Privacy in an Ambient World

M.A.C. Dekker[1], S. Etalle[2], and J. den Hartog[2]

[1] Security Group, TNO ICT, The Netherlands
marnix.dekker@tno.nl
[2] Department of Computer Science, University of Twente, The Netherlands
sandro.etalle@utwente.nl, jerry.denhartog@utwente.nl

Summary. Privacy is a prime concern in today's information society. To protect the privacy of individuals, enterprises must follow certain privacy practices, while collecting or processing personal data. In this chapter we look at the setting where an enterprise collects private data on its website, processes it inside the enterprise and shares it with partner enterprises. In particular, we analyse three different privacy systems that can be used in the different stages of this lifecycle. One of them is the Audit Logic, recently introduced, which can be used to keep data private when it travels across enterprise boundaries. We conclude with an analysis of the features and shortcomings of these systems.

1 Introduction

The last decades, people have started to use network services in their everyday lives. For example, in most countries there is now a widespread use of internet services, such as online stores, online forums etcetera. To be able to use these services, users often have to reveal privacy sensitive data to the enterprise operating the services. This data is often needed to provide (a better) service or for other purposes; however, once the private data has been disclosed, the enterprise could also misuse it, e.g. by trading it to marketing agencies. To prevent this, nowadays there exist laws demanding the enterprises to comply with precise privacy practices [1, 2].

For instance, the European Union in 1995 issued a directive to its member states that regulates the collection, storage and processing of personal data [1]. In 2002 this directive was extended to adapt to the ongoing changes in the electronic communications sector [2]. With these directives, the EU affirms the importance of privacy and the importance of aligning the privacy laws of the different EU member states, as exchanging private data across borders would be problematic, if countries had different privacy laws [2]. Among other things, the directives demand that enterprises only collect private data for specified, explicit and legitimate purposes and that the data may not be processed in

ways incompatible with those purposes [1]. To see an example of how the EU directives translate to requirements for computer systems, consider the setting of an online store. On the checkout page, the store requests the credit card number and the home address of users making a purchase. In addition, it asks users if their address can be given to partners, for the purpose of promotional mailings. For instance, the store sells airplane tickets, while the partners offer hotel rooms and cars for rental. The first requirement, that follows from the directive, is that the checkout page contains explicit statements about the purposes for which the credit card data and the home address data are collected. The second requirement is that the enterprise's web server, other systems in the backend and systems at the partner sites, must not process the data for purposes other than those stated on the checkout page.

In this setting, the lifecycle of the private data consists of three stages: The first stage is the moment of collection by the enterprise. The second stage is the processing inside the enterprise, while the third stage is the processing outside the enterprise, at the partner sites. In this chapter we illustrate three (complementary) privacy systems, each of which can be used in one of the these stages: Surveys [3] show that, for websites, P3P [4] is the most widely used system for the expression of the purposes for which private data is collected. Therefore, in Section 2, we analyse P3P and we give an example of how it is used in practice. Secondly, in Section 3, we analyse E-P3P [5]. E-P3P is basically the only system which was designed precisely to address the problem of ensuring that inside the enterprise private data is used for the right purposes (see for related work Section 5). Finally, in Section 4 we analyse the Audit Logic, a system introduced by recently [6], that can be used for the protection of private data across enterprise boundaries.

2 Privacy Statements

Web sites often ask users to disclose their private data, like name, address, email address etcetera; this information may be needed by the webservice to provide a better service, though it could also be used for other unwanted purposes. This raises the need to inform the user about how his personal data is being treated: e.g. who will see it, for how long it will be stored, and for which purposes it is going to be used. Actually, in many countries, websites *have* to provide a *privacy statement* explaining how personal data is used [1, 7].

However, privacy statements are often too long and detailed to be understood by the ordinary internet user. P3P - which was introduced in 1997 by the W3C, but only became an official recommendation in 2002 - is devised to solve this problem by supporting automatic analysis of privacy statements. P3P is now used by many popular websites [3].

P3P allows enterprises to translate their privacy statements into a standardized XML-based format, using a common vocabulary, to be placed on

their websites [4]. When a website supports P3P, a visitor can employ an automatic tool to analyze the website's privacy statement and quickly decide if they are satisfactory.

To illustrate how it works, let us see an example.

