Modeling X.509 Certificate Policies Using Description Logics

Stephan Grill
Institute for Applied Information Processing and Communications
Graz University of Technology, Austria
Email: stephan.grill@plusultra.at

Abstract

Public Key Infrastructures are gaining importance in today's IT environment for managing certificates and keys. It is recognized, that the quality and trustworthiness of certificates depend to a large extend on the practices and procedures a certification authority applies when issuing certificates. These procedures are documented in certificate policies, which are generally text-based documents and therefore cannot be processed by machines. This paper describes a framework based on description logics that addresses this situation. Subsumption will be used to compare policies. Based on a case study of modeling real policies some features of this framework will be described. Learnings and an outlook of future work conclude this paper.

1 Introduction

Public Key Infrastructures (PKI) are emerging as an important cornerstone of today's communications systems. They are envisioned to enable a wealth of services ranging from electronic id cards, digital signature, authorization schemes, etc. and are already increasingly used in web-applications, e-mail, e-payment.

Processes and protocols to manage and use private keys and certificates are well understood and corresponding standards [1] are currently in the process of being defined.

The basic concept of a PKI is that participants use key pairs consisting of a private and a public key. The private key never leaves the trusted environment of the user and might for example be used for signing a document, whereas the public key is published for others to verify the signature created with the corresponding private key. To associate a person with a public key a trusted third party issues certificates, which express this association.

A trusted third party (also called certification authority CA) can issue certificates according to different policies. Policies define practices followed by the CA in authenticating the subject, the users obligations in protecting the private key, the legal obligations of a CA, etc. Different policies represent different security levels. For example, a policy might state that certificates are issued for public keys whose corresponding private keys are generated and stored on a smart card and a certificate requestor has to be authenticated in person using an id card, whereas a different policy might state that the private key is stored on a PC and only the existence of the e-mail address of the requestor has been verified. Obviously, information contained in a certificate issued according to the first policy is much more reliable and the certificate is therefore more secure than a certificate issued according to the latter policy. Consequently, the trustworthiness of signatures depends on the security level of the associated certificate and users verifying signatures have to make sure they are aware of the associated policies.

However, these policies are currently represented in textual form and therefore hardly ever inspected! This paper describes how description logics can be applied to represent certificate policies in a structured manner and to make them comparable.

2 Requirements for the Representation of a Policy

To define requirements for a policy representation, cases for using these models will be discussed first:

- Users should be enabled to *retrieve* specific parts of a certificate policy (e.g. they want to check where the associated private key of a certificate is generated and stored)
- Users might want to *specify properties* a certificate policy must match in order to be accepted (e.g. they want to specify that only certificates are accepted, if the associated private key is generated and stored on a smart-card).
- These comparisons might be based on equality comparisons but also other *comparison operators* must be supported which might be applicable for the quality of certificates (a policy might indicate that associated certificates are more trustworthy than certificates issues according to another policy).

From these possible use cases a set of requirements can be derived:

- Current text-based policies describe complex objects and complex practices. A formal model must support this complexity. The representation has to support *<attribute, value>* pairs, which might be organized in a hierarchical structure, and where *value* by itself might be a complex object.
- The representation must allow to define a metric and/or classification scheme to support not just equality comparisons, but also additional relational operators.

– The representation must be declarative (opposed to a procedural representation) in order to support operations on this representation.

3 Proposed Solution

A two-phased approach was chosen to address this problem. In the first phase a possible semantic representation is investigated, and in the second phase the syntactical representation of the defined semantics is defined.

Description logic was chosen for the semantic model because it provides an expressive data model and the required operators [3].

Currently an investigation is ongoing to identify a suitable syntactic representation. Likely candidates are the Resource Description Framework RDF [5] of the W3C or DAML+OIL [9].

Certificate policies are modeled in process consisting of three-steps:

1. definition of the semantics of the taxonomy
2. definition of a reference ontology
3. definition of individual policies

3.1 Definition of the Semantics of the Taxonomy

The objective of using a classification based knowledge representation mechanism is to automatically induce an order relationship in the concept space. For modeling policies, the taxonomy is used to represent information related to security aspects:

> *if concept C1 subsumes concept C2 then C1 is less secure than C2 – i.e. concepts higher up in the concept taxonomy will be considered less secure than concepts further down in the hierarchy*

This decision is basically an arbitrary, but a meaningful one. Concepts higher in the taxonomy are less detailed specified as concepts further down. From a security point of view concepts with a more detailed specification are preferred to concepts with a less detailed specification. This is the case because the more information is given about a concept the less ambiguity is possible.

Other definitions are possible as well; however it is not allowed to mix taxonomies, which represent different semantics, within a specification of a single policy.

