RZ 3674 (# 99684) 10/30/06
Computer Science 30 pages

Research Report

A Meta-Data and Reasoning Framework for Open Assertion and Evidence Exchange and Query

Giles Hogben

Joint Research Centre, Ispra site
Via E. Fermi 1
21020 Ispra (VA)
Italy

Dieter Sommer

IBM Research GmbH
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

Email: dso@zurich.ibm.com

Research
Almaden · Austin · Beijing · Delhi · Haifa · T.J. Watson · Tokyo · Zurich

A Meta-Data and Reasoning Framework for Open Assertion and Evidence Exchange and Query*

October 2006

Giles Hogben
Joint Research Centre, Ispra site, Via E. Fermi 1, I-21020 Ispra (VA), Italy
Dieter Sommer
IBM Research, Zurich Research Lab, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland

Abstract

We provide a framework and architecture for enhancing the privacy and openness of identity management systems. We propose that anonymous credential systems may be used as cryptographic mechanism to satisfy legal and social requirements for both privacy and security within identity management systems. We outline the use cases and requirements for integrating anonymous credential systems into our framework and architecture. We propose an ontology-based meta-data model for satisfying these requirements and show how this can be applied to the use cases. We also discuss a security model for the use of ontologies within this model.

*Part of the work reported in this paper is supported by the European Commission through the IST Project PRIME. The PRIME project receives research funding from the European Community's Sixth Framework Programme and the Swiss Federal Office for Education and Science.

1 Introduction

Efficient, legal and user-friendly management of identities has emerged as one of the key problems to be solved in current IT infrastructure. Industry and ecommerce require high volume, well-authenticated and accurate personal data, with efficient protocols and architectures for collecting, updating and querying such data.

End users struggling with password-fatigue require solutions which automate the provision of authorization credentials. Surveys also show that, given the choice, significant numbers of end-users prefer to minimize the amount of identifiable information disclosed in electronic transactions [4, 18, 14]. It has been shown, however that users are willing to invest very little time and effort in understanding identity management systems and in dealing with mechanisms for increasing their privacy [28, 24].

Government legislation places stringent requirements for privacy of personal information, but under certain conditions, requires strong identification and revocability of this privacy. A key requirement coming from legislation is created by the principle of minimization of data collection, whereby data collected by services should be minimal for the purpose required.[17] This is an important requirement for the design of identity management solutions. Existing solutions almost always request more data than is required. This is in part because additional data often cannot be captured later and it is difficult to determine the minimum information required for a given purpose. Furthermore, even when the data corresponding to minimum disclosure can be clearly determined, traditional data structures and even certificates often do not provide the appropriate semantics for selecting and transmitting such data. Traditional certificates for example either show all the attributes contained in the certificate or none at all. Federated identity solutions using SAML, for example, allow for selection of single attributes, but as we will show later, the flexibility available within the flat attribute/value data structure provided does not allow for true minimization.

For example, strictly speaking, for the purpose of renting out a car, a rental agency needs to know that a person is the holder of a valid driver's licence and that they have paid for the service. Any further information is required only under certain conditions, such as the occurence of an accident. However, existing data structures are set up in such a way that the only way to prove that one is the holder of a valid driver's licence is to certify all the information held in a driver's licence. They do not provide for proving for example, ONLY that you hold a credential of a certain type (e.g. a drivers' licence), without providing the credential.

Apart from such problems for servers in expressing assertions required for access control, another important problem is the expression and evaluation of evidence offered in support of assertions. The most common manifestation of this problem is the difficulty of users in evaluating the trustworthiness of security certificates. Users often do not know whether to trust individually named organizations which are described by cryptographic certificates. Such decisions are also very time-consuming and users often make poor decisions as a result [30]. Furthermore, there is no semantics available for automatic evaluation of such certificates based on delegation to preferences expressed in machine-readable rules.

This paper proposes a semantics for describing assertion request and exchange (including anonymous credentials) capable of describing minimalized personal data for authentication and authorization. It also describes a framework for machine-readable evidence offered in support of those assertions, which allows for user-friendly descriptions of certificates, abstracted from the key and organization name. We also propose an architecture and semantics for abstract classification of certificates which allows users and administrators to create high-level rules applied to assertion evidence. It then describes how this semantics can be used to integrate anonymous credential systems within architectures for large-scale, secure, user-focussed identity management. This report presents the preliminary results of our ongoing research work in this area of digital identity management.

2 Terminology

Principal: The primary object of an identity management service. The subject about which assertions are requested or made

Identity Provider: A service offered for making and authenticating claims about identity principals. Identity provisioning is often realized by the issuance of a security token to the principal. State-of-the-art federated identity management protocols require that such a token be issued for each identity provisioning transaction the identity principal executes. Anonymous credential systems

allow that a token (anonymous credential) issued once can be reused many times without re-involving the identity provider.

Relying Party: A service which receives and makes use of claims about a principal and evidence presented to support those claims

Certificate: Evidence or testimonials concerning rights to actions or reputation made by one entity (issuer) about a principal.

Certification: The provision of a certificate.

Assertion: A statement, claimed to be true by the party making it.

Private Certificate (anonymous credential): A special kind of certificate that can be used by its holder in a privacy-friendly way to assert attribute information to relying parties: The holder of a private certificate can compute evidence for assertions that are consistent with the attributes of the private certificate without revealing the private certificate. This particularly allows for revealing partial information on the attributes of the private certificate. Multiple usages of a private certificate are unlinkable to each other and to the issuance transaction of the certificate unless the released attribute information allows for establishing linkability. Private certificates are a generalization of anonymous credentials. See, for example, [] for details on the protocols for handling private certificates.

Evidence: Tokens, testimonials or procedures offered with the purpose of increasing the belief that an assertion is true. Note that while this is currently usually PKI-based tokens, we propose a general architecture which also includes evidence such as reputation and distributed key trust.

Access control policy: A machine readable policy governing access rights to electronic resources.

Object: The object of an access control policy is the specific item of data for which it specifies and controls access conditions.

Data Minimization: Minimization of data used to satisfy access control requirements.

Vouching party: A party vouching for the truth of an assertion (usually using a cryptographically signed assertion).

3 Identity Management Requirements

Based on the constraints outlined in the introductory discussion, we derive the following requirements for a general IDM system. These also apply to both traditional IDM systems and systems based on anonymous credentials.

1. A mechanism for requesting assertions about identity principals (either from the owner or an identity provider). This mechanism should be able to describe a set of assertions which a desired response must be a member of. In other words, the relying party must be able to communicate to the principal which assertions it needs to be assured of before allowing access to a service. This request can also be passed on to the identity provider as a request for security tokens.

2. A mechanism for communicating assertions about identity principals (either by the owner or from an identity provider). For example the system must be able to communicate an assertion that 'Bob's email address is 'bob@foo.com'. Such assertions may also be communicated by the identity provider.

3. The assertion language should be able to express assertions which minimize the increase in the relying party's knowledge about the principal for a given interaction. For example instead of providing an exact birthdate in order to buy a 12 rated movie ($Equals(valueOf(\text{User}, \text{Age}), 12)$, it should be possible to prove the exact minimal assertion required – i.e. $Greaterthan(\text{User}, \text{age}, 12)$.

   When interpreted strictly, this requirement has the important implication that the assertion request and return language must allow for arbitrary arity predicates rather than restricting assertions to atomic name-property-value (binary) predicates as in current identity management systems.

4. Assertion *requests* should, according to the knowledge available to the relying party, include in the set of assertions requested, the assertion which minimizes the knowledge transferred about the principal, while still satisfying the access control request. Henceforth, we shall call this the *minimal-disclosure assertion*. Consider, for example, a cinema ticket webstore which needs to know that the person being issued the ticket is over the age of 12. If the server requests $Greaterthan(\text{User}, \text{age}, 12)$ – evidence that the user's age is greater than 12, then all assertions stating an exact birth date before 1994 would be a member of the requested set, AND the minimal disclosure assertion – that the user's age is greater than 12 is also a member of the requested set. If, instead, the server requests $Equals(valueOf(\text{User}, \text{Age}), ?)$ – i.e. only the exact birth date of the user is satisfactory, then the minimal disclosure assertion, $Greaterthan(\text{User}, \text{age}, 12)$ is not included in the requested set, therefore this requirement is not met.