Example 1. Claudia visits an online store and after choosing a product she goes to the checkout page. Here she fills out a form with some private data: i.e. her name and credit card number. The store states in a privacy statement that it will use this data only to complete the transaction. In addition, the checkout form has a non-obligatory field for the customers email address. The store states (in a second privacy statement) that this information will be used for promotional mailings. Both privacy statements can be translated in P3P. The resulting policy is shown in Figure 1.

```
<POLICIES xmlns="http://www.w3.org/2000/P3Pv1">
 <POLICY name="checkout"
         entity="Store, 5th Avenue, Manhattan, PO 10001, USA">
 <DISPUTES>service="PrivacySeal.orG/DisputeResolution"</DISPUTES>
 <ACCESS><none/></ACCESS>
  <STATEMENT>
   <PURPOSE><current/></PURPOSE>
   <RECIPIENT><ours/></RECIPIENT>
   <RETENTION><stated-purpose/></RETENTION>
   <DATA-GROUP>
    <DATA ref="#user.name"/>
    <DATA ref="#dynamic.miscdata"/></DATA-GROUP>
  </STATEMENT>
  <STATEMENT>
   <PURPOSE><contact required="opt-in"/></PURPOSE>
   <RECIPIENT><ours/></RECIPIENT>
   <RETENTION><stated-purpose/></RETENTION>
   <DATA-GROUP>
    <DATA ref="#online.email"/></DATA></DATA-GROUP>
  </STATEMENT></POLICY></POLICIES>
```

Fig. 1. A sample P3P policy

This allows us to see the elements of a P3P policy: In the first place, the *entity* indicates the issuer of the policy. Secondly, the *disputes* element describes how possible conflicts over the privacy policy may be resolved (e.g., by which court, or other entity). This is not binding, in the sense that the enterprise is still subject to the legal ways to resolving a privacy dispute. The *access* element indicates whether the submitted data may be accessed by the subject after it has been collected. This can be used for instance to verify the accuracy of the collected data. This policy states that access is not possible. Finally, the key elements of the P3P policy are the *statements* which describe,

per data item, for which *purpose* it is collected, who is allowed to access it (in the *recipient* element) and for how long it will be stored (in the *retention* element). In the figure the purposes are respectively *current*, which refers to the online purchase and *contact*, which indicates that the information can be used to contact the user for "marketing of services or products". The purpose element may also contain an attribute indicating how a user can express his consent to the purpose. In this case, explicit *opt-in* is required for the purpose contact. The recipient value *ours* means that the data can only be accessed by the store (e.g. it will not be given to third parties), while the retention value *stated-purpose* means that the data will only be retained for a period needed for the stated purpose. The *data* element is specified by a reference to an element in a so-called P3P data schema, e.g. `#online.email`. The data-schema defines the format and the meaning of the data-elements that may occur in a P3P policy. In the example, by not specifying a data-schema, we use P3P's default data-schema.

Going back to our example, if Claudia's browser supports P3P, it can compare the above policy with Claudia's privacy preferences. One of these preferences states that she wants to be warned when sites request information for purposes other than *current*. In this case the browser, can notify her that she *may or may not supply her email address for marketing of services or products*. Her advantage is that she does not have to read the site's privacy statement to find out what they mean and which fields are optional.

Since its introduction in 1997, P3P has received considerable attention [7]. Its deployment was particularly stimulated by the introduction in 2001 of a privacy slider in Microsoft's Internet Explorer 6. This privacy slider allows the user to determine which websites may set and retrieve *cookies*, according to their P3P policies. Cookies from websites with no P3P policies (or with an unsatisfactory one) are blocked by the browser.

A drawback of P3P is that, despite its simplicity, P3P policies can be ambiguous [8]. For instance, one could refer to the same data element twice with different retention periods, within the same policy. Ambiguities may result in legal risks for the issuers as their policies may be interpreted in unexpected ways [9]. This also makes the development of P3P compliant browsers more difficult. As a matter of fact, despite the fact that P3P was designed to be interpreted by browsers, there is no definition of how a browser should interpret policies, and there are no guidelines for writing 'browser-friendly' policies [9].

Finally, we should mention that while P3P addresses the problem of representing a website's privacy policy, it does not address the problem of enforcing them. The use of P3P alone does not give assurance about the *actual* privacy practices in the backend of the website. Critics have even suggested that the online industry, by adopting P3P, is only giving an appearance of protecting privacy, to avoid stricter legislation [10].

