13

Privacy Models and Languages: Assurance Checking Policies

Siani Pearson

HP Labs

Assurance policies express the security and data protection processes and mechanisms which should be in place to protect users' data. Users define such policies to express the minimum privacy protection which they wish to have in place by the recipient of their data. Service providers publish their policies to assert protection which should be in place, and for which they are able to provide evidence that this is indeed the case. Thus, assurance policies may be regarded as a specialised form of release and data handling policy, depending upon the context.

In this section we explain the motivation for using assurance policies, and show some formalisms used within PRIME.

13.1 Introduction

Assurance checking policies are formulated by people to obtain degrees of assurance from enterprises that their data will be processed according to their expectations, such as compliance to privacy, security and IT standards. In many cases, the user is specifying the type of device and environment where their PII data is being viewed. These would usually be checked up-front before PII was released, either in the preamble or in a negotiation phase before release of PII.

Assurance checking policies are separate from obligations and are constraints and conditions usually expressed by people before they engage with enterprises. They might just specify a particular regulatory context or, more generally, can require enterprises to provide degrees of proof about their ability to:

- Support the enforcement of predefined privacy policies and obligations with respect to laws and legislation

J. Camenisch, R. Leenes, and D. Sommer (Eds.): Digital Privacy, LNCS 6545, pp. 363–375, 2011.

- Run their processes, services and data repositories in a secure way
- Use secure and trusted systems, such as trusted computing platforms, to increase the level of security and trust in their operational activities.

Assurance checking policies may also be expressed by the services side, to obtain degrees of assurance from third parties that any data shared with these third parties will be processed according to the service provider's expectations, and also to express how data provided to the service provider by users will be processed, and to provide evidence that this is actually the case.

In summary, the overall motivation for defining assurance policies is to allow people to make judgements about the trustworthiness and privacy compliance of the remote receiver of their PII data. For example, a user might check for the compliance of an organisation against customised preferences prior to disclosure of PII data. Their disclosure of PII data or the continuation of a business interaction could be subject to the outcome of this checking.

13.1.1 Principles

The principles underlying the use of assurance policies are as follows:

- The assessment goes beyond just promises on behalf of the service provider
- It assesses the levels of proof that can be provided about privacy-providing mechanisms that are used and even how they are operating
- A broader range of information and assurance is assessed than just security information
- In some cases assurance policies can be checked independently of access control
- Broadly speaking, the conditions checked are necessary, not sufficient for the transaction to proceed
- Ultimately, it is the user who decides how to proceed

13.1.2 Natural Language Examples

The following natural language expressions give examples of the type of constraints people may want to express using assurance policies:

- "the processing platforms must give a high level of protection for my PII data"
- "the back end must have a valid privacy seal issued by a provider I (or some authority I delegate) trusts"
- "the back end must give tamper-resistant protection to secrets"
- "the service provider should support obligation management"
- "my PII will only be processed within EU"

13.1.3 Overview of Different Potential Approaches

There are a range of different approaches to defining assurance policies. We shall consider two approaches, both of which we considered within PRIME:

1. Trust and assurance constraints can be represented as first order predicate logic expressions: for example, `hasValidPrivacySeal(Issuer) & isTrusted(Issuer)`. These would be conjoined to other conditions within the 'conditions element' of data handling policies, transfer policies, etc. In particular, we may extend the access control representation given in Section 11.4.3 by defining a set of trusted-based predicates of the form `predicate_name(`*arguments*`)`. In this case, the assurance control checking is invoked as part of the Access Control Decision assessment within PRIME (see Section 11.4.3). Here, the assurance policy can be injected into the existing access control policies as predicates. For example: `IF (Access Control checks) AND IF (Assurance Control checks)`
2. An alternative approach is to use a much higher-level representation in which individual human-readable clauses within the policies are first class objects, thus defining a completely separate policy representation language from the other approaches presented in this chapter. Here, the checking is invoked as an independent assurance control function invoked by an entity to conduct tests against a compliance template: for example, a user initiating compliance checking of a website against their "e-commerce" policy.