   Note that conversely to the minimization requirement, the relying party may deliberately choose not to specify the full set of assertions which would satisfy the request. From the point of view of the relying party's policy secrecy, the assertion set should be as big as possible (to release as little information on the policy as possible). The larger the set of possible assertions in the relying party's request, the less information is revealed about the relying party's policy. This maximization of the assertion request set is known as policy sanitization. For example a recruitment company might ask for assertions about the user's height instead of the minimal set of assertions it actually requires which is that he/she must be over 1m60 tall. Bonatti and Samarati [3] introduced the idea of sanitizing a request before sending it to the user. Bonatti and Samarati refer to this as *policy filtering*.

5. A mechanism for describing, requesting and evaluating evidence for assertions (in order to make trust/risk evaluations). For example, at a minimum, the assertion $Greaterthan(\text{User}, \text{age}, 12)$ can be simply declared to be true. In this case, no evidence is offered other than the fact that it is being asserted by a party whose identity is known with a certain degree of trust. Or at the other extreme, it can be signed by a vouching party whose identity is assured by cryptographic means and who is trusted by the relying party. In between these 2 extremes, evidence may also be presented which is not cryptographic, but still increases the trust of the relying party to a certain degree. For example the URL of a web page which appears to have been created by the principal more than 12 years ago might be presented as evidence that he/she is more than 12 years old. Obviously not all evidence carries equal weight when evaluating the trustworthiness of an assertion but meta-data structures must be provided which allow different grades of evidence to be expressed unambiguously.

6. Evidence meta-data should allow for the efficient evaluation of the trustworthiness of assertions. Generally, decision making processes are based on access control policies which express conditions over assertions and the evidence offered to support them (currently usually none, or cryptographic certificates). Such policies are usually evaluated automatically by the relying party's system. So for example if a set of url's is offered as evidence, then the characteristics of these urls which support the assertions should be described by the metadata in such a way as to allow an automated trust decision on their basis.

7. If evidence is provided in the form of a testimonial from a vouching party (usually a public key certificate), then the meta-data must provide a) the means to verify the identity of the vouching party (in most cases this means that the name, verification method and means to access the public key of the verifier should be accessible); b) all properties of the vouching party relevant to a trust decision on this evidence. In other words, the meta-data should allow a means to verify the identity of vouching parties and, given succesful authentication, meta-data to decide the weight to be given to the vouching party's testimonial.

8. A mechanism for deciding whether, if an assertion about an identity principal is correct, it satisfies the access control requirements for the service. This means that the syntax and semantics provided should allow the creation of efficiently decideable access control rules. An example of a semantics which would violate this requirement is one with a very large space of data types - for example with a discrete model of numeric values, which would make the evaluation of access control rules very heavy computationally.

4 Prior Art

A number of mechanisms exist in the identity management space which (partially) address these requirements. A large body of work exists in the field of identity management, in particular in the areas of (secure) attribute provisioning and access control. These two areas are orthogonal building blocks for an identity management architecture. Even a brief summary of existing work in this area would be an undertaking far beyond the scope of this paper. Thus we will give an overview of basic classes of mechanisms and their properties.

4.1 Attribute Provisioning

Currently, the most common way of establishing a digital identity on the internet is to use a *form filler* (that is, *declarations* of attributes) and provide identifying data. Typically, the data provided are sanity-checked by the recipient in order to ensure a minimum degree of data quality. This way of identity management has multiple drawbacks:

- Web forms are insecure in terms of input not being endorsed by an identity provider, that is, the recipient cannot be sure whether the data are correct and sanity checking only solves the issue partly, e.g., for credit card number correctness.
- The approach is tedious for users as they have to type their attributes in each new interaction, perhaps with some support from form fillers.
- In terms of privacy, the approach is not appealing if the user's civil identity has to be revealed in an interaction, which is the case in almost all e-business interactions.
- When using the identity, all transactions using this identity become linkable.

We refer to Pfitzmann and Hansen [25] for terminology and concepts related to privacy-enhancing identity management.

Using *traditional user-side certificates*, e.g., ones based on the RSA [27] or DSA [23] signature schemes, can add security to the asserted attributes by having them third-party endorsed (certified) by the certificate issuer (identity provider). This gives the attribute recipient confidence in the attributes. But each transaction using the same certificate becomes linkable and in each transaction all attributes of the certificate have to be released. Thus, the method does not account for sufficient protection of the user's privacy as unnecessary information is leaked in each transaction with another party.

Currently established *federated identity management* (FIM) approaches improve on privacy compared to the user-side certificate approach and still maintain the security of the traditional certificate-based approach. In these solutions an identity provider issues a signed token to the principal for each transaction that he/she wants to execute. The token contains exactly the attributes that the principal wants to release in this particular interaction. The principal then provides this token to the relying party. The token can be considered a "one-time certificate," thus security is equivalent to the user-side certificate approach, but privacy is improved in the way that only the required attributes are contained in the certificate. As a drawback, the identity provider must be involved in each transaction, thus increasing the cost of the approach. The involvement of the identity provider in each transaction requires a security model with a fully trusted identity provider. If the identity provider were to share their transaction transcripts with different recipients, they together could establish comprehensive profiles of the principal. This is—in our opinion—a severe privacy problem as the required security model requires a high degree of trust in the identity provider.

Anonymous credential systems improve on this issue by breaking this linkability of transactions without having to trust the identity provider not to share their transaction transcripts [5]. A generalization of an anonyomous credential system in terms of high expressivity of assertions that can be proven with them is referred to as a *private certificate system*. We use both terms interchangeably in this work. Private certificate systems possess all the privacy and security features of anonymous credential systems. In a private certificate system, a user obtains private certificates from identity providers and keeps them in a local certificate store. Private certificates are intended to be used many times, like traditional user-side certificates, but unlike the tokens in many FIM systems. Each use of a private certificate may release a subset of the attribute information of the certificate. Each new transaction with a private certificate is unlinkable to other transactions involving the same certificate, unless the attribute information released makes the transactions linkable. This is a major improvement in terms of privacy. The identity provider

need not be trusted any more by the user not to collaborate with data recipients to infringe the user's privacy by building extensive user profiles.

Both current FIM systems and private certificate systems can be built in order to fulfill the requirements presented in Section 3. However, current FIM solutions seem to lack key aspects such as the request of the minimal disclosure assertion or the description of evidence metadata.

4.2 Access Control

A wide range of commercial products in the access control domain are available, each one using its own policy language. Within the last few years the XACML standard for access control has emerged, providing both an architecture and a policy language for access control. XACML is missing support for using ontologies in the decision process and it is designed to express binary assertion requests as described in Section 3. It assumes that the requester's attribute information is available at the time of access, thus separating the assertion request from the access control mechanism. XACML is however extensible in many respects, thus still being one viable access control language for our framework, provided that it is properly extended. Current research has made available advanced approaches to tackle the problem of trust negotiation, that is, establishing mutual trust between two parties. See for example Bonatti and Samarati [3] for a method for trust negotiation, going for beyond what a standard access control solution can offer.

5 Architecture

We put forth an architecture for identity management that provides for open assertion exchange. We present the overall architecture below, details of the reasoning architecture can be found in Section 8.4.

Two types of players are involved in the main interaction we describe: The *principal* wants to gain access to a resource of the *relying party*, the relying party has an access control policy in place that governs access to their resources. A party controls their own *access control system* which mediates access requests to resources and computes access decisions for those access requests.

The architecture is symmetric in that every party can act as principal or a relying party at different points in an interaction. This allows for symmetric multi-step trust establishment between the parties analogously to the system of Bonatti and Samarati [3].

The following describes the steps executed for a principal accessing a resource of a relying party:

1. The principal sends a request for a resource of the relying party, to the relying party.

2. The request is evaluated by the relying party's access control system. The access control policy evaluation yields an assertion request targeted at the principal or an access decision of "yes" or "no". We assume that the result is an assertion request, otherwise the user would be granted or denied access and could/could not access the requested resource. The assertion request is expressed over a data type ontology and optionally, the certification ontology introduced in this paper. That is, the assertion request contains all the information, including trust information for assertions, needed for the principal to make their data release decision, based on concepts defined by these ontologies.

