A Methodology for Designing Controlled Anonymous Applications

Vincent Naessens[1] and Bart De Decker[2]

[1] K.U.Leuven Campus Kortrijk, Department of Computer Science,
E. Sabbelaan 53, 8500 Kortrijk, Belgium
Vincent.Naessens@kuleuven-kortrijk.be
[2] K.U.Leuven, Department of Computer Science,
Celestijnenlaan 200A, 3001 Heverlee, Belgium
Bart.DeDecker@cs.kuleuven.be

Abstract. Many anonymous applications offer unconditional anonymity to their users. However, this can provoke abusive behavior. Dissatisfied users will drop out or liability issues may even force the system to suspend or cease its services. Therefore, controlling abuse is as important as protecting the anonymity of legitimate users. However, designing such applications is no sinecure. This paper presents a methodology for designing controlled anonymous environments. The methodology generates a conceptual model that compromises between privacy requirements and control requirements. The conceptual model allows to derive performance and trust properties and easily maps to control mechanisms.

1 Introduction

Many existing privacy-enhancing applications (anonymous mail systems, anonymous publication services, etc) offer unconditional anonymity to their users. As unconditional anonymity can provoke abusive behavior (spam mail, publishing copyrighted or criminal contents, etc) and as new application domains are explored (anonymous payments, anonymous auctions, etc), anonymity control becomes an extremely important issue. However, designing controlled anonymous applications is no sinecure because of several reasons:

- *Opposite requirements.* A reasonable trade-off must be found between the interests of the users (which mainly have privacy requirements) and the interests of the system administrators and law enforcement (which mainly have control requirements).
- *Complexity of building blocks.* Anonymous credentials [2, 3] are used as building block for anonymity control. However, enhancing an application with anonymous credentials while preserving the anonymity requirements is no sinecure. Some models achieve an acceptable level of control but do not comply with the anonymity requirements. Others do not succeed to capture all control requirements.

This paper presents a methodology that eases the design of controlled anonymous applications. The advantages of the methodology are fourfold. First, the methodology allows designers to express anonymity requirements and control requirements at a high

Please use the following format when citing this chapter:
Author(s) [insert Last name, First-name initial(s)], 2006, in IFIP International Federation for Information Processing, Volume 201, Security and Privacy in Dynamic Environments, eds. Fischer-Hubner, S., Rannenberg, K., Yngstrom, L., Lindskog, S., (Boston: Springer), pp. [insert page numbers].

level. Second, the methodology generates a conceptual model from which anonymity/-trust/performance properties can be derived. Third, the methodology provides alternative design decisions that partially avoid conflicts between requirements and proposes reasonable conflict resolution strategies. Fourth, the conceptual model can easily be mapped to control mechanisms.

The paper is organized as follows: section 2 describes some terminology; an overview of the methodology is given in section 3. Section 4 evaluates the methodology. Section 5 discusses related work. We conclude in section 6 with a summary of the major achievements.

2 Basic terminology

This section defines the basic terminology that is used in the rest of this paper, namely actions, environmental attributes and rights.

An *action* is a sequence of interactions between two subjects: a user and a service provider[3]. The set of actions in a particular system is $\mathcal{A} = \{A_1, ..., A_k\}$. Actions are either anonymous or identifiable. An action is identifiable if any publicly known unique environmental attribute of the sender is revealed when the action is performed. Otherwise, it is anonymous.

An *environmental attribute* is a user's attribute whose lifetime extends to multiple actions. $\mathcal{E} = \{\varepsilon_1, ..., \varepsilon_l\}$ defines the set of environmental attributes. Access to certain services can depend on the value of these attributes. Some environmental attributes (such as a mailbox address) are *unique* and others (such as age) aren't. A special type of environmental attribute is defined, namely *ID*. *ID* refers to any publicly known unique environmental attribute (such as SSN, driver's license number, etc) of an individual.

A *right* is a token that is required to perform an action. Each right r_x belongs to a type R_i. $\mathcal{R} = \{R_1, ..., R_m\}$ defines the set of right types. We assume the following properties of rights[4]. First, the values of a set of environmental attributes can be stored in a right. Users retrieve new rights when performing certain actions successfully. Thereafter, the user can prove the ownership of that right. In addition, the subject may choose to prove any attribute (or property of these attributes). Hence, rights are used to fulfil access control conditions. We further assume that two or more proofs of the same right cannot be linked unless the value of a unique environmental attribute is proven. Moreover, a proof can be deanonymizable. If so, the proof can be linked to the right itself by a designated external entity if a condition is fulfilled.

