Actual documents state quite explicitly the powers but also the goal for with these powers have been delegated. If I have to sell a house I may do all that I deem fit, but everything got tagged by that purpose and if my action is clearly not necessary I might be asked to pay for incurred losses. This is the missing important twist: in our trust management and data processing systems we have implemented only the description of powers but not the purpose.

Back to the cleaning lady, in giving her the door's key we have stipulated (or better our administration has) a contract that she has the permission of entering the room with the goal of cleaning it. Any other action would constitute breach of contract and would result in fees or contract resolution.

That's our claim: *we feel that our privacy is protected because the permission we are granting is linked to a purpose*. Notice that the the purpose is not at all security related but rather a functional goal of the system.

There are already solutions to link permissions to purpose in databases, such as the work on Hippocratic databases [1], but we found no proposals for other methods for data and credential management in distributed trust management or trust negotiation. However, it is not difficult to add yet another field to an X.509/SPKI/etc. certificate format.

The intriguing issue is how can we link the permission in a credential doled out by the security sub-system to the actual functional goals of the entire system?

The strategy that we envisage is the following:

- find a "traditional" requirements methodology in which functional goals can be made explicit;
- enhance the methodology with security-related features such as trust and delegation that are linked to the explicit goals;
- each time a delegation of permission must be foreseen by the system designer, she can trace back on the design the goals and make that explicit.

2 Related Work

The last years have seen an increasing awareness that privacy plays a key role in system development and deployment. This awareness has been matched by a number of research proposals on privacy. Next, we present some of those adopting solution to link permissions to purpose.

Platform for Privacy Preferences[5] (P3P), developed by the World Wide Web Consortium (W3C), is an emerging standard whose goal is to enable users to gain more control over the use of their personally identifiable information (PII) on web sites they visit. P3P enables web sites to express their privacy practices

[5] http://www.w3.org/TR/P3P/

in a standard format that can be retrieved automatically and interpreted easily by user. P3P provides a way for a web site to encode its data-collection and data-use practices in a machine-readable XML format known as a P3P policy.

P3P policies provide contact information for the legal entity making the representation of privacy practices in a policy, enumerate the types of data or data elements collected, and explain how the data will be used. In addition, policies identify the data recipients, and make a variety of other disclosures including information about dispute resolution, and the address of a site's human-readable privacy policy. In other words, P3P policies represent the practices of the site. Each P3P policy is applied to specific web resources listed in a policy reference file. By placing one or more P3P policies on a web site, a company does not make any statements about the privacy practices associated with other web resources not mentioned in their policy reference file, with other online activities that do not involve data collected on web sites covered by their P3P policy, or with offline activities that do not involve data collected on web sites covered by their P3P policy.

Agrawal et al. [2] propose an server-centric architecture for P3P. The P3P protocol has two parts: *Privacy Policies*, an XML format in which a web site can encode its data-collection and data-use practices, and *Privacy Preferences*, an XML format for specifying client privacy preferences. In the server-centric architecture, a web site first installs its privacy policy in a data system. Then database querying is used for maching a user privacy preference against privacy policies. Finally, web site sent result of matching preference against policy to the client, and the client requests web page if policy conforms to his/her preference. This approach is different from ours, because it does not explain the permission purpose that is implicit in privacy policies of the web site. Further, the client have to choose without knowing all the available policies of the web site. For example, there are web sites that provide services only if a client agrees a certain policy, but they show multi-policies without specifying which policy is sufficient to get the service. In this case the client could grant more permissions than necessary.

In summary, as a first step towards managing privacy, organizations publish privacy promises. The P3P statements can be used by a P3P client (e.g. the Internet Explorer 6 web browser) to notify the user automatically whether the privacy policy of the enterprise matches that configured by the user. However, this is not sufficient to guarantee the enforcement of the promises that enterprises have made, and has resulted in privacy violations, even from well meaning companies. In fact, P3P is an language for expressing privacy promises on web sites, but it cannot be used to enforce them within an enterprise.