3. The relying party sends the assertion request to the principal. The principal decides how to satisfy it with the identity data she has and (optionally) returns an assertion and evidence. These steps are described in detail in Section 8.4.

4. The relying party obtains the assertion and evidence from the principal. Both are stored temporarily for the duration of the transaction. These items are available as input to its policy evaluation engine in further steps.

5. The access control policy of the relying party is re-evaluated against the original request of the principal with the assertions and evidence that have been provided by the principal as input. In case the access control decision is "grant," the principal is given access to the resource. The decision can be another assertion request in case the principal's assertion and evidence don't yet fulfill the policy. This would trigger another round of the protocol. In case the access decision is "no," the principal is denied access.

We note that in the general case, the principal involves their access control system in any assertion request. These access control decisions can then yield counter-requests to the relying party for assertions they have to provide before any assertion is released by the principal. At this point, the roles of principal and relying party are swapped. Typically, however, a two-step negotiation process is sufficient for a large set of practical scenarios.

6 Detail of Abstract Semantics of Assertion-Evidence Tuples

We propose a formal abstract semantics to describe the properties set out in our requirements. In Section 8 we present a concrete semantics and syntax which covers the handling of both traditional certificates and anonymous credentials as evidence. Furthermore, the abstract semantics in this section can cover current federated identity management protocols and the use of reputation based evidence.

6.1 Assertion Semantics

State of the art identity management frameworks allow only the assertion of binary predicates with the user's pseudonym as the first argument (in other words pseudonym property object). As we have seen in the requirements, this does not satisfy the minimizeability requirement of privacy protection legislation (and best practice). Given that the minimal-disclosure assertion for a given access control decision may be expressable only by an n-ary predicate, where $n > 3$ and only one argument is the principal's pseudonym, we need to extend the semantics to include such n-ary predicates. Minimized assertions may also be boolean formulae. For example, it is often the case that proving that you have one of 2 credentials such as a passport OR a drivers' licence (but not proving which one you have), is sufficient to gain access to a system. This has significant implications for the anonymity of end-users. For example if a user can prove that they have a passport from any EU country (but not which one), then they do not need to reveal their nationality by proving such an assertion.[1] It is also worth noting that restriction to simple binary predicates is a severe limitation on the power of such systems to handle arbitrary personal information, especially within enterprise processes.

We therefore define the abstract syntax of an assertion and evidence to support n-ary predicates, using tuples of the form

$$\langle A(P_i(N_j)), E(A)\rangle$$

where A is a boolean propositional formula composed of n-ary predicates P_i with arguments N_j, $(P_1(N_0, N_1) \wedge P_2(N_2) \vee P_3(N_3) \wedge \ldots P_i(N_j, N_{j+1}))$ etc. $\ldots E$ is evidence offered in support of A.

6.1.1 Request Semantics

The request specifies a set of assertions and accompanying evidence which can be used as access credentials to the service offered by the relying party. In general, such a request may be described in terms of a set of conditions on assertions and evidence which must satisfy the access control request.

$$\forall(A_i, E_i(A_i)) \in \mathbf{R}, C_j(\langle A_i, E_i(A_i)\rangle)$$

$\mathbf{R}$ is a satisfactory response, A_i are n-ary predicates and E_i are evidence for A_i and C_j is a condition predicate on $\langle A_i, E_i(A_i)\rangle$. Note that C_j is essentially a query, specified by a query language – that is, a language for specifying acceptable sets of results from within a larger set. The abstract syntax does not stipulate that this language should be expressable in terms of templates for the final assertions set. So for example, C could state that for $\langle A_i, E_i(A_i)\rangle$ to be a member of the response set, it must have been stated between 8am and 6pm.

The request must be capable of addressing the space covered by assertions in the response. Therefore, as the response is defined in terms of boolean formulae of n-ary predicates, such conditions should be defined in terms of a boolean formula of n-ary predicates, in combination with a query semantics for defining conditions on the assertion space. An example of such a condition is $C(\mathit{Equals}(\text{User}, \text{Age}, ?x), \mathit{greaterThan}(?x, 21))$. We will see later how this can be mapped to a standard query semantics.

Note that A_i may be a propositional formula, so it is possible to make requests for disjunctions of atomic assertions. For example,

$$C((\mathit{Equals}([\text{User}], \text{Age}, ?x) \mathit{lor holds}(\text{User}, \text{driversLicence})), \mathit{greaterThan}(?x, 17))$$

[1] This would assume that EU passports be issued by one EU governmental body instead of by the individual member states.

[User] is a placeholder, which is interpreted by the query engine to mean an identifier of the particular user of which the request is being made.

The request must also make clear which predicates are required in the response set and which are part of conditions (in the abstract syntax, this is denoted by the position of the response assertions in the argument set of C_j).

Similarly, the request can contain conditions which must be satisfied by evidence offered in support of assertions. For example, the condition that the public key of the certificate presented as evidence of a principal's email address must be a member of a trusted set, would be expressed formally in abstract syntax as e.g. $A_1, \mathit{Cert}(\text{key}, A_1), \mathit{memberOf}(\text{key}, \text{trustSet})$. We assume that the syntax contains the implied or explicit semantic that the second term in the tuple (the evidence term) automatically applies to the entire set of assertions in the first term.

6.2 Evidence Mechanisms

Multiple mechanisms for creating evidence are available as support for an assertion. Below we provide an overview of the most prominent mechanisms for creating evidence and discuss their security and privacy properties.

Declaration A declaration is a statement made by a party without any endorsement by another party. Thus, it has no security guarantees. In terms of privacy, declarations allow for providing incorrect information in case this is acceptable by the relying party. In many cases of services, the relying party will want to sanity check the information the principal provides, thus leading to excessive data release.

Traditional signatures A traditional signature by a party, that is a signature with a scheme like RSA or DSA, on an assertion endorses this assertion. The signed token can be verified using the public signature verification key of the signer. The method guarantees the integrity of the assertion. Traditional signatures provide security, but have the privacy problems of linkability as outlined above.

Traditional certificates A traditional certificate is an assertion that is signed with a traditional signature scheme such as RSA [27]. The certificate (and an associated key) can be used by its holder to prove the assertion to a relying party.

Private certificates A private certificate is obtained once by a requester and can be used many times for creating proofs. A proof over private certificates can establish identity attributes securely while still maintaining privacy by not establishing linkability between transactions and by allowing for the release of a subset of the attribute information of a certificate in a particular proof. Thus, such a proof provides for the security the relying party requires and the privacy the user requires. Proofs over private certificates are based on the cryptographic technique of zero-knowledge proofs, that is, proofs that do not reveal any further information than the validity of the assertion. Zero-knowledge proofs can be compressed into one message or be interactive, each having different advantages. Of course, with a set of private certificates proofs for assertions can be only created if the assertion is consistent with the attribute information of the private certificates.

The currently most powerful private certificate systems in use are based on the signature schemes and protocols of Camenisch and Lysyanskaya [6, 7]. Private certificate systems based on one of these signature schemes are known as *idemix* systems. An idemix system based on [6] has first been introduced in [5], the generalization to the signature protocols has been done in the later publication. Camenisch et al. [8] present the abstract syntax and semantics for a general private certificate system based on the cryptographic mechanisms of Camenisch and Lysyanskaya [6, 7, 5].

The idemix systems supports multi-show unlinkability, that is, multiple transactions with the same private certificate are unlinkable unless the attribute information allows for linkability between transactions.

A particularly interesting feature of private certificate systems is that attribute information from private certificates can be provided in encrypted from together with a proof that the ciphertexts will decrypt to the attribute information as stated. This allows for realizing anonymous yet accountable transactions when a third party is involved for decrypting in case a well-defined decryption condition is fulfilled [1]. We think this is a key feature as unconditional anonymity is not acceptable for many classes of applications.