3 Design methodology

First, the requirements are classified according to four categories: privacy requirements, control requirements, performance requirements and trust requirements. Next, a sam-

[3] Pfitzmann [11] distinguishes between senders and recipients. PRIME [1] distinguishes between users and data controllers. Hence, the terminology depends on the specific setting in which the concepts are used.

[4] Rights are introduced at the conceptual design phase to enforce control requirements. At the implementation phase, rights are mapped to anonymous credentials.

ple application is introduced on which the design steps will be applied. The conceptual design phase is discussed in the third subsection. Thereafter, we focus on how conflicts between design decisions and requirements can be avoided/resolved. Finally, the conceptual model is transformed using high-level credential primitives.

3.1 Requirements

Privacy requirements Privacy requirements are typically defined in terms unlinkabilities between two items. $Unlinkable(x, y)$ expresses that the two items x and y may not be linked. In our methodology, x and y are either actions or environmental attributes. However, unlinkability requirements between application data can easily be transformed to unlinkability requirements between actions. Assume that $d_i \in appl\_data(a_i)$ and $d_j \in appl\_data(a_j)$, then $Unlinkable(d_i, d_j)$ can be transformed to $Unlinkable(a_i, a_j)$. Pfitzmann [11] distinguishes between anonymity and unlinkability. However, each anonymity requirement can easily be transformed to an unlinkability requirement as follows: $Anonymous(a_i) \Leftrightarrow Unlinkable(a_i, ID)$. Although coarse-grained parameters reduce the maximal amount of unlinkability requirements, many expressions are still possible. The maximal amount of unlinkability requirements of an application with 8 actions and 7 environmental attributes is $(8 + 7)^2 = 225$. To reduce the amount of anonymity requirements that have to be expressed explicitly, unlinkability requirements are split according to three categories: high priority requirements should be preserved in all circumstances; moderate priority requirements should be preserved as long as the user abides the rules; low priority requirements are not expressed explicitly. However, the methodology aims to preserve unlinkabilities as much as possible.

Control requirements In controlled environments, access to certain services is restricted. The *access requirements* are split according to three categories. First, order/multiplicity constraints restrict the order of actions and the number of times actions may be performed. Second, service providers may enforce users to authenticate. Authentication can either be identifiable or attribute-based. In the latter case, the user must reveal the value (or properties) of environmental attributes.

Unfortunately, access policies cannot prevent all types of abusive behavior. *Control measures* are defined to discourage such behavior. However, control measures are performed only if some type of abusive behavior is detected: $fulfils(a_i, cond) \rightarrow measure$. The methodology supports three types of control measures. First, the system may demand to reveal the identity of the user (i.e, accountability). Second, it must be possible to revoke permissions of the subject. Third, the system may need to link attributes from the same user.

Performance requirements Modelling anonymity/control requirements may decrease the performance of the system. First, proofs/initializations of rights increase the *processing time* per action. Second, service providers have to *store evidence* to enable control measures. Moreover, the system may be more attractive if rights are stored on smart

cards. However, the storage capacity of smart cards is limited. Therefore, the stakeholders can define an upper bound for the amount of proofs/initializations per action, the maximal amount of rights in the system and the maximal amount of evidence that has to be stored.

Trust requirements Trust requirements are also defined from two viewpoints. Users fundamentally mistrust system entities. They expect them to collaborate with each other and to combine their knowledge. Consequently, at least one trusted external entity should be required to reveal certain links between items. On the other hand, the system needs to rely on entities to make evidence available when performing a control measure.

3.2 Description of the application

The design steps that are introduced in the next sections are applied to a sample application, namely an integrated student environment. Users (i.e, students) register at the registration desk R. Users must reveal uniquely identifying information when registering. After registration, they can create a mailbox and activate an e-learning environment. The mail servers and the e-learning environment are administered by M and E respectively. An *address* is assigned to each mailbox. Students can only retrieve mail from their own mailbox.