The Enterprise Privacy Authorization Language[6] (EPAL), developed by IBM, enables an enterprise to formalize the exact privacy policy that shall be enforced within the enterprise. It formalizes the privacy promises into policies and associates a consented policy to each piece of collected data. This consented policy can then be used in access control decisions to enforce the privacy promises made. The EPAL policy language categorizes the data an enterprise holds and

[6] http://www.zurich.ibm.com/security/enterprise-privacy/epal/

the rules which govern the usage of data of each category. An EPAL policy is essentially a set of privacy rules. A rule is a statement that includes a data user, an action, a data category, and a purpose. A rule may also contain conditions and obligations.

Next, we present an architecture for implementing privacy management based on EPAL. During submission of PII, the privacy management system (by submission monitors) will create the submission records. This data is a permanent record of when PII was submitted, what privacy policy version was in place at that time, and what the users preferences were. Later, when PII is to be accessed, the privacy management enforcement monitors ensure that only data accesses are allowed that conform to the privacy policy. They also create access records that record which user accessed the data and for what purpose. The combination of submission and enforcement monitors allow the enterprise to prove that it is a good data keeper and gives the enterprise some assurance that it is enforcing its stated privacy policy.

EPAL aims at formalizing enterprise-internal privacy policies. This requires a vocabulary that formalizes the privacy relevant aspects of an enterprise. It also includes a hierarchy of purposes for which the enterprise collects data. On the other hand, P3P aims at formalizing privacy statements that are published by an enterprise. The goal is to define a machine-readable equivalent for the human readable privacy promises that are published as a privacy statement on a web page. Unlike EPAL, P3P defines a global terminology that can be used to describe the privacy promises for any enterprise. Although P3P is well suited for expressing policies, it is not as suitable for expressing an internal enforceable privacy policy. EPAL on the other hand is designed specifically to express an internal privacy policy that can be enforced by an enterprise privacy management system. IBM is currently investigating how to project a P3P policy from EPAL.

There are already solutions to link permissions to purpose in databases. Following this approach, Agrawal et al. show that the database community has the opportunity to play a central role re-designing databases to include responsability for the privacy of data as a fundamental tenet. Inspired by the Hippocratic Oath, they propose to call Hippocratic databases [1] those databases that have privacy as a central concern. Agrawal et al. propose the key principles for such Hippocratic database systems, distilled from the principles behind current privacy legislations and guidelines. Particularly, such principles highlight that the purposes for which PII has been collected shall be stored with that information in the databases, that the purposes associated with PII shall have consent of the owner of data, and that the PII collected, used, and stored shall be limited to the minimum necessary for accomplishing the specified purposes.

Hippocratic databases can be useful to add enforcement dimension to P3P. As we have seen above, a P3P policy essentially describes the purpose of the collection of information along with the intended recipients. The policy description uses data tags to specify the data items for which the policy is being stated. P3P's concepts of purpose and retention can be mapped directly into analogous concepts in Hippocratic databases. Thus, from a P3P policy it is possible to

generate the corresponding data structures (i.e., a privacy-policies table) in the Hippocratic database system.

What is still missing in these proposals is capturing the high-level privacy requirements, without getting suddenly bogged down into security solutions or cryptographic algorithms. If we look at the requirements refinement process of many proposals, we find out that at certain stage a leap is made: we have a system with no privacy features consisting of high-level functionalities, and the next refinement shows encryption, access control and authentication. The modeling process should instead makes it clear why encryption, access control and authentication are necessary. This work is a step in this direction closing the gap between the functional and privacy requirements and the trust management architecture that is now emerging as the standard way to implement security in distributed systems.

3 Goal-Oriented Security Engineering

The basic building block is a "traditional" requirements methodology in which functional goals are explicit: the Tropos framework. It is an agent-based software engineering methodology [3, 4] that strives to model both the organizational environment of a system and the system itself. It uses the concepts of actor, goal, plan, resource and social dependency for defining obligations of actors (dependees) to other actors (dependers). *Actors* have strategic goals within the system or the organization and represent (social) agents (organizational, human or software), roles etc. A *goal* represents some strategic interest of an actor. A plan represents a way of doing something (in particular, a plan can be executed to satisfy a goal). A resource represents a physical or an informational entity. Finally, a *dependency* between two actors indicates that one actor depends on another to accomplish a goal, execute a plan, or deliver a resource. In the sequel, when the distinction between goal, plan or resource is not essential we use the term *service* to denote any of them.