Web of trust The web of trust is the model underlying Pretty Good Privacy (PGP); the model provides a certification infrastructure for assertions (in PGP the assertions make statements on bindings between public keys and identities) without a centralized or hierarchic structure. The approach allows that each participant may endorse an assertion when they trust the assertion. Typically trust in the assertion arises from the party themselves making the statement, or the statement being endorsed by parties they trust. This way, trust can propagate in a web-like graph, hence the name of the scheme, without involving a centralized trust infrastructure. The model is typically less appropriate if an assertion can lead to liability of a party as it is easier to spoof than a PKI-based approach. This makes the tradeoff between the less open, centralized PKI-based approach and the very open web-of-trust approach clear.

Probabilistic approach The probabilistic approach proposed by Golbeck, Parsial, Hendler and others [21, 20] works without requiring cryptographic mechanisms. The fundamental idea of the approach is that if the same assertion is found multiple times on the Web, published by sites with appropriate reputation and trust properties (established either directly or transitively through other resource), then trust in the assertion increases. The trust rating is done by considering the trustworthiness of the sources. Reputation services are an aggregation of this kind of evidence.

7 Evidence Metadata

7.1 Certification Ontology

Certification is the most important semantics to be implemented for evidence meta-data. Cryptography is currently the most important means of assuring the authenticity of assertions. However, the use of cryptographic certificates in the context of automated access control policy evaluation has so far been extremely limited. Policies for certificate evaluation are restricted to lists of trusted public keys against which certificates are matched. This tends to lead to closed federations of certification providers where often only one issuer is accepted. There has not been an attempt instead to model abstract properties of certificates. We suggest that an abstract model of certification properties brings several important advantages:

(a) It allows users to describe rules over properties which they can easily understand, rather than having to understand technical aspects of certification or be familiar with individual certification authorities. For example, certificates could be modelled in relation to properties of non-electronic certification (government-issue, falsifiability, etc...)

(b) It facilitates the distribution of default rule sets over certificates, which can be directly written using concepts from (for example) legislation which are contained in the abstracted certification properties. For example, a default ruleset distributed might specify that the Identity Provider Trust Level should be OECD government for accessing criminal record data in a database.

We now propose a basic model for such a certicification semantics which can be used to instantiate the evidence term. We describe the ontology depicted by the following diagram, proceeding in numerical order through the labels. We note that the creation of an abstraction model for certificate properties requires the model itself to be cryptographically certified otherwise the model breaks the trust chain. This is discussed in detail in Section 7.4.

7.2 Certificate Properties and Classes

We refer to the figures in Appendix 1 for illustration of the relation between concepts.

7.2.1 Algorithm

Algorithms are also denoted by abstract properties which can allow users or legislators to describe them without having to understand technical details.

Multiple algorithms are available for endorsing data and verifying its correctness. Prominent traditional signature algorithms are RSA and DSA. Both of them require a cryptographic hash function to be used in addition. Appropriate hash functions are for example SHA-256 and SHA-512. Note that SHA-1 has been successfully attacked by Wang Yin and Yu [29], thus we recommend to use one of the other algorithms.

In the domain of anonymous credential systems the most useful algorithms are the SRSA-CL and BL-CL algorithms of Camenisch and Lysyanskaya [6, 7, 2]. Both these algorithms are advanced signature algorithms that allow for proving holdership of private certificates issued using these algorithms rather than sending the certificates.

We recommend that for high security, moduli of at least 2048 bits be used for the signature schemes. See [19] for recommendations on minimal key lengths for certain security requirements.

7.2.2 Identity Verification Method

This is a classification of how the identity of the principal of the assertions in the certificate is verified according to the procedures known for the certification provider. We have provided a number of classes of verification methods: face to face, biometric, secret, and token. For example, non-electronic credentials used in face to face may lead to different values of this property.

7.2.3 Identity Provider Trust

This is a set of categories analogous to those found for physical certificates, which can be used to categorize identity providers issuing certificates. For example certificates issued by government authorities (e.g. Drivers' Licences) have different trust properties from those which are self attested. We also suggest to provide a set of quantitative trust levels which can be attributed to various types of certificates. These may use the strong, medium, weak values provided for general use in the ontology.

7.2.4 Security Method

This is a classification of the physical security methods applied within the authentication scheme of the identity provider. For example if processes are verified by an audit certificate or protection profile [13] of a certain level.

7.2.5 Privacy and Security Levels

We have assigned abstract categories to describe privacy and security levels of algorithms and security methods. The privacy level property describes the linkability features of the algorithm or security method. For example if the algorithm provides unlinkable transaction pseudonyms at the identity provider level by default, then it is assigned a high privacy level. If, as in the case of RSA, all transactions with a certificate are linkable, the method is assigned a low privacy level. Security is high in both cases.

7.3 Provider Ontology

In order to be useful in the evaluation of actual certificates, the reasoning engine must have access to a mapping between public keys and such trust categories.[2] This attribution of properties to individual certificates is an ontology in itself, which we shall call the *provider ontology*. This ontology (a list of individual providers) is a key point of vulnerability in the trust model of the system because if bogus authorities can insert themselves into the list, then they can gain access by attributing themselves trusted categories. Furthermore such an ontology would be particularly sensitive because attribution to a more or less trusted category would have important commercial, legal and security consequences. It is also likely to be subject to change. It would therefore only make sense if a highly trusted party such as a government authority were to compile and distribute such a list. The list would clearly not be included within the certificate ontology, but rather imported using trusted means into the reasoner's data model at evaluation time. We therefore propose that this part of the ontology should be imported dynamically into the main ontology using trusted means. We discuss this in more detail in Section 7.4.

7.4 Security Model

A key requirement of our approach is to allow a party to choose the ontologies they trust and to allow for specifying the ontologies that may be used for reasoning on each data type appearing in their access

[2] W.l.o.g. we assume that the provider ontology defines mappings between public keys and trust categories. Note that in practice, multiple different types of certificates could be issued with one key where the certificate type is encoded in the certificate. In this case, the mapping would map pairs of public key and certificate type to trust categories.

control policy. It is of high importance that only ontologies that a party trusts for reasoning on a particular data type are used when reasoning on this data type for making an access decision.

As a motivation for the importance of this issue, consider an attacker providing an ontology that defines that certificates issued by him are equivalent to OECD-country-issued passports. If this ontology would be accepted by a service provider for reasoning over the age data type, the attribute security would be non-existent, as the attacker could issue certificates asserting any value for the age attribute and still be trusted. This shows the need for a secure way of specifying the trust assessment of ontologies and assigning ontologies to data types in the access control policy of a party.

As shown, restricting the use of ontologies to trusted ones is one of the key requirements for obtaining a secure identity federation system. Once a party has assessed the trustworthiness of ontologies they use for reasoning and trust decisions, the party can rely on these ontologies not to jeopardize their security or privacy. The approach applies equally to relying parties (service providers) and principals (i.e. end users). A relying party defines ontologies to be used for reasoning on data types in the access control policy protecting their services, while a principal defines ontologies to be used for reasoning on data types in the access control policy protecting their attributes. For a particular data type requested by a relying party's policy, the principal considers both her own and the service provider's ontologies for each data type as this is required to fulfill both parties trust requirements. The service provider only considers their ontology for deciding on whether to grant a principal access to the service.

An observation is that parties need different degrees of security for different data types they request/provide to/from others. For example, identity information in a context with legal implications needs to be highly secure (for example, a real-world passport), whereas attributes like a person's reputation in a weblog will typically require much less security. This yields the requirement for a party to specify the ontologies that may be used for reasoning for specific data types. This choice is always a tradeoff as high security comes along with a more restricted set of ontologies and thus identity providers, whereas less security allows for a wider choice between ontology providers and thus also attributes.

The remainder of this section discusses our model and mechanisms for achieving the abovementioned requirements.

Our security model is based on ontology providers issuing ontologies to the parties (relying parties and principals) in the system and vouching for these ontologies. A party assesses these third-party-issued ontologies and their ontology providers and formalizes the assessments and a specification of what ontology may be used for what reasoning purposes, that is, data types to reason on. These formalized assessments and assignments are captured in the party's trust ontology and used whenever access control decisions to the party's resources (services or attributes) are required to be computed. The trust ontology represents the complete set of the party's trust relationships that the party uses in the assertion and evidence exchange process. Thus, the trust ontology formalizes an assessment of ontology providers and ontologies where the latter define trust and security properties of identity providers as assessed by the ontology provider. Overall, the model allows a party to leverage the assessments of identity providers by third parties sufficiently trusted for this purpose instead of requiring assessment of identity providers and also define all their trust relationships on their own.