3 Enterprise Privacy

As mentioned earlier, in many countries, legislation regulates the collection and the use of private data. This requires enterprises to enforce privacy policies that prescribe for example when certain data should be deleted, by whom it may be accessed and for which purposes. As we saw in the previous section, P3P can be used to represent such privacy policies on websites, but it does not address the problem of enforcing them inside the enterprise. The Platform for Enterprise Privacy Practices (E-P3P) - introduced in 2002 by Karjoth et al. - addresses exactly this problem [5]. E-P3P provides an XML-based language to express privacy policies as well as a framework with specific privacy functionality to enforce these policies. Before giving a practical example of how E-P3P works, we give an overview of the main components of the E-P3P system.

In the E-P3P architecture an enterprise collects private data at so-called *collection points*. Here individuals, e.g. customers, submit private data to the enterprise, after agreeing with the enterprise's privacy statements. Each collection point has a *form* which associates the private data with its subject, declares its *type*, e.g. medical record or postal address, and the subject's *consent choices*. This association remains intact in the enterprise's backend and it may even travel to another enterprise. In E-P3P this is called the *sticky policy paradigm* [5]. The sticky policy does not refer to enterprise policies but refers to the privacy statements and the filled in consent choices on the data-collection form that stick to the private data.

The privacy officer of the enterprise declares, by using E-P3P's policy language, the privacy policies by specifying who can access which type of data for which purposes. The privacy policy can also refer to the subject's consent choices and to certain privacy obligations, e.g. *delete the data in 30 days*. Operations in the enterprise's legacy applications are then mapped to terminology used in the privacy policies, and, in the reverse direction, privacy obligations used in the privacy policies are mapped to operations in the legacy applications. For example, the 'send' operation of a mass-mailer system, used in the marketing department, is mapped to the term *read for the purpose of marketing* in the privacy policy. Conversely, the term *delete the subject's email* in the privacy policy is implemented as an 'unsubscribe' operation of a mailing list system.

Finally, access to the private data of a subject is granted in two steps. The access to the legacy enterprise application is evaluated by an access control system, for instance taking into account employee roles, which is independent of the E-P3P system. Then, the legacy application makes an access request to a *privacy enforcement system* for the subject's private data. The privacy enforcement system decides wether access should be granted, by evaluating the enterprise policy and by matching against the subject's consent choices. If access is granted, then the privacy enforcement system also executes possible privacy obligations specified in the enterprise policy.

Example 2. Consider the previous example of an online store collecting private data on its checkout page. The enterprise that owns the online store wants customers to trust its privacy practices. To this end, it has published privacy statements on the checkout page and uses E-P3P to ensure that enterprise systems behave according to them.

These privacy statements specify that Claudia's name and credit card number may be accessed by the employees from the Billing department provided that the purpose is 'billing' and that the data is subsequently deleted. In addition, employees may use Claudia's email address for marketing purposes, if Claudia opted in to this purpose. The corresponding E-P3P policy is shown in Figure 2.

```
<ep3pPolicy
 version = '1.2'
 issuer = 'Store'
 vocabulary-ref = 'http://www.Store.com/Voc'
 default-ruling='deny'>
  <rule>
   <dataCategory>allData.creditCardData</dataCategory>
   <purpose>business.billing</purpose>
   <userCategory>employees.billing</userCategory>
   <ruling>ALLOW</ruling>
   <action>read</action>
   <obligation action=deleteWithIn(30)</obligation></rule>
   <condition/>
  <rule>
   <dataCategory>allData.contactData</dataCategory>
   <purpose>business.marketing</purpose>
   <userCategory>employees</userCategory>
   <ruling>ALLOW</ruling>
   <action>read</action>
   <obligation\>
   <condition>OptInToMarketing=True</condition>
  </ruleset>
</ep3pPolicy>
```

Fig. 2. A sample E-P3P policy

Now suppose that an employee of the marketing department wants to send an email with promotions to a number of customers (including Claudia), by using a mass-mailing system. The mass-mailing system, after checking that the employee is authorized to use the system, sends a request to the E-P3P privacy enforcement system to see whether access should be allowed on the basis of the enterprise's privacy policy. The request has the following form:

```
<ep3pQuery>
```

```
  <userCategory>employees</userCategory>
  <dataCategory>allData.contactData</dataCategory>
  <purpose>business.marketing</purpose>
  <action>Read</action>
</ep3pQuery>
```

The request is *matched* against the E-P3P policy by the policy enforcement engine. The policy prescribes to check wether Claudia gave consent to receiving promotional mailings, in which case the privacy enforcement system grants the request, `allow`, otherwise, it will reject the request `deny`, which is also the default value.