Students have to prove the *discipline* for which they subscribed to activate the e-learning environment successfully. After successful activation, students can perform certain actions (such as viewing announcements, consulting schedules, etc) within the e-learning environment under a pseudonym identifier *pid*. They can also submit complaints anonymously. Moreover, they can register for the courses within their discipline after which they can take self-tests about the course and participate in course discussions on chat boards under a *nym*. Each course is administered by the teacher T.

Users must be held accountable for sending abusive mail and for submitting offensive messages to chat boards. Further, system should take measures to discourage spam. In addition, one high priority unlinkability requirement and three moderate priority unlinkability requirements are expressed:

UR_1^h: $Unlinkable(pid,ID)$
UR_1^m: $Unlinkable(address,ID)$
UR_2^m: $Unlinkable(send, address)$
UR_3^m: $Unlinkable(nym,ID)$

3.3 Conceptual design phase

The conceptual design phase consists of three steps and uses multiple graph-based formalisms. In the initial design step, *multiplicity/order constraints* are modelled and represented in a Flow Graph. Next, the Flow Graph is transformed to a Petri Net representation which is further enhanced with *access constraints*. Finally, *control measures* are modelled. In addition, a linkability graph is used to evaluate the anonymity/trust properties. As multiple formalisms are used, the semantics of each attribute within a formalism

and a set of rules to transform a model within one formalism to a model within another formalism must be defined. For a detailed description of the transformation rules, we refer to [9].

Flow Graph In the initial design step, *multiplicity/order constraints* are represented graphically by a Flow Graph $FG = \{\mathcal{A}, E\}$. Each vertex represents one action $A_i \in \mathcal{A}$. Each directed edge $e = [A_i, A_j]$ defines that action A_i must precede action A_j. Moreover, a multiplicity is assigned to each edge. The multiplicity (with $multiplicity(e) \in \mathbb{N} \cup \infty$) denotes the number of times action A_j may be performed after action A_i. The order dependencies in the student application are represented in figure 1. Note that the service provider is assigned to each action.

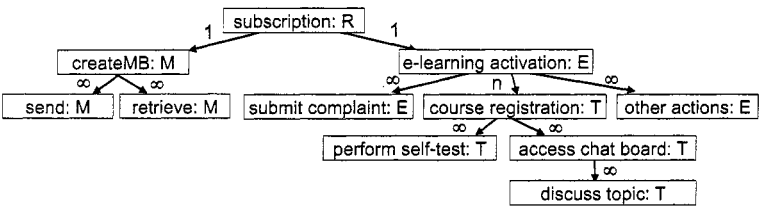

Fig. 1. Flow Graph.

Petri Net The Flow Graph is transformed to a Petri Net. A Petri Net is a four-tuple $(\mathcal{A}, \mathcal{R}, \mathcal{P}, \mathcal{I})$. Each transition corresponds to an action $A \in \mathcal{A}$; each place $R \in \mathcal{R}$ defines a container of rights of the same type; each token within a place R defines a right $r_i \in R$. each input arc $[R, A] \in \mathcal{P}$ defines that the user must spend a right $r \in R$ to perform action A; each output arc $[R, A] \in \mathcal{I}$ corresponds to a right $r \in R$ that is retrieved when action $A \in \mathcal{A}$ is performed successfully. A *natural number* is assigned to each input arc and each output arc. This number defines the number of rights that are spent and retrieved respectively. The default value is 1.

Each node in the Flow Graph maps to a transition in the Petri Net. The transformation of an edge in the Flow Graph depends on its multiplicity. A bidirectional arc between a place and a transition denotes that the number of valid tokens in the place is not changed when the transition has fired (see figure 2). Semantically, a bidirectional arc defines that the user must prove the ownership of a right; a unidirectional input arc defines a right that is spent.

After transformation, the Petri Net is enhanced with *access constraints*. First, each *input arc* is labelled with a set of (properties of) environmental attributes that a user must prove to perform a certain action. For instance, the user must prove the discipline $disc$ for which he subscribed when activating the e-learning environment. Similarly, the user must prove that the $course$ for which he wants to take a self-test forms part

of his curriculum. Second, each *output arc* is labelled with a set of attributes that are initialized. Note that each attribute must be initialized before it is used in proofs. Figure 2 represents the Petri Net that is derived from the

Flow Graph enhanced with constraints on environmental attributes. Several *performance properties* can be derived from the Petri Net. The number of input and output arcs connected to an action define the number of proofs and initializations respectively. As each token is mapped to a credential (see section 3.5), the maximal number of tokens is linear to the amount of storage space for users. The latter property can be derived by running Petri Net simulations. For other properties that can be derived from Petri Nets, see [8]. Among others, *liveness* shows how often actions can be performed, *persistence* investigates if enabled actions remain enabled after performing other actions, etc.