Ontology Providers An *ontology provider* is a party which issues ontologies, in particular provider ontologies and certification ontologies, and which vouches for these ontologies. An ontology provider is a specialized party which assesses identity providers regarding their trust and security properties, formalizes these assessments in an ontology and vouches for this ontology. Such an ontology is then trusted by a principal or relying party—depending on their assessment of the ontology provider—to make derivations for access control decisions for particular data types.

An ontology provider can be any party, such as an OECD government, a bank consortium, a telecommunications operator, a large public corporation, or a well-known technology weblog operator. Typically, one would trust an OECD government in providing ontologies that are used for reasoning on assertions where a high level of security is required, but would not trust a technology weblog operator for this purpose. On the other hand, an OECD government would probably not specify and issue ontologies that talk about the reputation of technology weblogs as this is too specialized a matter and not their responsibility. Particularly, liability plays an important role in this arena and motivates trust decisions regarding ontology providers.

Trust Ontology A trust ontology O is defined by each party and expresses the party's security and trust assessment of ontology providers, ontologies, and so called ontology compartments, as well as

specifying the assignment of ontologies to compartments. A compartment is a class of ontologies defined by the trust ontology which serves to separate sets of ontologies which may have unforeseen harmful dependencies. It serves to guarantee that ontologies chosen from a particular compartment can safely be used together. Without this guarantee, combining knowledge from different sources may have unexpected consequences for security.

For example, the trust ontology of a party can define a compartment C_1 containing one provider ontology issued by the European Union. Another compartment C_2 is specified to contain a provider ontology issued by the Canadian Government. Both compartments are associated with the "high security" concept from the ontology. The ontology can subsequently be queried for compartments that have the property "high security" associated. Such a query would result in the two compartments C_1 and C_2 from above and the ontologies contained in those compartments. The user agent may only use one of the 2 compartments concurrently if he wants to gaurantee the safety of the ontology used.

A very flexible and natural way of expressing trust and security properties of ontologies are security levels on which an ordering is imposed expressing increasing trust/security properties. For example, the ontology can specify security levels 1 to 4. Then, it defines that level l fulfills every level $l' \leq l$. As a next step, those levels are assigned to ontologies according to their trust assessment. The trust ontology can then be queried for all ontologies that meet level 1, being the ones meeting level 1, 2, 3, and 4 following the definition of security levels.

A trust ontology is used to pick the right set of ontologies that may be used to reason on a specific data type. Each data type in the assertion request may be tagged with a concept or compartment from the trust ontology. Using the reasoner, a set of applicable compartments can be found for a data type in the assertion request.

7.4.1 Ontology Compartmentalization

Problems can arise from combining ontologies, which can have unexpected security implications when reasoning on their union. For example if Ontology A defines the concept of OECD government trust level and Blog trust level and Ontology B contains the statement that Blog trust level is a subProperty of OECD government trust level.

A compartment C_i only contains ontologies which are guaranteed to be free of harmful interdependencies. Derivations may be made over ontologies from a single compartment at a time. Derivations from multiple compartments apply disjunctively. This is an important concept in order to guarantee that no ontology from an attacker is composed with a highly trusted ontology, thereby introducing vulnerabilities to the composite ontology.

One compartment of a party can, for example, contain the ontologies that the party considers highly trusted, like ones issued by their government or by their telecommunications operator. The ontology resulting from a composition of those ontologies is used for reasoning over attributes that are key to the transaction and/or whose correctness is required for legal reasons. For specific cases of high security assertions it can also make sense to only allow one ontology within a compartment that is used for reasoning on particular assertions. This is particlarly important as liability of the ontology provider is involved. Another compartment can, for example, contain ontologies tagged with "moderately trusted" that are used for doing reasoning over assertions that are not business critical and where wrong assertions would not have any legal implications. Requests for data types tagged with such ontologies correspond to "nice to have" attributes like the reputation of a weblogger that is to be transferred from one weblog to another.

Creation of a Party's Trust Ontology As a party's trust relationships are formalized by their trust ontology, ways of establishing a party's trust ontology need to be discussed. This greatly depends on whether the party is a service provider or a user, as for either case different approaches are typically used, although this separation is not mandatory.

The case of a *relying party* is simpler assuming that a system administrator has knowledge of specifying their trust ontology from scratch, by modelling their assessment of ontology providers. In special cases, a service provider might not want to rely on external assessments at all and specify their provider ontology completely on their own.

The case of a *principal* is more difficult because of the required simplicity of the approach for users. It is likely that a typical user specifies their trust ontology gradually as new trust relationships emerge. This is much similar to the concept of real-time definition of policies, that is, users are challenged with an option to enhance their trust ontology in case a new ontology is required for a particular situation.

The initial trust ontology for a user can be provided by either of the following means: (1) the trust ontology is shipped with the user-side software; (2) the trust ontology is provided by the root identity provider of the user; (3) the trust ontology is obtained from an independent trusted party like a consumer protection authority; (4) a combination of the previous approaches.

Option (1) is useful in the case where the software is trustworthy, e.g. a major, software company with a good reputation. In Option (2), the already existing trust relationship to the root identity provider can also be leveraged for the ontology provisioning. The option thus allows for the user only to choose the identity provider to be trusted and not make further decisions and thus is very user-friendly due to its transparency. A root identity provider could be a party like a large bank consortium or a governmental organization. Option (3) is useful for advanced users who can leverage the trust relationship to such a third party. However, it requires additional sophistication of the user and thus does not apply for the average user.

Access Control Policy For each data type (or, more generally, assertion request), which can be a condition over a data type, specified in the access control policy of a party, the party may specify a concept or compartment from the trust ontology. The semantics is that only ontologies from compartments that fulfill this concept are used for reasoning on how to fulfill this assertion request (user side) or whether the assertion request has been fulfilled (server side). The eligible ontology compartments then exhaustively define the subset of the party's trust relationships regarding the assertion requests at hand.

Certification of Provider Ontologies The security model considers two basic ways to ensure the integrity and authenticity of provider ontologies: (1) obtaining ontologies from trusted ontology providers, similar to obtaining keys in a PKI setting. (2) obtaining ontologies using the *Web of Trust* approach [21] without assuming the existence of single parties trusted for ontology issuance. Method (1) in a PKI setting requires that a PKI be set up and that an ontology be digitally signed by the ontology provider. Signing ontologies can impose restrictions on the ontology as discussed by Carroll [10, 11]. For RDF ontologies the requirement is that the amount of blank nodes is small as the number of canonical representations of the ontology increases exponentially with the number of blank nodes. Method (2) may be based on signed ontologies analogously with key distribution in the PGP Web of Trust. It can also be based on the assumed difficulty for an attacker to compromise a large number of locations where the same ontology is residing unsigned. Thus, by many parties vouching for the same ontology by posting it (or a hash of its canonicalization) on their Web site, trust in the ontology can be established as well. The advantage of this approach is that it does not require a PKI and can be used to create highly open identity federations. Considering the feasibility of attacking this method, it is probably not the method to use when substantial attribute security is required, although it can be useful for many use cases on today's Web.

Processing Each assertion request A in a party's security policy may carry a tag L with the trust requirements regarding the ontologies that may be used for reasoning over this assertion.

When a principal P retrieves assertion requests $A_1, \ldots, A_k$ from a server S (service provider), she also receives the tags $L_1, \ldots, L_k$ for the assertion requests and the trust ontology O_S of the service provider the tags L_i are taken from. These items together specify completely which evidence may be used to satisfy the access control policy of S.

For each condition P receives, P has to decide on how to fulfil it, e.g., by which evidence a certain condition may be fulfilled. For each condition $A_i \in \{A_1, \ldots, A_k\}$, P computes a list T_S of sets of ontologies as follows: P queries the trust ontology of the service provider (data requester in this case) for ontology compartments fulfilling the concept L_i required for A_i. Let $C_1, \ldots, C_\chi$ be the list of compartments of ontologies resulting from the query. Next, the list T_P of ontology compartments is computed analogously for the concept L'_i for the data types from the user's access control policy for her attribute data. The resulting compartments express the trust relationships of the user.