Accordingly, there are different possible levels of representation that can be used within assurance policies, respectively:

1. At a low level e.g. $\forall x \in$ `ProcessingSystem (hasWorkingTPM(x)` $\vee$ `...) & usesAdequateEncryption(x) & isPatched(x) & hasWorkingOMS(x)`, with reference to a lower level semantics
2. At a high level e.g. `checkTrustedProcessingSystem`, with reference to a higher level semantics

The result of the assurance checking is an analogous structure to the assurance policy input (with the resulting values of the assurance clauses). The complexity can be shown to the user if desired, and the structure can simplify to a Boolean value, for instance so that the access control decision point may make a decision about what to do next.

The following sections consider these different approaches in turn.

13.2 Defining Trust Constraints: A Lower Level Representation

Our initial PRIME implementations used the notion of trust constraints [Pea06]. Trust constraints are part of a broader representation of constraints

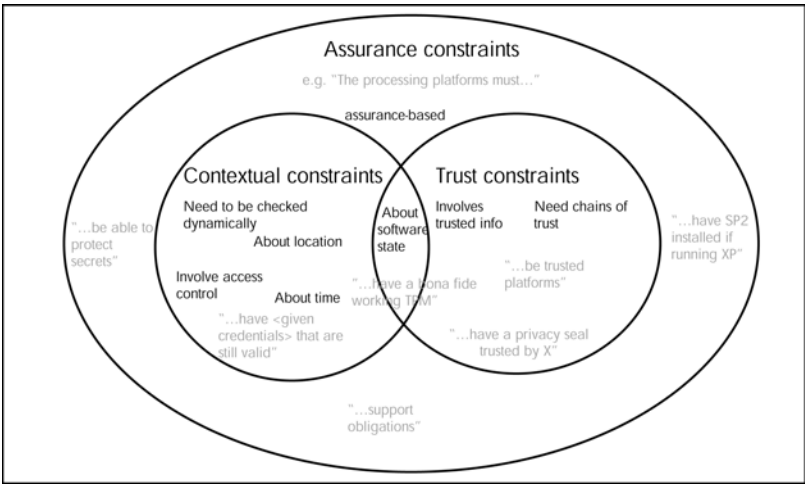

Fig. 13.1 Trust and assurance constraints

within policy languages. Figure 13.1 illustrates how, within the context of policies and preferences, they are a subset of a broader set of constraints about data processing: assurance constraints.

Assurance constraints may be contextual constraints, i.e. formulated by people to restrict the cases in which their data will be processed, according to parameters that may vary dynamically (such as time, location or platform state), or trust constraints. Figure 13.1 shows how these can be related and provides some examples. To a greater or lesser degree, all assurance constraints could be regarded as trust constraints since something must ultimately be trusted to make the assertions (and indeed the policy compliance checker must be trusted to issue compliance statements), but for convenience we may distinguish those statements that directly involve trust-related information and for whose automated evaluation a compliance checker needs to take chains of trust into account.

These constraints may be expressed within user-side preferences or policies. Such policies (*assurance policies*) would then include a set of conditions and constraints formulated to obtain degrees of assurance from enterprises that their data will be processed according to people's expectations, such as compliance to privacy, security and IT standards. On the client side the assurance control module (AC) can check their satisfaction (via information provided by the service-side AC) prior to disclosing any personal information or during the negotiation process (when AC provides input to both the local user and service-side requester). It can also be desirable to check contextual constraints on the service-side after information has been disclosed. This would be through the use of sticky policies, which are negotiated using the preferences and then associated with data as it travels around, perhaps just using a weak binding, although preferably this would use a strong binding

provided by cryptographic mechanisms: see for example [CPB03]. Furthermore, such constraints may be expressed within service-side access control policies to help enterprises comply with privacy legislation, such that service-side AC will not allow a transaction to be continued unless the constraints are fulfilled.