Tropos has been designed with cooperative information systems in mind and therefore it is already equipped with a methodology for reasoning about *functional delegation* of goals. Goal delegation arises quite naturally among cooperative, rational actors: every actor pursues its own goals, goal partitioning is a standard divide-and-conqueror strategy, and usually in a collaborative environment there are enough hierarchy and trust relationships, so that an actor is likely to find some other one to delegate a subgoal. When considered from an actor coordination perspective, goal delegation has two main facets:

– *Delegation of commitment.* This means that the delegatee should embrace the intentions of the delegater, trying to fulfill the goal as it was one of its own. From delegater point of view, this requires a kind of trust: the delegater has to believe that the delegatee is trustworthy and will honestly try to achieve the goal.

– *Delegation of strategy.* Delegating a declarative goal instead of an operational plan means that the delegater is interested only in the resulting outcome and not in a specific way the delegatee fulfill it.

The incorporation of security features in Tropos is not trivial and is discussed in another paper [7]. It is essentially based on the following intuition: in the dependency relationship it is implicitly assumed that if I delegate the execution of a service to somebody else I'm implicitly also the owner of this service. This implicit assumption is no longer true when also security requirements and not just functional requirements are part of the target.

In this modeling framework, four relationships (beside the old functional dependency) can be singled out:

trust (among two agents and a service), so that A trust B on a certain goal G;
delegation (among two agents and a service), whenever A explicitly delegates to B a goal, or the permission to execute a plan or access a resource;
offer (between an agent and a service), so that A can offer to other agents the possibility of fulfilling a goal, executing a plan or delivering a resource;
ownership (between an agent and a service), whenever an agent is the legitemate owner of a goal, plan or resource.

Note the difference between trust and delegation. Delegation marks a formal passage in the requirements modeling: a TM certificate will have to be eventually issued for the delegatee when implementing the system. In contrast, trust marks simply a social relationship that is not formalized by a "contract" (such as digital credential). There might be cases (e.g. because it is impractical or too costly), where we might be happy with a "social" protection, and other cases in which security is essential. In this model, there is no relationship between trust and delegation

The requirements engineering methodology proposed in [7] specifies how to derive the trust management system (aka the delegation relationship) from the general requirement:

1. design a trust model among the actors of the systems;
2. identify who owns goals, plans, or resources and who is able to fulfill goals, execute plans or deliver resources;
3. define functional dependencies (functional delegations) of goals among agents building a functional model;
4. define a trust management implementation (e.g. based on the Delegation logics by Li et al. [8–10]) in which the delegation of permissions is defined.

In [7] it is also shown how one can use Datalog and the DLV system [5] to model check the correctness of the implementation wrt the previous two model or the consistency of the functional model with the trust model.

However, in [7] there is no notion of Goal-linked permission, though the framework have all necessary machinery, simply because all considered target trust management systems for the last phase have no notion of Goal-linked permission.

4 A basic e-Health case study

Here we show a very basic case study, based on a modelling an e-Health service. We consider the following actors:

- *Patient*, that depends on the hospital for receiving appropriate health care;
- *Hospital*, that provides medical treatment and depends on the patients for having their personal information.
- *Clinician*, physician of the hospital that provides medical health advice and, whenever needed, provide accurate medical treatment;
- *Health Care Authority* (HCA) that control and guarantee the fair resources allocation and a good quality of the delivered services.
- *Medical Information System*, that, according the current privacy legislation, can share the patients medical data if and only if consent is obtained. The *Medical Information System* manages patients information, including information about the medical treatments they have received.

In order to provide rapid and accurate medical treatments, clinicians need a fast access to their patient' medical data. Similarly, HCA needs a fast and reliable access to the data in order to allocate effectively the available resources, and guaranteeing then that each patient can receive a good quality of medical care. Furthermore, HCA wants to be sure that the system cannot be defrauded in any way and that clinicians and patients behave within the limits of their roles. To the other hand, the obvious right of the patient to restrict access on his/her medical data and moreover, to be able to use some safeguards on the privacy of these data, should be taken into serious consideration. The patient's consent must be requested, and he must be notified when its data is shared.