For all pairs $(C_{P,x}, C_{S,y})$ of compartments where $C_{P,x}$ is from T_P and $C_{S,y}$ is from T_S, compute the intersection of the set of ontologies in $C_{P,x}$ and $C_{S,y}$. The resulting set of ontologies is added as one set to a result list, T.[3]

[3] An even more general model of determining the ontology trusted by both parties is to do a *semantic intersection* on the ontologies, that is, an intersection on the triples of the ontology in a sense that the resulting ontology allows for all inferences that both ontologies allow for.

For each set of ontologies in the result list T, reasoning is done on the ontology that is the composition of the ontologies in the set. The reasoning results apply alternatively. Note that the number of sets in T is usually very small, and will in many typical cases be 1.

The reasoning results fulfil the (security) requirements of the party's trust ontology and also the (privacy) requirements of the requester due to construction of our method: Only ontologies that fit the trust labels in the policy or assertion request and that are considered sufficiently trusted by the respective party are being used.

7.4.2 Discussion

Ontology Intersection Typically, a principal will not have the same trusted ontologies as the relying party, but a set of ontologies that overlaps with the relying party's. This is not a problem as long as the user can compute a valid fulfillment using the intersection of the ontologies. The principal, of course, also has the choice, to opt for trusting some of the relying party's other ontologies in order to be able to compute a fulfillment of the its policy.

Using the intersection of ontologies for the reasoning on the principal side is necessary in order to prevent excessive attribute release in case the relying party does not trust the certificates the user uses as evidence for access control credentials, that is, in case the principal uses additional ontologies for reasoning, which the relying party does support.

In case the principal has certificates that the relying party's ontologies don't mention, the user will not be able to compute a proof specification fulfilling the relying party's policy.

Tradeoff Between Security and Openness There is an inherent tradeoff between security and trustworthiness of ontologies and the openness of the identity system. In case highly trusted ontologies are required for an assertion, a party may only use few ontologies from highly trusted ontology providers. This restricts the openness as many ontology providers are precluded from being considered. This naturally limits the openness in terms of possible issuers for high-security assertions. This tradeoff seems to be natural when considering trust relations in the non-electronic world: People would not trust security-critical statements coming from anyone they meet on the street, but only from people they have a strong, long-standing trust relationship with.

The other side of the tradeoff is ontologies with a moderate or low trust level that are used for attributes of less importance. Such attributes should nevertheless be provisioned in a more secure way than by uncertified assertions of an identity principal. For such attributes with less emphasis on attribute security, the party might adopt an approach of trusting a broader range of ontology providers for issuing ontologies in order to get a wider coverage in terms of possible identity providers. This provides for a more open system of attribute exchange in that a larger set of possible scenarios can be covered with the ontologies being considered. This again increases the risk that one of the ontology providers is malicious or has no well-controlled procedures in place, thus potentially leading to attesting trust to a malicious certificate issuer (e.g., himself). This would allow the attacker to issue certificates himself and thus would circumvent the security of the assertions the ontology is used for reasoning about.

Security and Privacy Considerations *Security* refers to attributes being provisioned securely, that is, attribute correctness. The security is defined through the trust in the certificate issuers. Security is mainly a concern for relying parties. Security can be jeopardized by an attacker inserting a rogue ontology into the set of trusted ontologies of the relying party. That is, to guarantee security, decisions on which ontologies to trust should be taken conservatively, thus restricting the set of trusted ontology providers and thus ontologies.

Privacy means that only relevant information is released to relying parties. Privacy can be jeopardized if a principal is tricked into trusting an ontology that establishes relations over attributes and this ontology makes the user release different attributes than she would otherwise want to release. For example, a malicious ontology could lead to a release of her SSN instead of her age. Privacy is mainly a concern of the end-users. A conservative decision on which ontologies to trust also helps to enforce the privacy of a party.

7.4.3 Example

This example assumes both governmental and private ontology providers vouching for ontologies. Assume that all OECD governments agree to accept each other's ontologies as trusted. An OECD government

issues an ontology that defines the identities vouched for by any of the OECD governments as highly trusted. Such an ontology covers all electronic identity cards, electronic passports, and other certificates issued by those governments.

For certain application domains, ontologies are likely to emerge which are generally accepted within the domain. Such domain ontologies can be restricted to a small context, e.g., a group of major mutually trusted technology weblogs. Such a domain ontology takes the burden off from each individual party to define for themselves which attributes of other parties of this set are trusted and to what degree. The agreed domain ontology captures this for them. HOwever, nothing prevents each individual site from defining further trust relations based on their individual requirements and making them available as their own trust ontology.

A domain ontology of a set of parties can be also used by other parties to be allow acceptance of certificates issued by those parties. Each party can freely choose which (domain) ontologies to trust to which degree.

A user (the identity principal) who obtains a certificate from their local government G, is also provided the ontologies issued by their OECD government. This ontology will typically have high trust associated with it. The ontology is signed by the government, thus cryptographically binding the identity of its provider with the ontology. The ontology resides in its own compartment to prevent other ontologies from injecting (rogue) statements.

Additionally, the user participates in a technology weblog B, and has an excellent reputation of making high-quality contributions. Eventually she would like to leverage this reputation in another technology blog C in order to get better privileges like being able to act as a moderator immediately without building up a reputation there over time. Thus, she obtains a certificate from B that asserts her good reputation. In addition, she obtains a domain ontology covering some of the major technology weblogs on the Web and making trust statements on the certificates they issue. This ontology is also provided by B and contains B as one of the trusted issuers of reputation certificates.

A service provider (the relying party), in our example, party C, also obtains an ontology from their local OECD government (or any government they trust more) and assigns it a high trust level. Typically, they would choose the ontology of the government of the country whose legislation they are subject to, as they are relying on the ontology in order to make trust decisions for attributes with legal relevance. Liability issues could even mandate that the ontology from their local government be used. Additionally, C obtains the domain ontology for technology blogs and checks whether it agrees with it. The service provider may also obtain other domain ontologies from similar domains to have a better coverage of other weblogs. Party C associates the domain ontology with a moderate trust class in order to reflect its trustworthiness appropriately. The government ontology is defined to reside in one compartment without the possibility of composing other ontologies with it during the reasoning.

When the user makes a service request to the service provider, the service provider replies with an assertion request derived from their access control policy and their trust ontology.[4]

Assume party C requires the user to provide their name, address, and SSN where high attribute security must be guaranteed by requiring the compartmentalized government-issued ontology to be used. C also requests the certified reputation where moderate security is sufficient, thus the domain ontology issued by B may be used; this ontology happens to cover party B's certificate asserting the user's reputation with weblog B.

The user computes the composed ontology for her civil identity attributes as the government-issued one. This is the only composed ontology applicable. For the reputation attribute the domain ontology issued by B turns out to be the only applicable ontology. Using those ontologies guarantees that the user will not consider identity provisioning mechanisms and issuers not trusted by the server.

The user computes a fulfillment of the policy that is consistent with what the server will accept by deriving possible protocols and identity providers from the determined ontologies. She then provides an appropriate assertion and proof to the server. The server does the reasoning on his ontologies and accepts the proof given that the user has computed everything correctly.

8 Concrete Syntax and Implementation

We now describe a complete implementation, concrete syntax and data flow for integration of this system with an access control system. As concrete syntax, we propose to use a combination of multiple existing

[4]Note that this does not work in a scenario where the relying party is to stay anonymous. In such a scenario a different approach of including further negotiation of commonly trusted ontologies would have to be taken.

standards for implementation as shown below.

8.1 Standards used

We use the following standard syntaxes to implement the above model.

- *RDF* [22] is used for expressing assertions. RDF is able to express predicates of arbitrary arity and is expressly designed to allow for flexible semantics. It also integrates with OWL, the ontology language used in this solution.
- *SPARQL* [26] Is a query language specifically designed for querying over RDF assertions in an assertion store.
- *OWL* [15] is used to describe the semantics of certificates and assertions and to allow for reasoning in query matches. That is, rather than matching only the available assertions or evidence, the system is able to match over an expanded space of assertions and evidence, inferred using the ontologies applicable following the trust model.