This example shows the key elements of an E-P3P policy: a reference to the vocabulary used, the policy's default-ruling and the policy's ruleset. The ruleset is a list of E-P3P rules that declares which user categories can perform which actions on which data categories and for which purposes. The vocabulary allows one to define *hierarchies* of data categories, purposes, and data users, which are convenient to refine a privacy policy in a hierarchical sense. For example, the allow ruling inherits downwards in the hierarchies: When a rule allows a request for 'allData', then a request for 'allData.creditCardData' is also allowed. Denials, on the other hand, are inherited both downward and upward, for example if a rule denies access to allData.creditCardData, then the requests for allData or allData.creditCardData.cardType are also denied.

E-P3P is introduced by Karjoth et al. [5], while the full XML-based language and semantics for E-P3P policies was defined by Ashley et al. [11]. EPAL [12], a language very similar to (and derived from) E-P3P, was submitted by IBM to the W3C for standardization, but at the time of writing it has not been endorsed. IBM has also implemented EPAL in the *IBM Tivoli Privacy Manager*, a system delivering automatic management of private data to bring down the enterprise's costs of privacy management and to decrease the risks of unauthorized disclosures.

As we mentioned, E-P3P also allows data to be moved from one enterprise to another, together with the form that was used to collect it; the sticky policy paradigm. This way, the destination enterprise receives private data with a privacy policy, the enforcement of which might require the composition of policies or checking that one policy is a *refinement* of the other. The precise definition of the composition and refinement operations for E-P3P policies is given by Backes et al. [13].

It is worth remarking that, although the names of E-P3P and P3P are very similar, they are used in different settings. One is used to manage privacy rules internal to an enterprise, the other is used to communicate, in a standardized way, privacy policies to internet users. To link these two aspects, Karjoth et al. [14] propose to generate and publish P3P policies directly from internal enterprise privacy policies and to update them regularly to reflect the enterprise's current practices. Yu et al. [8] on the other hand argue that P3P

policies should be more long-term promises, which should not change each time an internal business rule is updated.

4 Audit Logic

Where P3P and E-P3P offer methods for specifying privacy policies and enforcing these policies within an organization, the issue of how to protect privacy when data can be modified and/or travels across different companies remains open. In this section we describe the Audit Logic [6], which provides an alternative approach to privacy protection. The Audit Logic addresses the issue of compliance to policies for data which moves across different security domains.

Example 3. Company A and B are member of a federation that shares customer contact information for the purpose of marketing. The federation rules require that the companies build audit trails for their commercial mailings, which may be checked by an independent authority.

When company A collects information from clients it also asks for permission to provide this information to its partners, e.g. through a *Do you want to receive offers from our partners?*-checkbox on a webform. When this box is selected the email address is shared with company B, which is given permission to send one email a month regarding its offers.

After the contact information is provided to company B, company A can no longer control the use of this data; even if both company A and B are using P3P and/or E-P3P, A cannot ensure that the data is used according to its privacy policy. A method is needed which will allow company A to place a privacy policy on the data it provides to company B and will give A confidence that this policy will be adhered to.

When data leaves the security domain, Access Control [15, 16, 17] is not sufficient for protecting the data; as the control over the access moves with the data. Digital rights management (DRM) techniques [18, 19] on the other hand are designed to ensure policy compliance for data which moves across security domains. Licenses and keys are needed to access the data and describe the policies for this data. While the data centric approach of licenses is useable for providing the privacy policies, DRM techniques are often not flexible enough or have requirements, such as the need for special (trusted) hardware, which are not realistic in our corporate collaboration scenario. For a privacy protection mechanism to be viable it should not unduly increase the costs or required effort for the companies involved.

The Audit Logic applies data centric techniques in an auditing approach where compliance to policies is not enforced a-priori but instead actions of the users may need to be justified a-posteriori, i.e. users may be audited. By holding the users accountable for their actions, the Audit Logic approach

tries to deter rather than prevent policy violation. By using this approach the Audit Logic formalizes the audit trail which is already present in many companies and enables its use for privacy protection. In some cases it may be needed to protect the audit trail, for instance by using techniques from tamper-proof logging [20].

4.1 The Audit Logic Framework

The Audit Logic framework consists of agents executing actions, optionally logging these actions and being audited to check if their actions adhered to the relevant (privacy) policies. Actions can be e.g. sending an email to an address, reading and updating information in a medical file, but also providing a new policy to another agent. Figure 3 shows an example execution in the framework: In the first step (I), agent a provides a policy ϕ to agent b which b records in his log (II). Next (III) agent b reads document d which is stored in the company database. At a later point the auditing authority, which is checking access to privacy sensitive files, finds the access of b (IV) and requests b to justify this access (V). In response, b shows that the access was allowed according to the policy ϕ which was provided by a. The auditor, initially unaware of a's involvement, can now (VI) audit a for having provided the policy ϕ to b.