Figure 1 and Figure 2 show respectively the trust model and the functional model. Actors are represented as circles, goals by ovals, plans by polygons and relationships by labelled arrows. The ownership relationship has an edge labelled by **O**. We use trust (**T**) to model the basic trust relationship between agents and permission (**P**) to model the actual transfer of rights in some form (e.g. a digital certificate, a signed paper, etc.). This is the delegation of permission. The plain **D** is used for dependency, that is the functional delegation.

In the trust model *Patient* trusts *HCA* and *Clinician* for his personal information, and *HCA* trusts *Hospital* for it. Further, *Hospital* trusts *HCA* for checking equity resource distribution. *Clinician* trusts *Hospital* for medical treatment and for requesting specific professional consulting, and *Hospital* trusts *Clinician* for providing such consulting and for patient personal information. We also consider the trust relationship between *Hospital* and *Medical System Information* for patient personal information. Notice on top of Fig. 1 that there is a trust relationship between two actors (HCA and Hospital) on a resource that is owned by neither of them.

In the functional model, *Patient* depends on *Hospital* for medical treatments, and in turn, *Hospital* depends on *Clinician* for such treatments. To provide accurate medical treatment, *Clinician* can request specific professional consultancy

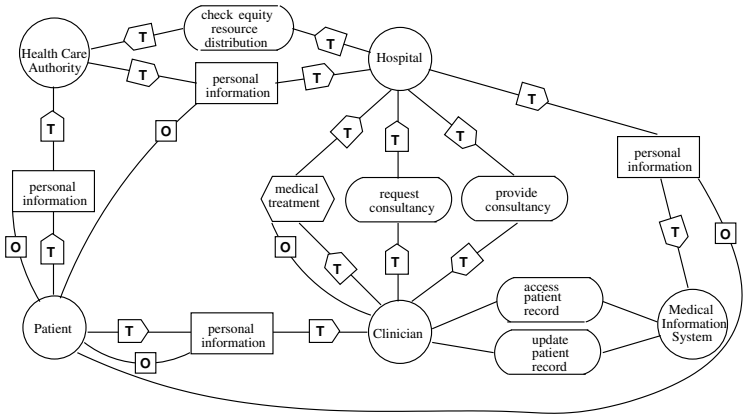

Fig. 1. Health Care System trust model

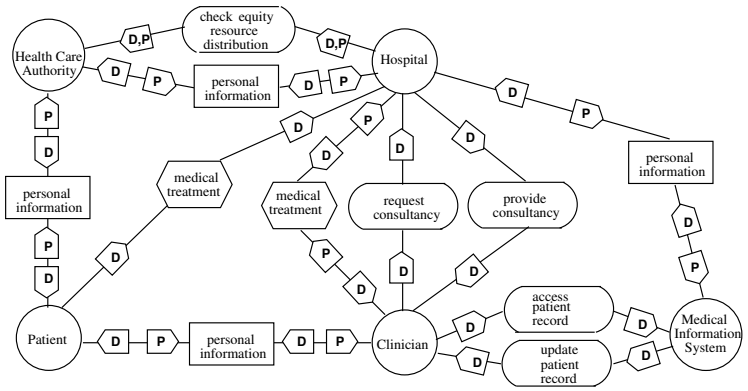

Fig. 2. Health Care System functional model

to *Hospital* that depends on other *Clinicians* for this consultancy. *Hospital* delegates the goal of checking equity resource distribution to *Health Care Authority*. *Clinician* and *Health Care Authority* need patient personal information to fulfill their service. Thus, *Patient* delegates them his personal information. Further, *Health Care Authority* re-delegates these data to *Hospital*.

Notice that it is not necessary for the owner of the data to delegate the data directly to the entity that will use the service. For example here the patient can delegate the usage of his personal information to the HCA with a certain depth so that HCA can eventually re-delegate it to the actual service provider.

This trust-functional-TM implementation process is not static: requirements can be refined, new actors can be introduced, delegations can be split or passed further down the line, etc. To clarify refinement analysis in Tropos, one can use rationale diagrams that explain relationships among actors and decompose high level goals into subgoals as in Figure 3. This makes even more explicit how permissions should be linked to high-level goals.