8.2 Reasoning Over Ontologies

The knowledge provided by the ontology allows the access-control engine to reason about policies expressed over abstract properties of certificates. For example, a policy can specify that an assertion of the user's name must be backed up by evidence in the form of a government-issued X.509 certificate, without specifying a list of certificates which satisfy this condition. This requires a matching of abstract properties within policies against abstract properties attributed to certificates.

In our architecture, abstract properties of certificates are not provided directly by certificates. Instead, they are inferred from a certified mapping between public keys and classes within the certification ontology. That is, the public keys are defined as instances of the certification ontology classes. This makes a clean separation between certificates and their metadata with minimum invasion into the certification infrastructure.

The Personal Information ontology architecture also allows matching of data types by inference rather than standard string matching. For example, age > 18 can match age $= 23$, Maritalstatus = married can match name.Prefix ="Mrs" etc...

8.3 Base Assertions for Anonymous Credentials

Anonymous credentials (private certificates) securely store a set of assertions which may be proven if required, but are able to prove less informative derivations from these assertions. The knowledge provided by an assertion also includes the set of all possible entailments from that assertion. For example, a certificate storing the assertion age $= 23$ can prove age > 18, age > 12 etc... Passport or driving licence credentials are able to prove attributes within the passport, but are also able to prove the derived assertion that the possessor is the holder of the passport or driving licence.

We distinguish an assertion from its possible entailments by calling the assertion the *Base Assertion*. A Base Assertion, A_b relates to the other assertions in its entailment set, $\mathbf{E}$ as follows:

$$\forall(A_i) \in \mathbf{E} : (A_b \Rightarrow A_i), \forall(A_i, i \neq b) : \neg(A_b \Rightarrow A_i)$$

We call the maximal assertion in a set of possible assertions the *Base Assertion*. The Base Assertion Set of a certificate is the set of maximally informative assertions which can be proved by each certificate, none of which entail any others. Only the base assertions (e.g. age $= 23$) are stored in the data store. This means that queries can be made on the assertion store which do not request the maximally informative assertions, but they will always be satisfied if they can be proven by the certificate. For example the query

```
SELECT ?cert
WHERE
{
 ?cert Prime-Cert:evidence  ?graph.
     GRAPH ?graph {
       ?x
         Prime-PII:User.age ?age.
         FILTER(?age > 17)
     }
}
```

will be satisfied by the Base Assertion (in N3) `User age 23`.

This does not mean that the certificate will necessarily be used to prove the Base Assertion, it just tells the query engine that it can prove the less informative assertion. Most importantly, it means that the credential metadata (certificate metadata) is able to determine whether a given query may be satisfied by the credential verification engine.

8.4 Architectural Components

- An assertion and evidence knowledge base. This is a database storing metadata about assertions which can be proven by the available certificates. The *Base Assertions* are grouped into RDF named graphs [9]. A link table stores the named graphs (conjoined groups of assertions) and credentials which are able to prove those named graphs. Credentials which may be used as evidence for those assertions refer to the named graphs. The credential store stores the certifier name, public key and evidence, e.g. a signature.

  Queries to the assertion store will be satisfied if the Base Assertion can be proven, however if the query requests a subset of the knowledge provided by the base assertion, the request proof will then be made for the less informative query, but on the basis of the earlier query, it is then known that this can be proven by the available credentials. This even applies to boolean queries such as a query for the possession of a Passport OR a Driverslicence.

  Note that the evidence stored is only a public key and a certificate. Only when a decision is made on the evidence are the required inferred properties derived from the stored public key and matched against the abstract properties in the policy.

- The (private) certificates themselves.

- A certified mapping between public key instances and ontology classes (the provider ontology).

- An ontology of abstract properties. That is the certification ontology detailed above.

- A Disclosure Decision module. If more than one certificate is able to satisfy an assertion request, this module decides on the basis of user-defined preferences, which one should be used. For example it might be preferable to prove the user's name using a self-signed certificate rather than a government-issued passport certificate, which may, by inference, also reveal attributes such as nationality.

- A crypto engine, which provides proofs based on assertion requests, if those proofs are available from stored certificates. The query to the assertion and evidence store determines which certificates should be asked to verify a query. The query is then passed to the proof engine to provide the proof for the requesting party.

8.4.1 Link between Evidence and Assertions

Both assertions and evidence (in Abstract Syntax $\langle A(P_i(N_j)), E(A)\rangle$) are expressed in terms of RDF graphs (stored in an RDBMS). In order to connect evidence to assertions, each assertion is assigned to a named graph [12]. There may be several assertions in one named graph (meaning a credential proves a conjunction) or just one. Graphs may also be UNION graphs if a disjunction is proven (see later examples).

Conjunction and disjunction between assertions is expressed by multiple graphs (conjunctions) and UNION graphs (disjunctions). We use JENA's rule language to express ontology inference rules.

For example the following graphs (taken from the example in Section 8.6.2) show a certificate applied to the user's age.

Default Graph

```
<http://www.example.org/Cert#123>
      <http://www.example.org/Cert#evidence>
              <http://www.example.org/DB#Graph1> .
```

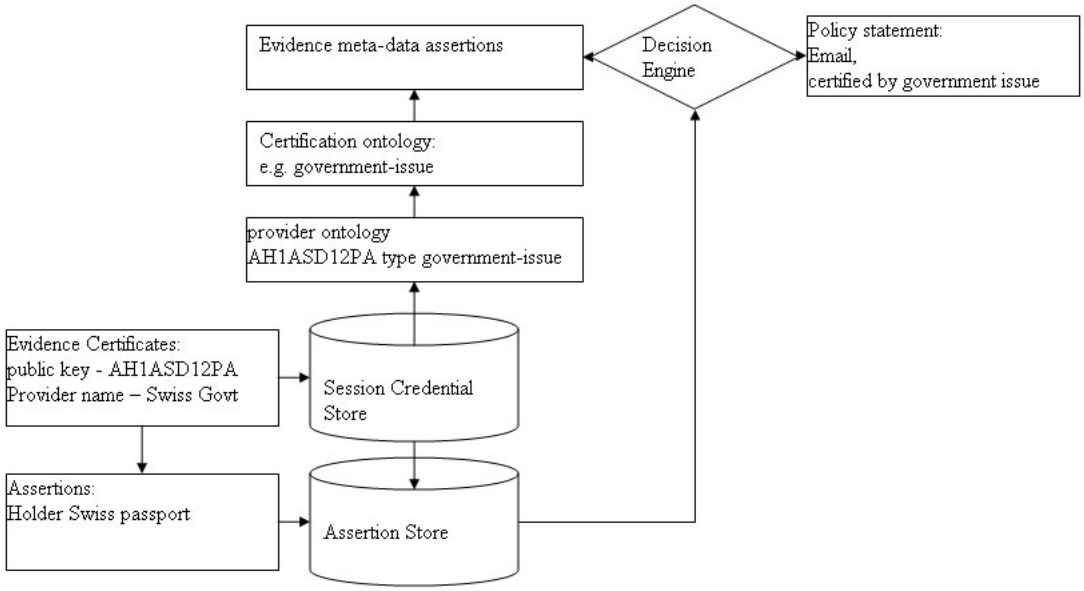
Graph1

```
<http://www.example.org/DB#123>
      <http://www.example.org/PII#User.age>
              "21"^^<http://www.w3.org/2001/XMLSchema#positiveInteger> .
```

A query for such a credential uses SPARQL's named graph syntax, for example:

```
SELECT ?cert
WHERE
{
?cert Prime-Cert:evidence  ?graph.
     GRAPH ?graph {?x  Prime-PII:User.age ?age.}
}
```

This means to look for a certificate applied to the user's age. This syntax provides a very flexible way of querying assertions and evidence. The model mirrors very precisely the semantics of the transaction.



8.5 Data flow

1. Assertion requests are sent over the wire using an XML language which allows integration with XACML and SAML. This is then translated by the receiving API into SPARQL for execution over the RDF graphs.