3.2 Definition of a Reference Ontology

The objective of defining a reference ontology is to

– define a *common* and re-usable *terminology*,

- create definitions that make individual policies *comparable*
- optionally define a *primitive taxonomy* explicitly

It defines concepts and individuals, which can be refined and combined by subsequent policy descriptions using specialization and the definition of new concepts respectively. These concepts and individuals define a core terminology, which ought to be accepted as a common framework. It will be possible that several of such reference frameworks will be developed and these frameworks can be combined.

3.3 Definition of an Individual Policy

Finally concepts will be defined to create certificate policies. These concepts use the reference ontology and therefore are based on the order relationship created through the definition of the taxonomy of the reference ontology.

4 Case Study

To verify the applicability of this approach a reference ontology using the framework defined in RFC 2527 [2] and two actual policies have been modeled using the DL-system NEOCLASSIC [4]. The chosen policies are

- Certificate Policies for the Government of Canada (GoC) Public Key Infrastructure [6]
- Swedish SEIS Certificate Policy [7]

The GoC PKI Policies actually define policies representing different assurance levels: rudimentary (1), basic (2), medium (3), high (4). In the following examples policies for signature certificates will be identified with the prefixes `GocSign[1234]`.

The reason for choosing these policies was, that all are based on the framework suggested in RFC 2527 - and therefore provide a possibility to make them comparable.

The following examples will show this approach more specifically.

4.1 Key Sizes

The first example is rather simple; it is using the predefined properties of the build-in concept `Integer`.

The core terminology based on RFC 2527 requires describing the minimal length of the used keys:

```
( createConcept Rfc2527AsymmetricKeySizes
    ( all keyLength Integer ) )
```

Concept `Rfc2527AsymmetricKeySizes` has one role named `keyLength`, which is of type `Integer` (a built-in NEOCLASSIC concept).

The Government of Canada PKI policy defines the following restrictions on the above concept:

```
( createConcept GocSign1AsymmetricKeySizes
    ( and Rfc2527AsymmetricKeySizes
          ( all keyLength ( minimum  512 ) ) ) )
```

Concept `GocSign1AsymmetricKeySizes` is a sub-class of `Rfc2527AsymmetricKeySizes` with the additional restrictions that all values of the attribute `keyLength` must be greater-equal 512. Similar restrictions apply for the remaining assurance levels:

```
( createConcept GocSign2AsymmetricKeySizes
    ( and Rfc2527AsymmetricKeySizes
          ( all keyLength ( minimum 1024 ) ) ) )

( createConcept GocSign3AsymmetricKeySizes
    ( and Rfc2527AsymmetricKeySizes
          ( all keyLength ( minimum 1024 ) ) ) )

( createConcept GocSign4AsymmetricKeySizes
    ( and Rfc2527AsymmetricKeySizes
          ( all keyLength ( minimum 2048 ) ) ) )
```

The SEIS policy is denoted as:

```
( createConcept SeisAsymmetricKeySizes
    ( and Rfc2527AsymmetricKeySizes
          ( all keyLength ( minimum  1024 ) ) ) )
```

Using the built-in properties of NEOCLASSIC's type `Integer` an ordering scheme is predefined: `Rfc2527AsymmetricKeySizes` defines a class of instances with the attribute `keyLength`, which may take all integers as value; `GocSign1AsymmetricKeySizes` restricts the values to greater equal 512; `GocSign[23]AsymmetricKeySizes` restricts the values to greater equal 1024; `GocSign4AsymmetricKeySizes` restricts the values to greater equal 2048. Because of the properties of `Integer` the application of the subsumption reasoning service results in the following ordering:

- `Rfc2527AsymmetricKeySizes` subsumes `GocSign1AsymmetricKeySizes`
- which in turn subsumes `GocSign[23]AsymmetricKeySizes`
- which in turn subsumes `GocSign4AsymmetricKeySizes`

Given the defined semantics of the taxonomy end users can conclude that certificates issued under security policy `GocSign4AsymmetricKeySizes` can be relied on most.

NEOCLASSIC also automatically determines that

– `GocSign[23]AsymmetricKeySizes` is equivalent to `SeisAsymmetric-KeySizes`

If the only requirement would be to compare numeric values the general subsumption mechanism would not be necessary - PICS [8] does something similar. However, not all properties described in a policy can be represented by numeric values, which can be seen in the next examples.

4.2 Key Pair Generation

In order to support comparison operations it is necessary to define an order relationship amongst newly defined concepts.

Example: a key pair generated in hardware might be more trustworthy than a key pair generated in software.

```
( createConcept Rfc2527Sw     Rfc2527ModuleTypes )
( createConcept Rfc2527HwOrSw Rfc2527Sw )
( createConcept Rfc2527Hw     Rfc2527HwOrSw )
```

Above statements define that

- `Rfc2527Hw`, `Rfc2527HwOrSw`, `Rfc2527Sw` are sub-classes of `Rfc2527ModuleTypes`
- `Rfc2527Sw` subsumes `Rfc2527HwOrSw`
- `Rfc2527HwOrSw` subsumes `Rfc2527Hw`.