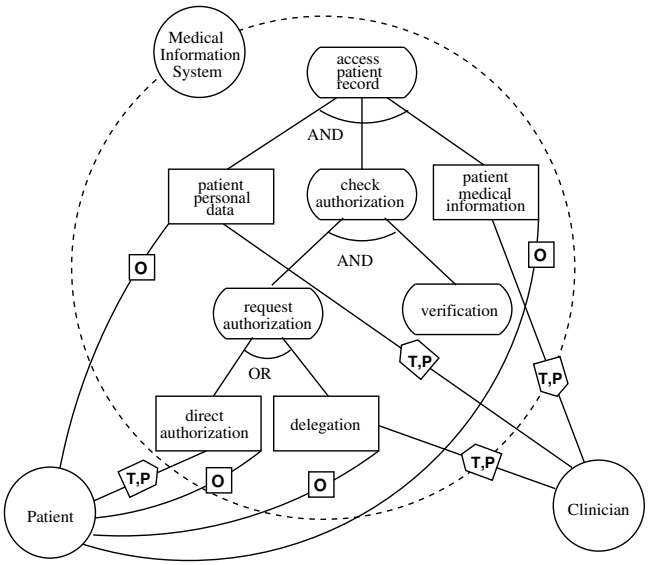

Fig. 3. Rationale Diagram

5 Linking Permission to Purpose

Now we have all the necessary machinery to ensure the patient that his privacy will not be violated. It is of course possible to specify in details all possible delegations in this model. Indeed, this is what has been done in the paper [7]. The formal analysis carried out there has shown that this process is extremely error prone and even the expansion of delegation certificates to include blacklists may not be sufficient to rule out certain delegation paths.

In this setting is may seem unnecessary to link permission to goals: after all, few lines above we have just defined delegation as a ternary relations between a pair of agents and a service (which might be a goal). This is not what we meant by tagging permissions to purpose. The three-place delegation is a delegation of a permission. Simply the framework makes it possible to delegate something better than a simple action such as exec, or a resource. So I can delegate a plan (which is just a big composite *fixed* action) or a goal in which case I simply delegate all possible plans that can fulfill the goal. If we go back to the special power of attorney we have only mentioned the how but not the why.

The *direct base case* for linking permission to purpose is the following:

1. the owner and the depender of a service are the same actor
2. the service is a primary service of the depender and has not been obtained by refinement from other goals[7].

[7] In the Tropos methodology only goals can be refined. We stick to this line here, though from a security perspective also resources may be additionally refined for fine-grained access control.

In this case the delegation of the service from the owner to the provider of the service is tagged by the service of the functional dependency of the owner.

In the *reverse base case* the owner and the dependee of the service coincides. Also in this case the credential should be tagged with the goal of the owner but it is worth noting that the emission of this credential is not trivial. Indeed, the owner of the service should have some reason to delegate a service besides a cooperative spirit.

Notice that this is not really the case in our e-health example. For example the Hospital do need the patient personal data (of which the patient is at the same time the owner and the dependee). However, in this case the patient personal data is obtained during the design process as a refinement of the patient's own goal of obtaining care which has been delegated to the hospital. In most practical cases this is the typical format: an high-level goal of agent A is delegated to agent B and after a suitable number of recursions and refinements a subgoals is delegated from C back to A.This is typically providing information for actually fulfilling some other task necessary for the overall goal.

In the *general case* we have that

1. owner A, depender B, and dependee C of a service are all different actors,
2. the service S is a derived service from another goal G_d of the depender,
3. the service is also a derived service from a possibly different goal G_o of the owner.

The simplest solution is to add the *conjunction* $G_d \wedge G_o$ of the owner's goal and the depender's goal to the digital credential. There might be cases in which the designer may want different schemes.

After the linkage between permission and purpose has been put in place, we can check that a delegation chain is also an appropriate delegation of purpose by checking that each goal along the delegation is a subgoal of the initial base case and each step is also appropriate. This can be easily done within the same datalog framework used in [7]

An interesting question is whether we should have also noted in the delegation credential the actual agent names to which the "purpose"-goals belonged. So far we could not find a situation in which such information is needed and cannot be reconstructed from the delegation chain.