To clarify exactly what we mean by assurance policies (or constraints), let us consider the W3C Platform for Privacy Preferences (P3P) [Wor02] and Enterprise Privacy Authorisation Language (EPAL) [IBM04] schemas representation of privacy policy rules. These rules are formed of six elements, namely data user, data item, action, purpose, conditions and obligations. Assurance policies could be thought of as an extension to privacy policy rules in that they contain certain trust, contextual or assurance constraints which, if fulfilled, are not sufficient for the transaction to proceed: semantically, these constraints are necessary conditions. An example of an assurance policy which is an access control policy would be:

```
subject with subjexp can action on object with objexp if
condition onlyif assurance constraint.
```

(Alternatively, such an assurance policy could be represented by conjoining the assurance constraint to each subcondition.) There can be other, similar, forms of assurance policy, such as a policy that is attached to data and that contains assurance constraints that must be satisfied before certain actions may be performed on the data.

Within PRIME, we used this approach to define an assurance policy by defining trusted-based predicates (including those shown in Figure 13.1) within the access control policies (for further details see Section 11.4.3). Within the PRIME implementation, when such constraints were presented for evaluation to the access control module, the trusted-based predicate (i.e. the logical expression involving assurance constraints) would then be passed to the assurance control module for evaluation, and the result passed back to the access control module in order to calculate the overall policy satisfaction.

Assurance constraints (including trust constraints) can also be thought of in an orthogonal sense as breaking down into subconstraints, such that there can be functional decomposition of higher-level privacy and trust goals into one or more lower-level goals, and so on recursively until facts about the knowledge base (e.g. checks about the value of constraint settings, the presence of software, the availability of services for a given minimum uptime, etc.) are invoked at the lowest level. This decomposition is captured by rules within our system that hook into the PRIME ontologies used so that the meaning can be agreed across multiple parties. For example, even a fairly low-level trust constraint such as that the receiving party should use tamper-resistant hardware to store key information must be defined in such as way as to make clear the manufacturers, version numbers and other ancilliary information such as degree of tamper resistance that would or alternatively would not be acceptable. In practice, a third party would define such rules in advance and

then they would be viewable and/or customisable if desired at a later stage by users or administrators. See [Pea06] for further details about this approach.

13.3 Defining Clauses as First Class Objects: A Higher-Level Representation

The above representation was used within the initial phases of the PRIME project. However, in the final phases we refined this approach to replace it by making clauses be first class objects. We found this approach to be preferable to the previous approach, because humans need to read and interpret the assurance policies. Therefore, we wished to tilt the balance in favour of ease of understanding, with the clauses being expressed in natural language, at the expense of the expressivity and richness of the language. We wished to push the complexity of the checking to the third parties involved in the production of evidence, and make things as simple as possible for the end users.

Our model of assurance policies can be analysed by means of different (but equivalent) perspectives, namely the conceptual view, the formal view, and the operational view. We consider these views in turn, and provide examples of assurance polices and the language used.

13.3.1 Conceptual View

Both users and service providers have the freedom to create policies to suit their needs. In order to bring the two together a common vocabulary is developed. This comes in the form of privacy statements or privacy clauses which are a basic primitive of our solution. A clause is a statement concerning a particular privacy aspect of PII. It is succinct, clear, and unambiguous and clearly communicates its intended purpose at a level that does not require expert knowledge of privacy systems or their implementation. It is expressed in natural language with the aim that both clients and services will be able to understand each other more clearly. This empowers an end-user, of whom it is assumed to not have technically advanced knowledge, to communicate their privacy preferences in a language they understand. Later in this chapter we discuss how our policies relate to previous work on policy definition.

A policy is a collection of clauses, crafted for a particular purpose depending on the context of the interaction. Both the user and service provider will invoke the policy that they feel is the most appropriate depending on the context. For the user interacting with a bank they may invoke an "on-line banking" policy; for a service provider interacting with an on-line shopper they may invoke a "website customer" policy. The policies will be geared towards making sense of the context in which they are used. So an "on-line banking" policy may have stricter and more numerous clauses than a "signing up for free email account" policy. It is up to the user and service provider to maintain a pool of polices and invoke them under the proper circumstances.

This should then marry up the amount of processing and level of assurance required dependent upon the situation.

There is still an issue about where the clauses come from in the first place, and who provides guidance or establishes what is an appropriate policy for a particular purpose and what is not. In order to facilitate both problems it is important that there be some agreement about privacy in general and clauses and policies in particular. A way of doing this is through standardization. Trusted entities, such as governments or standardization bodies such as the W3C, who have experience in this field through efforts like P3P, can be called upon to provide a working pool of clauses and provide guidance on how to go about creating a privacy policy that is appropriate for a particular activity as a template (see further discussion below).