This taxonomy is now associated with a security related semantic: module types whose concept descriptions subsume others are less trustworthy than the module types associated with the subsumed concepts.

Using this taxonomy `Rfc2527KeyGeneration` is defined with two attributes: `caKeyGen` and `eeKeyGen` (generation of keys for CA operations and generation of keys for end user operations respectively). Both of which require values that belong to the concept `Rfc2527ModuleTypes`.

```
( createConcept Rfc2527KeyGeneration
    ( and ( all caKeyGen Rfc2527ModuleTypes )
          ( all eeKeyGen Rfc2527ModuleTypes ) ) )
```

The Government of Canada policy can be specified as:

```
( createConcept GocSign2KeyGeneration
    ( and Rfc2527KeyGeneration
          ( all caKeyGen Rfc2527HwOrSw )
          ( all eeKeyGen Rfc2527HwOrSw ) ) )

( createConcept GocSign3KeyGeneration
    ( and Rfc2527KeyGeneration
          ( all caKeyGen Rfc2527Hw     )
          ( all eeKeyGen Rfc2527HwOrSw ) ) )
```

```
( createConcept GocSign4KeyGeneration
    ( and Rfc2527KeyGeneration
          ( all caKeyGen Rfc2527Hw     )
          ( all eeKeyGen Rfc2527Hw     ) ) )
```

NEOCLASSIC will determine that `GocSign2KeyGeneration` subsumes `GocSign3KeyGeneration`, which subsumes `GocSign4KeyGeneration`. This can then in turn be interpreted in such a way that certificates associated with policy `GocSign4` are more secure than certificates associated with policy `GocSign2`.

The SEIS policy can be described as:

```
( createConcept SeisKeyGeneration
    ( and Rfc2527KeyGeneration
          ( all caKeyGen Rfc2527HwOrSw )
          ( all eeKeyGen Rfc2527HwOrSw ) ) )
```

NEOCLASSIC will recognize `SeisKeyGeneration` as being equivalent to `GocSign2KeyGeneration`.

5 Summary

This paper shows how description logics can be used to represent certificate policy information. It has been discussed how subsumption can be used in order to compare the quality and trustworthiness of certificates.

Performing the case study of modeling different policy the following observations have been made:

- The definition of a core terminology in form of an ontology is necessary. RFC 2527 actually provides some kind of framework that can be followed to specify such a reference terminology.
- It also became apparent that policies that follow RFC 2527 are difficult to compare because this framework leaves too much room for interpretation and expressing different aspects.
- This shows that users who want to compare the quality of certificates actually do face a major problem, as existing policies are difficult to compare.

Planned future work comprises the investigation on the use of more expressive DL-systems and how a DL-based language can best be syntactically represented.

While performing this work it also became apparent that different domains are using varying semantic data models to represent authorizations, capabilities, rights, etc. These different representations in turn require domain-specific processing algorithms. It seems promising to study how a DL-based system can be used as a unifying scheme for a generic policy specification.

6 References

1. Housley, R., Ford, W., Polk, W., Solo, D.: Internet X.509 Public Key Infrastructure: Certificate and CRL Profile, IETF RFC 2459, 1999.
2. Chokhani, S., Ford, W.: Internet X.509 Public Key Infrastructure: Certificate Policy and Certification Practices Framework, IETF RFC 2527 (1999)
3. Donini, F. M., Lenzerini, M., Nardi, D., Schaerf, A.: Reasoning in Description Logics, CLSI Publications, Principles of Knowledge Representation and Reasoning, (1994) 193-238
4. Borgida, A., Patel-Schneider, P. F.: A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic, Journal of Artificial Intelligence Research 1, (1994) 277-308
5. Lassila, O., Swick, R.: Resource Description Framework (RDF): Model and Syntax Specification, W3C Recommendation, (1999)
6. Treasury Board of Canada, Secretariat: Digital Signature and Confidentiality Certificate Policies for the Government of Canada Public Key Infrastructure“, Version 3.02 (1999)
7. Secured Electronic Information in Society: SEIS Certificate Policy SeisS10-1: 1.0, High assurance general ID-certificate with private key protected in an electronic ID-card, Version 1.0 (1998)
8. Krauskopf, T., Miller, J., Resnick, P., Treese, W.: PICS Label Distribution Label Syntax and Communication Protocols, W3C Recommendation (1996)
9. Harmelen, F. van, Patel-Schneider, P., Horrocks, I., “Reference description of the DAML+OIL (March 2001) ontology markup language”, http://www.daml.org/2001/03/reference.html, March 2001