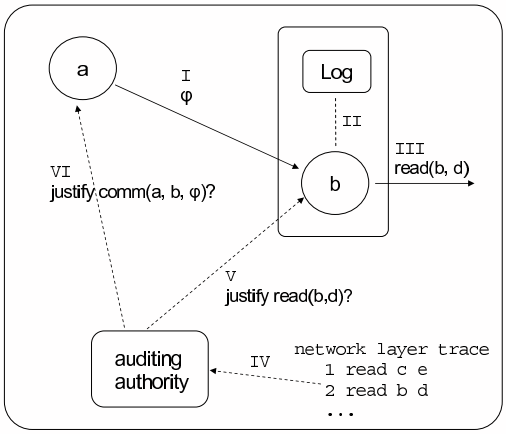

Fig. 3. Sample deployment depicting actions, the logging and interaction with an auditor.

The policy language

As illustrated by the example, policies needs to be able express permissions to execute actions. For example a policy may be `mayRead`(b, d) expressing the permission to execute the action `read`(b, d). Besides expressing basic facts

and permissions, the policy language of the audit logic allows combination of permissions and adding requirements or obligations: e.g. $\texttt{mayRead}(b, d) \wedge \texttt{mayWrite}(b, d)$ expresses that b may both access and update document d and $\texttt{isSysAdmin}(b) \rightarrow \texttt{mayWrite}(b, d)$ expresses that b may update document d provided he is a system administrator.

The constructions mentioned above provide an expressive language for writing privacy policies for a given agent. To enable agents to provide permissions to other agents, an additional construct "*says to*" is provided: This construct is used to describe delegation rights: e.g. a `says` $\texttt{mayRead}(b, d)$ `to` b expresses that a is allowed to give the policy $\texttt{mayRead}(b, d)$ to b.

Logged actions

When b receives the policy from a, b decides to store this policy in his log. The log is assumed to be secure and only able to store actions that actually happened and only when they happen. For the sending of policies this reflects the assumption that communications are non-refutable; b will be able to prove that a sent the policy.

Deriving permissions

When an agent wants to execute an action, the decision has to be made whether the policies allow this action. The policy framework uses a logical derivation system to decide whether a given set of policies, facts and obligations is sufficient to obtain a given permission. E.g. if b is a system administrator and got the permission $\texttt{isSysAdmin}(b) \rightarrow \texttt{mayRead}(b, d)$ then b has the permission to read document d: $\texttt{isSysAdmin}(b), a$ `says` $\texttt{isSysAdmin}(b) \rightarrow \texttt{mayRead}(b, d)$ `to` $b \vdash_b \texttt{mayRead}(b, d)$. Using the derivation system, agents can build a *compliance proof*, i.e. a formal deduction in the derivation system that shows that an action was allowed by the policies. Compliance proofs can be stored, communicated to the auditing authority and automatically checked.

Auditing

The auditing authority can ask users to justify actions that it observed. When audited the user needs to provide a compliance proof for each of these actions. The auditing process can be done effectively; the user should already have built a proof before executing the action and the audit authority only needs to check the correctness of the proof which is relatively straight forward.

Note that the auditing authority may actually consist of different entities for the different companies; in this case an entity auditing one company relies on the other entities to audit actions outside of its authority.

4.2 Implications

In the Audit Logic misuse is not prevented but deterred: *auditing authorities* have a mandate of checking whether the data was used in compliance with

the policies. Hence users should be *auditable* and sufficient *audit trails* should be available to the auditors. This fits well with e.g. hospitals or companies, where users can be held accountable for their actions and audit trails are often already part of the (security) requirements. It may be hard to realize these requirements in certain settings, such as large open networks (e.g. P2P). Although this may be changing; for instance EU law demands that ISP's keep IP traffic records of all their users.

5 Related work

In this section we give an overview of the literature related to P3P, E-P3P and the Audit Logic.