This approach has an important quality which we have dubbed *privacy positive*. A privacy positive statement is one that is privacy friendly. The clauses are created carefully and worded in such a way to be privacy positive: that is to say that a clause will never reduce the level of privacy afforded to the individual.

The predefinition of clauses is important for three reasons.

1. The clauses are not concerned with technical implementation details: only statements about privacy as required by law, good business practice, and consumer protection will be present. This abstracts away the technical details from the essence of the statements which are only concerned about what should happen with PII and not how it should happen. It prevents restrictions on the way the solutions are implemented and also prevents users from having to be technically savvy to use this scheme.
2. To protect users from having to understand technical details about privacy products and construct detailed policies which may be removed from practical reality, users use clauses that they care about to fashion their policies. In the same vein a service provider, although more technically knowledgeable, uses the same clauses and can speak the same language as its users and can communicate its responsibilities clearly.
3. Since the same pool of statements are being used by both the users and service providers it is an easy matter to match up expected policies with actual ones and negotiate the mismatches. At least in this way the glaring omissions in service providers' policies will become obvious and in the same way unrealistic expectations from users can be cleared up. Where there are deficiencies in specific clauses, the totality of the policy must be looked at. The set of clauses that form the policy is a stronger indication of the suitability of a policy than the individual clauses of which it is made up. Even if there is disagreement between a user and the service provider at least both know where the other stands on privacy.

We are aware that positive and negative clauses are subjective but it is hoped that through proactive efforts by lawyers and privacy experts in concert

with privacy groups it is possible to arrive at a standard of privacy expectations and conduct.

The idea of an *assurance policy template* is an optional extension to this approach, which can be convenient for users in order to help them build up their assurance policies with the help of entities that they trust. An assurance policy template can be thought of a set of default policies (or more specifically, clauses suggested to the user to include within their assurance policy), associated with a given context. For example, an assurance policy template might be suggested by a consumer group for a particular scenario (e.g. purchasing goods of value less than $100 online), and this template would list the checks that the consumer group recommended making in that case. There could be different templates for different contexts, and potentially more than one template for a given context; it would be up to the user to select the appropriate template which they wished to use, if any — this could be done automatically in fact, once the users' initial choices about which templates to use if different preconditions matched were selected and stored.

Templates for policies can provide a set of clauses that adhere to best practices or commonly held standards. To this a user can add or remove clauses depending on their preferences and needs. Templates are especially geared towards end users who may need help creating a privacy policy that would serve the purposes that the end user needed them for.

For example a template for on-line banking may recommend that:

1. PII remains confidential in transit
2. PII is only accessed by authorized personnel
3. A valid privacy seal is present
4. PII is not released to third parties without the consent of the user.

The template can be used in a policy editor to further ease the creation of policies. The end user could use the template as a solid starting point and then tweak it to their desires. This way they can concentrate on their privacy concerns rather than worry about technologies and get distracted from their original intentions. Similarly, a business could also use templates to the same effect although it would have to be careful to only include those clauses it had the actual capability to enforce. A business could not include a clause it could not honour into its policy since the involvement of trusted third parties (TTPs), to be discussed in Section 16.2.7, prevents this type of abuse.

13.3.2 Examples of Clauses

Examples of privacy positive clauses can be about any aspect of privacy, from:

- We will not share your data without your consent

to

- We will delete your PII after 30 days

A clause will not break common privacy expectations or allow circumvention by statements such as:

- We will share your data with third parties

or

- We reserve the right to store your data indefinitely

Such privacy negative clauses do not add to the privacy of consumers and would not be valid in privacy policies or adopted in the standard clause pool.