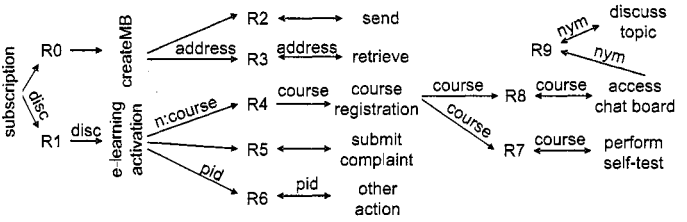

Fig. 2. Petri Net.

Linkability graph A linkability graph $LG_c = \{U, L\}$ is used to analyze the (un)linkability properties in the system. Each unit $u \in U$ is either an action, a right or unique environmental attribute. The set of entities to which the unit is revealed are assigned to each node. Each $l \in L$ defines a direct link between two units. Two properties[5] are assigned to each direct link, namely a set of additional entities (typically external/trusted entities) that are required to link both units and the conditions that must be fulfilled to link them.

A set of rules is defined to generate the LG_c from the Petri Net. The full lines in figure 3 denote the direct links[6] that can be derived from the Petri Net. Moreover, a set of queries are defined on LG_c. For instance, all paths between two given units that can be linked by any given subset of entities can be queried.

Modelling control measures In the next design step, control measures are modelled. To perform a control measure successfully, the system must be able to link the illicit action $action_{illicit}$ to another $unit_{target}$. To reveal the identity of a subject, the illicit

[5] Moreover, a cardinality ratio can be assigned to each link. It returns the maximal number of unit instances to which a given unit instance can be linked.

[6] For simplicity, the set of entities to which each unit is revealed is omitted in fig. 3.

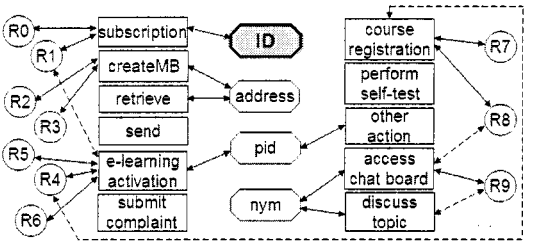

Fig. 3. Linkability graph.

action must be linked to any publicly identifying information *ID*. To deny a user to access a certain service, the illicit action must be linked to a right that is required to access the service. Frequently, units cannot be linked directly. The set of control measures that are possible in the current model can be derived from LG_c. For that purpose, LG_c is extended with conditional links that can be enforced using the following property that is defined on rights in section 2: *a deanonymizable proof can be linked to the right itself by a designated external entity if a condition is fulfilled.* Hence, if a subject sends an offensive message to a chat board, that unacceptable action can be linked conditionally to $R9$. As $R9$ is retrieved when accessing the chat board, $access\_chat\_board$ can be linked conditionally to $R8$, etc. The dotted lines in figure 3 denote the conditional links that can be enforced to take a countermeasure if some type of abuse occurs in action $discuss\_topic$. A right that is linked (in)directly to the illicit action in the extended linkability graph can be revoked. Similarly, the identity of a subject can be revealed if the illicit action can be linked to *ID*. Hence, the designer can get an overview of all possible control measures given a particular abuse.

Thereafter, the designer marks the desired control measures. All paths between the illicit action and the identity or the right that must be revoked can automatically be derived from the extended linkability graph. For instance, the following path can be used to reveal the identity of the subject after he has sent an offensive message (dead threat, blackmail, etc) to the chat board:

$$P_{sel} = [\, discuss\_topic, R9, access\_chat\_board, R8, course\_registration, R4, elearning\_activation, R1, subscription, ID]$$

If multiple paths exist to enforce a control measure, one path is selected. The selection strategy can depend on multiple factors: the path length (which is related to the amount of evidence that is stored), the trust level of each involved entity and the number of entities that are relied on, etc.