An extensive survey of social, legal and technical aspects of P3P was given by Hochheiser [7]. In a more technical approach, Yu et al. [8] investigate the semantics of P3P: they find several inconsistencies and show how to restrict the language to avoid them. Byers et al. [3] survey the use of P3P on a large number of websites. They argue that a large number of websites is not compliant with the P3P specifications, and that this may yield legal problems for these websites. The P3P Preference Language (APPEL) [21] was developed by Cranor et al. to allow users to express preferences about P3P policies. With APPEL, users can specify which P3P policies they find acceptable and which not. Yu et al. [8] develop another kind of P3P preference language. This approach is based on the semantics of P3P, unlike APPEL, which is based on the syntax of P3P. Related to P3P is the Resource Description Framework (RDF) [22]. RDF is developed to represent information on the web in a machine-readable format. Although it is not specifically intended to be used for privacy practices, it may be used to express P3P policies. The RDF query language is considered to be an alternative approach to specifying user preferences about P3P policies [21].

E-P3P is an extension of Jajodia et al.'s Flexible Authorization Framework (FAF) [23]. Like in E-P3P, FAF provides a policy language that can specify both positive and negative authorizations and uses hierarchies for objects and users. However, FAF does not allow the use of obligations, and does not include a special construct to express the purpose of an access request. The notion of privacy obligations in E-P3P is similar to the provisions in Jajodia's Provisional Authorization Specification Language (pASL) [24]. Here a principal is granted access to an object if it causes certain conditions to be satisfied. In E-P3P, obligations are treated opaquely, as methods that are called and return a value, while in pASL obligations are treated more in detail by using a temporal logic. E-P3P shares some similarities with XACML [25], an OASIS standard for access control systems. XACML is XML-based and uses object and data hierarchies, as well as conditions and obligations. XACML is also inspired by FAF [23], and, although it is not specifically intended for enterprise privacy policies, it can be used for protecting private data inside an

enterprise. As an example of this a policy for the protection of medical records is shown [25]. Although XACML does not have a special *purpose* construct, like the one in E-P3P, it has been added in XACML's so-called privacy profile. Anderson [26] compares EPAL [12] and XACML and concludes that EPAL corresponds mostly to a subset of XACML and that it lacks certain features required for access control and privacy. Stufflebeam et al. [27] present a practical case study of E-P3P and P3P. Here the authors implement a number of health care policies in both EPAL and P3P. Among other things, they conclude that many promises expressed in natural language privacy policies are neither expressible in P3P nor enforceable with EPAL. More closely related to the Audit Logic, Originator Control (ORCON) policies [28] were introduced as an alternative for discretionary and mandatory policies. In mandatory access control, the receiver of a document can not change the access rights on the document, while in discretionary access control, the receiver of the document can always change the rights on it. In ORCON policies, the original owner of the data can always change the access rights on the data, while the current owner of the data can not do so. This fits well with the privacy regulations in which the subject should retain control over its personal data [1]. Also in the Audit Logic, the owner of data can always change the rights on the data, however in the Audit Logic those rights are not stored centrally but can be moved between systems in a completely distributed setting. The policy language of the Audit Logic is based on a formal logics. Abadi [29] surveys a number of different distributed access control models that are based on formal logics. In these models an authorization request or an authentication credential corresponds to a logical formula and its proof corresponds to the authorization or authentication decision. For example, PCA [30] is a system for the authorization of clients to webservers, by using distributed policies. The Audit Logic, like PCA, uses the fact that checking proofs is easy and places the burden of finding the proofs, which is typically harder, on the clients requesting access. PCA however uses a higher order (classical) logic, while the Audit Logic is restricted to first-order logic, rendering a more tractable proof search. A more common example of systems where clients compile part or all of the authorization proof is SDSI [31], which allows clients to 'chain' together certificates to prove their authenticity. The Audit Logic language is closely related to Delegation Logic [32] and Binder [29]. They also use the *says* predicate introduced by Abadi et al. [29], which however can not be nested inside another says, for instance to express K *says* (M *says* P). This restriction is absent in the Audit Logic. Also in the Audit Logic we use a refined form of the *says* predicate, by specifying also the target agent. Conrado et al. [19] propose to use DRM to enable privacy distributed systems and vice versa to use privacy as a driver for a wider use of DRM. Licensescript is a novel DRM language using Prolog code to encode more content licenses [18]. DRM however, unlike the Audit Logic, requires the use of special hardware, which may make it hard to implement in the enterprise's legacy systems.

6 Conclusions

We have analyzed three different privacy systems. We will now conclude with their main advantages and shortcomings.

In the P3P system, privacy statements are formatted using XML and a common vocabulary, to allow for automatic analysis of the statements. P3P is well-established, in the sense that there are many popular websites that use P3P [3]. Also there are a number of tools that generate natural language statements from P3P statements [7]. A drawback of P3P is that it does not distinguish between different types of access. For example, it is impossible to specify that certain employees may update personal data, while others may not. This makes it cumbersome to use for certain enterprise privacy policies.