13.3.3 Formal View

From a formal perspective an assurance policy template can be seen as a $\langle ctid, pc, L(cc)\rangle$ tuple, where $\langle ctid, pc, cc\rangle \in \langle CTID, PC, CC\rangle$ and where $L(cc)$ defines a logical combination of cc, such that:

- $pc \in PC$: set of all preconditions
- $cc \in CC$: set of all assurance/compliance clauses
- $ctid \in CTID$: set of all unique identifiers

An assurance policy is a $\langle L(cc)\rangle$ list, where:

- $\langle cc\rangle \in \langle CC\rangle$
- $cc \in CC$ is the set of all assurance clauses

Further formalisation of this view is beyond the scope of this section.

13.3.4 Operational View

From an operational perspective, assurance policy templates (aka. compliance checking policy templates) can be seen as collections of compliance clauses. A representation would be:

```
CCPT ctid:
 IF  <pc>
Check L(cc)
```

In a similar way, assurance policies (aka. compliance checking policies) can be seen as collections of clauses as determined by pre-conditions:

```
CCP ctid:
   Check L(cc)
```

For example, *Transfer selected PII if it adheres to Government Policy Template* might correspond to:

```
CCPT ctid1:
IF  templateIS(Government)
CHECK
inEU(ReceivingLocality) AND trusted(ReceivingParty)
```

and *Transfer sensitive selected PII only if it adheres to Secure Storage Policy Template* might correspond to:

```
CCPT ctid2:
IF  templateIS(SecureStorage)
CHECK
NOT(isSensitive(t1)) OR (trustedPlatform(Device) AND
            encrypted(t1,minLevel))
```

13.3.5 Representation of Assurance Policies in XML Format

Within PRIME, we have used an XML format to represent assurance policies. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ccpolicy SYSTEM "ccp.dtd">
<ccpolicy>
<ctid>1337</ctid>
<clause gid="1">
<option>1285</option>
<option>AES</option>
</clause>
<clause gid="2">
<option>128</option>
</clause>
<clause gid="3">
</clause>
</ccpolicy>
```

where `ctid` is a unique identifier to identify each policy, `gid` is a unique well-known global identifier whose mapping to human-readable form is standardised, and `option` is a refinement to the clause if applicable.

The corresponding representation of (an example of) the results of the compliance checking process is:

```
<ccpolicy>
<ctid>1337</ctid>
<clause1 gid="1">
<constraint1>128</constraint1>
<constraint2>AES</constraint2>
<result>Y</result>
```

```
<signature>abcdef1234567</signature>
</clause1>
<clause2 gid="2">
<constraint1>128</constraint1>
<result>Y</result>
<signature>1341341234124</signature>
</clause2>
<clause3 gid="3">
<result>Y</result>
<signature>98765787646</signature>
</clause3>
</ccpolicy>
```

The DTD schema underpinning this XML format is:

```
<!ELEMENT ccpolicy (ctid, clause*) >
<!ELEMENT ctid (#PCDATA) >
<!ELEMENT clause (option*) >
<!ELEMENT option (#PCDATA) >
<!ATTLIST clause gid CDATA #REQUIRED >
```

The clauses in the standard clause pool are stored in any suitable form, e.g. a string, or a URI, and each of these is associated with a natural number that is the clause global identifier (`gid`) so that they can be referenced efficiently.

The user assurance policies specify which of these clauses within the standard clause pool the user wishes to check, and the service-side assurance policies specify which clauses within the standard clause pool the service provider is willing to testify that it can provide. These assurance policies are of the same form.

13.4 Analysis

During research it was found that assurance ontologies were not the best candidate for assessing the trustworthiness of the back end. The modelling of back end systems was difficult due to problems of classifying technologies and processes into a coherent assurance ontology that captured all types of systems and variations that are present in real world deployments. Also, this meant that there was a direct link between the technology in use by back-end systems and privacy policies, since privacy polices had to express privacy in terms that back end systems could understand. Another related problem was that even if the model were perfect and complete there existed an expectation that the end user be competent enough to gauge how these technologies benefited them. To avoid these two problems, standardized privacy clauses were introduced, the functionality of which was discussed above.

A policy in the context of this section is the formalization of another party's privacy compliance request. Here, the 'policy' parameter is taken very broadly,

and could just include references, or could be a very rich structure. There has been a great deal of work done on privacy polices [Wor02, HJW02, KSW02a, KSW03, MB03]. In these policy frameworks the focus has been on access control based on conditional logic. Our policies are a departure in that they are not processed against some rule set to produce a decision on whether data should be released. Rather, polices are just collections or groupings of clauses that serve a particular purpose under a particular context. Our solution takes into account that access control plays a big part in the control of PII and so the Assurance Control component works in concert with other components in the PRIME framework, namely Access Control Decision Function (ACDF) and Identity Control (IDCTRL), to address a variety of aspects needed within a privacy solution, from setting privacy preferences and handling PII requests, to controlling PII release.

P3P is a W3C specification that allows websites and end users to specify their privacy practices and preferences respectively in a standardized way that are easy to retrieve and interpret by end users. It allows a user to delegate the privacy policy "reading" by software agents that compare retrieved website polices against the one created by the user. Only policies that are in violation are flagged to the user who must decide what to do. There have been many critiques of P3P such as [Clab, Ack04, HJW02, Claa]. We shall ignore politico-economic arguments and focus on how our solution differs from P3P, the gaps it fills in, and how P3P could be used within the system we have implemented albeit with changes to its role.

Expressing privacy concerns in P3P is done by defining statements in a machine readable format written in XML [Wor02]. Although there are editors [P3P] that help with this process, there are two problems that are not yet addressed.

First, the P3P language and editor are tools but the end user must know what they wish to express in the first place. They must know what their privacy vulnerabilities are and how to check if a website will mitigate those risks. Most users are naïve and would not be competent enough to express privacy concerns beyond vague statements.

Second, even with the prerequisite privacy knowledge the definition of privacy polices must be in a language geared towards the facilitation of accessing PII based on conditions. Although useful, it cannot capture other aspects of privacy adequately without losing some of the essence of what the end user intended. Our solution addresses these concerns by introducing standardised privacy clauses that are written in human-readable form and are unambiguous, concise, and capture privacy concerns based on expert knowledge. To ease the creation of polices, templates are provided. The end user does not need to learn a language or an editor that requires knowledge of predicate logic.

As is the case with privacy seals, P3P cannot link the privacy practices expressed by the website with anything tangible on the back-end. This gap is where our solution introduces mechanisms to check that policies and the technical realities of the website's infrastructure are coherent. Claims made

in the privacy policies are backed up by capability checks as described in Section 16.2.5 and help to provide assurances that are missing from the P3P model.

Although P3P has its limitations, its strength as a robust policy definition language and logic model allows it to perfectly translate privacy clauses into machine-readable form. The resultant privacy policy would have to be vetted by TTPs and also certified, or an intermediate layer could be introduced that would drive the policy editor to receive clauses and output machine-readable policies. Since the clauses are defined and standardised the resultant XML would also be identical. In our model unique global identifiers are used to identify particular clauses, the drawback being that a unique identifier needs a lookup table to be maintained, whereas an XML policy would capture all the necessary information within itself. There have to be extensions to the present P3P vocabulary so that all aspects of privacy can be expressed.

13.5 Next Steps and Future R&D Work

A policy in the context of this section is the formalization of another party's privacy compliance request. Here, the 'policy' parameter is taken very broadly, and could just include references, or could be a very rich structure.

We have used a common standardized privacy clause pool to help communicate end user concerns as well as service provider promises. These clauses form high-level assurance checking policies. The benefits of this approach are in providing flexibility, and in being extensible and customisable. Chapter 16 gives details of the framework that maps these policies to back-end technology in such a way that this abstracts the complexity away for the end user and at the same time allows the service providers flexibility in how they implement and manage their infrastructure.

Since trust is not a black and white issue, we designed this approach such that the user must have overall control over how to proceed. Nevertheless, there is subjectivity and potential changeability of the decisions and representations involved, which could be an issue.

Since clauses are the central privacy vector they need to be developed further from the select set that are being implemented now. They need to be more complex and recognise complex privacy needs of sophisticated users as well as laws and regulations that businesses must adhere to. They also need to be stated in such a way that is unambiguous in any language. Only the true essence of the privacy objective of the clause must be present in its description. This will be an interesting area which will require participation from law, business, and security experts for further refining and establishing a coherent, effective, and simple language for defining privacy issues and concerns.

Further details about assurance control are given in Chapter 16.