However, certain paths may also conflict with different types of requirements. As $Unlinkable(pid,ID)$ is a high priority requirement, selecting the path P_{sel} to enforce the accountability measure is unacceptable (see figure 3). A selected path may also clash with trust requirements if at least one entity that has to deliver evidence is not trusted by the system. Finally, the amount of evidence that is stored by selecting a path may be unacceptable.

Therefore, a set of alternative strategies (or patterns) are defined to avoid conflicts[7]. These strategies enhance the Petri Net with new transitions, places and/or arcs. For instance, the conflict discussed above can be avoided by introducing a new place $R10$ and three new arcs, namely $[subscription, R10]$, $[R10, course\_registration]$ and $[course\_registration, R10]$. Hence, an alternative path can be selected to enforce the accountability measure. Finally, the LG_c is updated with new (conditional) links based on the selected path.

3.4 Conflict avoidance versus conflict resolution

Multiple alternatives exist to model control measures. Hence, conflicts between control requirements and anonymity/trust requirements can partially be *avoided.* However, some conflicts cannot be resolved by simply applying another strategy. As new arcs may be introduced, the set of proofs per action may increase which may imply conflicts with performance requirements. If so, it might be possible to avoid conflicts by revising the control measures that were modelled before (i.e, the methodology can be enhanced with automatic backtracking). This means that acceptable alternatives for previous control measures are tried if no acceptable solution is available for the current control measure.

If the backtracking algorithm fails too, a strategy must be defined to *resolve* conflicts. Abandoning the current control measure is reasonable if the priority of the control measure is low.

Another strategy is to weaken/remove other requirements. For instance, the maximal acceptable amount of proofs may be increased for one or multiple actions. Similarly, certain unlinkability requirements can be omitted prior to rerunning the backtracking algorithm.

A more advanced strategy is to provide the designer (and/or stakeholders) with a set of alternatives. Although the backtracking algorithm fails to return an alternative without conflicts, it may successfully return models for which the number of conflicts and/or for which the sum of the priorities of the conflicting requirements is lower than a predefined number.

3.5 Implementation phase

Anonymous credentials are used as building blocks to realize the model. Idemix [3] is an anonymous credential system that helps to realize anonymous yet accountable transactions. In this section, some rules of thumb are summarized to generate credential primitives. We refer to [9] for a detailed overview of the realization of the conceptual model.

Each right $R \in \mathcal{R}$ corresponds to a credential type. Input/output arcs are transformed to credential show/credential issue protocols. Credential shows sometimes result in a transcript that can be deanonymized by a third party if a particular condition is fulfilled. Deanonymization reveals the nym on which the credential was issued. Moreover, application data can be signed during a credential show.

[7] [9] gives an overview of alternative strategies.

In addition, the paths that are assigned to control measures are mapped to chains of evidence. Each element in a chain defines an atomic piece of evidence at the implementation level that is required to perform the control measure.

3.6 Evaluation of the student environment

The optimized Petri Net for sample application is depicted in figure 4. A subset of properties that can be derived automatically from the models are summarized below.

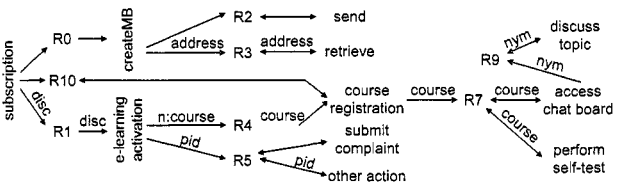

Fig. 4. Optimized Petri Net model.

Performance properties The maximum number of tokens (i.e, credentials) in $\mathcal{R}_x$={R0, R1, R2, R3, R5, R10} is 4. The maximum number of tokens in $\mathcal{R}_y$={R4, R7} is n (with n defines the number of courses). The space that is required to store one credential is slightly overestimated to 4 kB. Hence, all tokens that are required to go through activations, to use the mail services and use a subset of services within the e-learning environment (i.e, $\mathcal{R}_x$) can be stored on a smart card with 32 kB of rewritable memory. All services can be accessed from a smart card with 64 kB of rewritable memory assuming that the number of courses $n \leq 12$. Moreover, the processing time per action remains acceptable: only one proof is required for each type of action that can be performed unlimited.