The E-P3P system addresses this. E-P3P distinguishes between different types of access and enables the use of obligations and conditions. Although E-P3P itself is not an endorsed standard, it corresponds to a subset of an OASIS standard, i.e. the XACML access control language [25]. In a way they are complementary because E-P3P assumes the existence of access control policies, independent of the privacy policies. E-P3P policies can contain prohibitions, i.e. rules that deny access, which makes the language more expressive than the language used in the Audit Logic. However it seems complicated to move E-P3P policies from one enterprise to another. The new policy may cause conflicts and it may even be bypassed altogether due to other policies that are incompatible [13]. Moving policies may be needed in enterprise collaborations where private data is exchanged, guarded by policies. Furthermore, the use of E-P3P can only give assurances to other enterprises, when they assume that the enterprise is trusted to implement E-P3P correctly [5]. This may be a too strong assumption in the setting where enterprises dynamically form coalitions to exchange private data.

In the Audit Logic this assumption is relaxed. Here it is assumed that the enterprise can misbehave, while compliance to privacy policies can be verified by (external) auditors, through a formal auditing procedure. The Audit Logic is designed for a distributed setting, and it is easy to move policies across enterprise domains for instance accompanying private data. However, when policies are sent from one enterprise to another, the question is raised wether one can trust the sender of the policy. For example, a rogue enterprise could be set up for the purpose of distributing false privacy policies to the other enterprises. To solve this problem one could extend the Audit Logic with a trust management system to facilitate trust decisions about the sources of policies. Furthermore, it may be interesting to couple the reputation of enterprises to the outcome of past audits, like in reputation-based systems [33]. Finally, the Audit Logic uses formal (first-order) logic to express policies and lacks a tool that translates policies to natural language, like those for for P3P. Such a translation to natural language is important to improve the useability of policies based on formal logic [34].

References

1. The European Parliament and Council: The data protection directive (95/46/EC) (1995)
2. The European Parliament and Council: Directive on privacy and electronic communications (2002/58/EC) (2002)
3. Byers, S., Cranor, L.F., Kormann, D.: Automated analysis of P3P-enabled web sites. In: Proc. International Conference on Electronic Commerce (ICEC). (2003) 326–338
4. Cranor, L., Langheinrich, M., Marchiori, M., Presler-Marshall, M., Reagle, J.: The Platform for Privacy Preferences 1.0 (P3P 1.0) specification - W3C recommendation 16 april 2002. http://w3.org/TR/P3P (2002)
5. Karjoth, G., Schunter, M., Waidner, M.: Platform for enterprise privacy practices: Privacy-enabled management of customer data. In Dingledine, R., Syverson, P.F., eds.: Proc. International Workshop on Privacy Enhancing Technologies (PET). Lectures in Computer Science, Springer (2002) 69–84
6. Cederquist, J.G., Corin, R., Dekker, M.A.C., Etalle, S., den Hartog, J.I.: An audit logic for accountability. In Winsborough, W., Sahai, A., eds.: Proc. International Workshop on Policies for Distributed Systems and Networks (POLICY), IEEE Computer Society Press (2005) 34–43
7. Hochheiser, H.: The platform for privacy preference as a social protocol: An examination within the U.S. policy context. ACM Transactions on Internet Technology (TOIT) **2**(4) (2002) 276–306
8. Yu, T., Li, N., Antón, A.I.: A formal semantics for P3P. In: Proc. Workshop On Secure Web Service (SWS), ACM Press (2004) 1–8
9. Schunter, M., Herreweghen, E.V., Waidner, M.: Expressive Privacy promises - how to improve P3P. Position paper for W3C Workshop on the Future of P3P (2002)
10. Cattlet, J.: Open letter to P3P developers. `http://junkbusters.com/standards.html` (1999)
11. Ashley, P., Hada, S., Karjoth, G., Schunter, M.: E-P3P privacy policies and privacy authorization. In Samarati, P., ed.: Proc. Workshop on Privacy in the Electronic Society (WPES), ACM Press (2002) 103–109
12. Ashley, P., Hada, S., Karjoth, G., Schunter, M.: (Enterprise privacy authorization language (EPAL 1.2) - W3C member submission 10 november 2003)
13. Backes, M., Pfitzmann, B., Schunter, M.: A toolkit for managing enterprise privacy policies. In Gollmann, D., Snekkenes, E., eds.: Proc. European Symposium on Research in Computer Security (ESORICS), Springer (2003) 162–180
14. Karjoth, G., Schunter, M., Herreweghen, E.V.: Translating privacy practices into privacy promises -how to promise what you can keep. In: Proc. International Workshop on Policies for Distributed Systems and Networks (POLICY), IEEE Computer Society Press (2003) 135–146
15. Jajodia, S., Samarati, P., Subrahmanian, V.S., Bertino, E.: A unified framework for enforcing multiple access control policies. In Peckham, J., ed.: Proc. International Conference on Management of Data (SIGMOD), ACM Press (1997) 474–485
16. Park, J., Sandhu, R.: Towards usage control models: Beyond traditional access control. In Bertino, E., ed.: Proc. Symposium on Access Control Models and Technologies (SACMAT), ACM Press (2002) 57–64