6 Conclusions?

In framework we have proposed, privacy is considered during the whole process of requirements analysis modeling trust and delegation relationships between the stackholders and the system-to-be. In this way, the framework we propose allows to capture privacy requirements at an organizational level, and hence, to help designers to model privacy concerns throughout the whole software development process.

In this paper we have discussed how a trust management (sub)system can accommodate the notion of purpose of a permission by linking it directly with

the functional requirement of the overall information system. This makes possible to capture the high-level privacy requirements without taking cryptographic algorithms or protocols for trust negotiation into considerations.

There are a number of open questions that we have not answered and that are worth discussing:

- which *format* can be used for goals in certificates? A string field costs nothing to include but in this way all possibilities that we have listed of linking goals to subgoals would be lost as only equality would be tested. So a semantic web solution may be used but this may be costly from a processing perspective.
- are *distributed implementations* possible? and in particular can existing implementations of Trust Management systems be ported to this new framework (at least under the assumption that somebody has already magically derived the goal-oriented trust management implementation)? How effective and essential should purpose verification in the credential chain be?
- how *history* should be presented in the purpose? One possible solution is no history at all, just mention the last delegation step; another solution could be let history be recovered by the chain of credentials so that one could finally check if all credential have been gathered for the appropriate purpose.
- what kind of *automatic support* could be available for the automatic synthesis and validation of the trust management implementation.
- can we define a similar process for the definition of Hippocratic databases?

References

1. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic Databases. In *Proc. of the 27th Int. Conf. on Very Large Data Bases (VLDB'02)*, 2002.
2. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. An Implementation of P3P Using Database Technology. In *Proc. of the 9th Int. Conf. on Extending Database Technology*, volume 2992 of *Lecture Notes in Comp. Sci.*, pages 845–847. Springer-Verlag Heidelberg, 2004.
3. P. Bresciani, F. Giunchiglia, J. Mylopoulos, and A. Perini. TROPOS: An Agent-Oriented Software Development Methodology. *J. of Autonomous Agents and Multi-Agent Sys. (JAAMAS)*, (To appear).
4. J. Castro, M. Kolp, and J. Mylopoulos. Towards Requirements-Driven Information Systems Engineering: The Tropos Project. *Inform. Sys.*, 27(6):365–389, 2002.
5. T. Dell'Armi, W. Faber, G. Ielpa, N. Leone, and G. Pfeifer. Aggregate Functions in Disjunctive Logic Programming: Semantics, Complexity, and Implementation in DLV. In *Proc. of the 18th Int. Joint Conf. on Artif. Intell. (IJCAI'03)*. Morgan Kaufmann Publishers, 2003.
6. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen. *SPKI Certificate Theory*, September 1999. IEFT RFC 2693.
7. P. Giorgini, F. Massacci, J. Mylopoulous, and N. Zannone. Requirements Engineering meets Trust Management: Model, Methodology, and Reasoning. In *Proc. of the 2nd Int. Conf. on Trust Management (iTrust 2004)*, Lecture Notes in Comp. Sci. Springer-Verlag Heidelberg, 2004.

8. N. Li, B. N. Grosof, and J. Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Trans. on Inform. and Sys. Sec. (TISSEC)*, 6(1):128–171, 2003.
9. N. Li and J. C. Mitchell. Datalog with Constraints: A Foundation for Trust-management Languages. In *Proc. of the 5th Int. Symp. on Practical Aspects of Declarative Lang. (PADL'03)*, 2003.
10. N. Li and J. C. Mitchell. RT: A Role-based Trust-management Framework. In *Proc. of DARPA Inform. Survivability Conf. & Exposition (DISCEX'03)*, 2003.
11. K. E. Seamons, M. Winslett, T. Yu, L. Yu, and R. Jarvis. Protecting Privacy during On-line Trust Negotiation. In *Proc. of the 2nd Workshop on Privacy Enhancing Technologies*, 2002.
12. P. Syverson. The paradoxical value of privacy. In *Proc. of 2nd Annual Workshop on Economics and Inform. Sec. (WEIS 2003)*, 2003.