Control measures The entities that are required to perform a control measure can automatically be derived from the linkability graph and is listed below. We used a single trusted deanonymizer D to reveal certain conditional links. However, trust can be distributed over a set of entities. The evidence that must be stored by each individual entity is omitted for simplicity.

	R	M	E	T	D
$(discuss\_topic, offensive) \rightarrow reveal(ID)$	✓			✓	✓
$(send, criminal) \rightarrow reveal(ID)$	✓	✓			✓
$(send, spam) \rightarrow denyPermission(send)$		✓			✓

Anonymity properties The entities that are required to link two given units and the conditions that must be fulfilled to link them can also be queried on the linkability graph. Among others, AR_1^h : $Unlinkable(pid,ID)$ is preserved unconditionally. The table below returns the link conditions between environmental attributes/actions that are used in UR_1^m, UR_2^m and UR_3^m (see section 3.2). For instance, criminal/spam mails can be linked to the users' mailbox address. However, only criminal mails (illegal contents, blackmail, etc) can be linked to the identity of the initiator.

	ID	**address**	**nym**	**send**
ID	–	$crimin.$	$offensive$	$crimin.$
address	$crimin.$	–	$crimin. \wedge offensive$	$crimin. \vee spam$
nym	$offensive$	$crimin. \wedge offensive$	–	$crimin. \wedge offensive$
send	$crimin.$	$crimin. \vee spam$	$crimin. \wedge offensive$	–

4 Discussion

Multi-paradigm modelling is introduced as a challenging approach for domain-specific modelling in [6] and has proven its feasibility in many fields. The advantage of using multiple formalisms in the design process of controlled anonymous applications is twofold. First, vertical multi-modelling (FG versus Petri Net) allows to model at different levels of abstraction. Second, horizontal multi-modelling (Petri Net versus LG_c) allows to derive models that allow for analysis and evaluation. Moreover, $Atom3$ [5] provided a powerful tool to define formalisms and generate models within the predefined formalism.

The methodology allows to define multiple types of access requirements and control measures. Moreover, it provides powerful evaluation mechanisms and alternatives. Therefore, it may certainly ease the design other applications with interactions between users and service providers such as business and e-government environments. However, future research is needed to extend the methodology to environments (such as P2P systems) with dynamic access policies and variable service providers. In this paper, only a subset of the methodology is presented. First, the actions that are defined in the sample application are static blocks. However, or-splits/or-joins constructs [8] allow to assign conditions on initializers/proofs. For instance, a right to retrieve a driver's license is only granted if the subject passed the practical exam successfully. Second, all environmental attributes in the sample application are constants. However, the methodology also allows to enhance the application with environmental variables. Mutators are assigned to output arcs. They specify how the value of these variables is updated. For instance, the amount of available disk space decreases/increases if the subject stores/removes files, the reputation level of a participant in an auction system may change after each transaction, etc.

5 Related work

The methodology is complementary with existing design/evaluation tools. K. Irwin and T. Yu [7] introduce a formal framework that reasons about the acceptability of attribute

based access control policies with respect to identifiability and information sensitivity. The identifiability is the property of how specifically an attacker can narrow down the identity of a user given the properties that he has disclosed. The sensitivity represents the impact of revealing information. Additional parameters can be considered to decide about the acceptability of a certain access policy such as asymmetric attributes, cross-attribute predicates, benefit analysis, etc. However, these parameters are often difficult to quantify.

E. Van Herreweghen [13] shows how various service providers' behavior can be made verifiable and how trust of users and service providers in the correct operation of other service providers could be minimized by defining appropriate liabilities and punishments in service providers' certificates. The liabilities specify the obligations towards service providers to reveal information depending on business agreements and contracts.

A. Pashalidis and C. Mitchell [10] consider timing attacks that may be launched by colluding organizations who wish to link actions from the same subject and propose solutions to tackle those attacks.