17. Sandhu, R., Samarati, P.: Access control: Principles and practice. IEEE Communications Magazine **32**(9) (1994) 40–48
18. Chong, C.N., Corin, R., Etalle, S., Hartel, P.H., Jonker, W., Law, Y.W.: LicenseScript: A novel digital rights language and its semantics. In Ng, K., Busch, C., Nesi, P., eds.: Proc. International Conference on Web Delivering of Music (WEDELMUSIC), IEEE Computer Society Press (2003) 122–129
19. Conrado, C., Petkovic, M., van der Veen, M., van der Velde, W.: Controlled sharing of personal content using digital rights management. In Fernández-Medina, E., Hernández, J.C., García, L.J., eds.: Proc. International Workshop On Security in Information Systems (WOSIS). (2005) 173–185
20. Chong, C.N., Peng, Z., Hartel, P.H.: Secure audit logging with tamper-resistant hardware. In Gritzalis, D., di Vimercati, S.D.C., Samarati, P., Katsikas, S.K., eds.: IFIP International Conference on Information Security and Privacy in the Age of Uncertainty (SEC), Springer (2003) 73–84
21. Cranor, L., Langheinrich, M., Marchiori, M.: A P3P preference exchange language 1.0 (APPEL 1.0) (2002)
22. Lassila, O., Swick, R.R.: Resource Description Framework (RDF) Model and Syntax Specification - W3C Recommendation 22 February 1999 (2002)
23. Jajodia, S., Samarati, P., Sapino, M.L., Subrahmanian, V.S.: Flexible support for multiple access control policies. ACM Transactions on Database Systems **26**(2) (2001) 214–260
24. Jajodia, S., Kudo, M., Subrahmanian, S.: Provisional authorization. (In: Proc. 1st International Workshop on Security and Privacy in E-Commerce (WSPEC))
25. OASIS Access Control TC: eXtensible Access Control Markup Language (XACML) Version 2.0 - Oasis Standard, 1 Feb 2005 (2005)
26. Anderson, A.: Comparison of two privacy languages: EPAL and XACML. Sun Technical Report TR-2005-147 (2005)
27. Stufflebeam, W.H., Antón, A.I., He, Q., Jain, N.: Specifying privacy policies with P3P and EPAL: lessons learned. In: Proc. Workshop on Privacy in the Electronic Society (WPES). (2004) 35
28. Park, J., Sandhu, R.: Originator control in usage control. In: Proc. International Workshop on Policies for Distributed Systems and Networks (POLICY), Washington, DC, USA, IEEE Computer Society (2002) 60
29. Abadi, M.: Logic in access control. In Kolaitis, P.G., ed.: Proc. Symposium on Logic in Computer Science (LICS), IEEE Computer Society Press (2003) 228–233
30. Appel, A.W., Felten, E.W.: Proof-carrying authentication. In Tsudik, G., ed.: Proc. Conference on Computer and Communications Security (CCS), ACM Press (1999) 52–62
31. Rivest, R.L., Lampson, B.: SDSI – A simple distributed security infrastructure. Presented at CRYPTO'96 Rumpsession (1996)
32. Li, N., Grosof, B.N., Feigenbaum, J.: Delegation logic: A logic-based approach to distributed authorization. ACM Transactions on Information and System Security (TISSEC) **6**(1) (2003) 128–171
33. Shmatikov, V., Talcott, C.L.: Reputation-based trust management. Journal of Computer Security **13**(1) (2005) 167–190
34. Halpern, J.Y., Weissman, V.: Using first-order logic to reason about policies. In Focardi, R., ed.: Proc. Computer Security Foundations Workshop (CSFW), IEEE Computer Society Press (2003) 187–201