Graph-based models [4, 14] already exist for anonymity/unlinkability analysis. However, there are some important differences. First, existing models do not consider conditional links. Although omitting this feature is feasible to analyze anonymity properties in unconditional anonymous applications, evaluating conditional links becomes extremely important in controlled anonymous applications. Second, the current models analyze the anonymity properties towards one single entity (i.e, attacker/profiler). Our approach is more flexible as multiple entities are considered. On the contrary, some models support probabilistic links. Note that additional rules can be defined to add probabilistic links to LG_c. However, estimating reliable probabilities is often very complex and depends on many factors: the setting in which the system is used (i.e, the number of participants, etc), the semantics of application data (i.e, the message contents, etc), etc. Hence, those models are more appropriate for profiling purposes whereas LG_c is a useful tool to analyze unlinkability properties at the design stage of controlled anonymous applications.

6 Conclusion

In this paper, we presented a methodology for designing applications with two opposite types of requirements: privacy and control requirements. Using multiple models allows to evaluate anonymity properties, trust properties and performance properties. Several alternatives are defined at each design step that partially avoid conflicts between requirements. Moreover, fair conflict resolution strategies are defined. The final model foresees an easy mapping to control mechanisms. The paper also discusses how the methodology can be combined with other tools to improve certain properties. However, future research is required to apply the methodology to settings with dynamic access control policies and variable service providers.

References

1. E. Bangerter, J. Camenisch, and A. Lysyanskaya. A Cryptographic Framework for the Controlled Release Of Certified Data. In *Twelfth International Workshop on Security Protocols*, 2004.
2. S. Brands. Rethinking Public Key Infrastructure and Digital Certificates Building in Privacy. PhD thesis, Eindhoven Institute of Technology, 1999.
3. Jan Camenisch, Els Van Herreweghen. Design and Implementation of the Idemix Anonymous Credential System. Research Report RZ 3419, IBM Research Division, June 2002. Also appeared in *ACM Computer and Communication Security*, 2002.
4. D. Cvrcek and V. Matyas. On the role of contextual information for privacy attacks and classification. In *Privacy and Security Aspects of Data Mining workshop*, IEEE ICDM, Brighton, UK, 1 November 2004.
5. J. de Lara, H. Vangheluwe, and M. Alfonseca. Meta-modelling and graph grammars for multi-paradigm modelling in AToM3. In *Software and Systems Modeling (SoSyM)*, 3(3): pages 194–209, August 2004.
6. J. de Lara and H. Vangheluwe. Model-Based Development: Meta-Modelling, Transformation and Verification. The Idea Group Inc., 2005. http://www.cs.mcgill.ca/ hv/publications/04.OOmanagement.pdf
7. K. Irwin and T. Yu. An identifiability-based access control model for privacy protection in open systems. In *The Electronic Society archive Proceedings of the 2004 ACM workshop on Privacy in the electronic society*. Washington DC, p. 43–51.
8. T. Murata. Petri Nets: Properties, Analysis amd Applications. In *Proceedings of the IEEE*, Vol 77(4). pp.: 541–579.
9. V. Naessens and B. De Decker. Design patterns for modelling controlled anonymous applications. DistriNet Report, Dept. of Computer Science, K.U.Leuven, 2005.
10. A. Pashalidis and C. J. Mitchell. Limits to anonymity when using credentials. In *Proceedings of the 12th International Workshop on Security Protocols*, Cambridge, UK, Springer-Verlag LNCS, April 2004.
11. A. Pfitzmann and M. Kohntopp. Anonymity, unobservability and pseudonymity: a proposal for terminology. In *Designing Privacy Enhancing Technologies: Proceedings of the International Workshop on the Design Issues in Anonymity and Observability*, LNCS 2009, pages 1–9. Springer-Verlag, 2000.
12. W.B. Teeuw, H. van den Berg. On the Quality of Conceptual Models. In *Proceedings of the ER'97 Workshop on Behavioral Models and Design Transformations: Issues and Opportunities in Conceptual Modeling.*
13. E. Van Herreweghen. A Risk Driven Approach to Designing Privacy Enhanced Secure Applications. In *Proceedings of the 19th IFIP International Information Security Conference(SEC2004) - Embedded Workshop Privacy and Anonymity in Networked and Distributed Systems (I-NetSec'04)*, August 2004.
14. A. Zugenmaier, M. Kreutzer, and G. Muller. The Freiburg Privacy Diamond: An attacker model for a mobile computing environment. In *Kommunikation in Verteilten Systemen (KiVS) '03*, Leipzig, 2003.