   EXAMPLE

   ```
   <AssertionRequest>
     <hint>
       <group>
         <condition>
           <predicate>
             <name>greaterThan</name>
             <argument isLiteral="false">age</argument>
             <argument>18</argument>
           </predicate>
         </condition>
       </group>
     </hint>
   </AssertionRequest>
   ```

   Converted to:

```
SELECT ?cert
WHERE
{
 ?cert Prime-Cert:evidence  ?graph.
      GRAPH ?graph {
        ?x
          Prime-PII:User.age ?age.
          FILTER(?age > 17)
      }


}
```

2. The SPARQL SELECT query is run over the Base Assertions and credentials in the Knowledge Base to see if there are any credentials proving the requested assertion. E.g. if predicate in the XML wire protocol is $Userage > 18$, then a Base Assertion of $age = 23$ will prove this.

3. For the list of credentials found in the knowledge base metadata, the Disclosure Decision module is used to decide which credential is actually used. Then ONLY the initial assertion request is sent to the prover to prove that assertion. E.g. if the initial assertion request asks for $age > 18$ then the proof engine is asked to prove that thue user's $age > 18$ using the selected credential from the search in the previous step, even though the query was satisfied by user's age=23 in the assertion store.

4. If no credentials are available, then the query is run over any plain assertions available and these are returned.

5. Assertions and evidence are returned to the requester in the XML wire format, or the interaction is terminated if no satisfactory assertions are found.

8.6 Worked Examples

We provide a number of examples that have been implemented in order to show the applicability of our approach.

8.6.1 Subsumption reasoning

Assertion Request The following is the part of the assertion request which might express a condition on the certification expression.

Wire Format

```
<AssertionRequest>
  <hint>
    <group>
      <evidence>
        <certification>
          <type>Prime-Cert:governmentIssueCertificate</type>
        </certification>
      </evidence>
    </group>
  </hint>
</AssertionRequest>
```

Assertion Request Converted to SPARQL query Note that in all further examples we have omitted the namespace declarations.

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
PREFIX PII:  <http://www.example.org/PII#>
PREFIX Cert: <http://www.example.org/Cert#>
PREFIX DB:   <http://www.example.org/DB#>

SELECT ?cert WHERE{
?cert  Cert:evidence ?assertion1.
  GRAPH ?assertion1{
```

```
    ?certKey rdf:type Cert:governmentIssueCertificate.
   }
}
```

Evidence Graph (Default Graph) Note that all graphs are expressed in N3 [16] notation.

```
<http://www.example.org/Cert#123>
      <http://www.example.org/Cert#evidence>
              <http://www.example.org/DB#Graph1>.

<http://www.example.org/Cert#123>
      <http://www.example.org/Cert#key>
              <http://www.example.org/DB#certificate1Key> .
```

Assertions In this case, the assertions are not relevant in this example.

Provider Ontology

```
certificate1Key rdf:type PII:SwissDriversLicence
```

Certification Ontology The following triples are the relevant part of the certification ontology.

```
PII:SwissDriversLicence rdfs:subClassOf PII:DriversLicence
PII:DriversLicence rdfs:subClassOf PII:GovernmentIssueCertificate
```

Inference Model The inference model contains, at minimum, the following assertion deduced from subsumption reasoning over the Provider and Certification Ontologies:

```
certificate1Key rdf:type GovernmentIssueCertificate
```

If the query is now run on the inference model, and the assertion-evidence tuple, it will find evidence satisfying the request.

8.6.2 Operator Reasoning

Assertion Request

```
<AssertionRequest>
   <hint>
     <group>
       <condition>
         <predicate>
            <name>greaterThan</name>
            <argument isLiteral="false">age</argument>
            <argument>18</argument>
         </predicate>
       </condition>
     </group>
   </hint>
</AssertionRequest>
```

Converted to SPARQL Query

```
SELECT ?cert
WHERE
{
    ?cert Cert:evidence  ?graph.
       GRAPH ?graph {
           ?x  PII:User.age ?age.
           FILTER(?age > 17)
        }

}
```

Assertions The following assertions will be available to the query engine:

Default Graph

```
<http://www.example.org/Cert#123>
      <http://www.example.org/Cert#evidence>
              <http://www.example.org/DB#Graph1> .
```

Graph1

```
<http://www.example.org/DB#123>
      <http://www.example.org/PII#User.age>
              "21"^^<http://www.w3.org/2001/XMLSchema#positiveInteger> .
```

This comes from a simple RDBMS table as shown in Section 8.6.6. The data flow is then as follows:

- Assertion request converted to run SPARQL query over RDBMS graphs.
- SPARQL query returns credential based on Base Assertions stored in RDBMS.
- Proof engine asked to prove precise assertions requested using assertion request language.

Inference Model Not required.

8.6.3 Combination of Inferred Assertions and Operator Reasoning

Assertion Request

Wire Format Assertion Request

```
<AssertionRequest>
  <hint>
    <group>
      <condition>
        <predicate>
          <name>greaterThan</name>
          <argument isLiteral="false">age</argument>
          <argument>17</argument>
        </predicate>
      </condition>
    </group>
  </hint>
</AssertionRequest>
```

Assertion Request Converted to SPARQL Query

```
SELECT ?cert
WHERE
{
 ?cert Cert:evidence  ?graph.
     GRAPH ?graph {
       ?x
         PII:User.age ?age.
     ?age jena:greaterThan ?value
         FILTER(?value > 17)
     }

}
```

Assertions The following assertions will be available to the query engine:

```
<http://www.example.org/DB#123>
      <http://www.example.org/PII#holder>
              <http://www.example.org/PII#BritishDriversLicence> .
```

Datatype Ontology The datatype ontology contains the rule:

```
?user PII:holder ?y, ?y rdf:type PII:BritishDriversLicence. ->
?user PII:User.age B_Node_Age.
B_Node_Age jena:greaterThan "17"^^xsd:integer
```

Inference Model The inference model then contains the assertions:

```
B_Node_User  PII:User.age B_Node_Age,
B_Node_User type PII:User,
B_Node_Age jena:greaterThan  "17"^^xsd:integer
```

Clearly the query will match the assertions in the inference model. The assertion from the assertion request is then passed to the proof engine to prove it.

8.6.4 Assertion Conjunction

This architecture solves the following problem of how to require that the same evidence is applied to two different assertions (as in the case of an ecoin and currency). This can be done simply by using the same evidence applied to a conjunction of predicates:

Wire Format Assertion Request

```
<AssertionRequest>
  <hint>
    <group>
      <condition>
        <predicate>
          <name>equals</name>
          <argument isLiteral="false">&PII;eCoinValue</argument>
          <argument isLiteral="true">200</argument>
        </predicate>
      <condition/>
      <condition>
      <predicate>
        <name>equals</name>
        <argument isLiteral="false">
          &PII;eCoinCurrency
        </argument>
        <argument isLiteral="false">&Cert;Euro </argument>
      </predicate>
    </condition>
    <evidence>
      <certification>
        <type>&Cert;eCoin </type>
      </certification>
    </evidence>
   </group>
  </hint>
</AssertionRequest>
```

Assertion Request Converted to SPARQL Query

```
SELECT ?cert
WHERE{
    ?cert  Cert:evidence  ?graph1 .
    GRAPH ?graph1{
    ?ecoin  &PII;eCoinCurrency Cert:EUR ;
    ?ecoin   Cert:amount    ?amount .
    FILTER ( ?amount = 200 )
   }
 }
```

Assertions The following assertions are available as Jena DataSource objects. The semantics is that the credential expressed by the default graph is evidence for the assertions in Graph1
Default Graph

```
<http://www.example.org/Cert#123>
      <http://www.example.org/Cert#evidence>
              <http://www.example.org/DB#graph1> .
```

Graph1

```
<http://www.example.org/DB#234>
      <http://www.example.org/PII#ecoinValue>
              "200"^^<http://www.w3.org/2001/XMLSchema#positiveInteger> ;
      <http://www.example.org/PII#ecoinCurrency>
              <http://www.example.org/Cert#EUR> .
```

The query should answer yes iff the same ecoin is used to prove both assertions. That ecoin will then be selected for the proof and the proof engine asked to prove the conjoined assertions asked by the assertion request. Note that in this case, the verifier should remove assertions which are no longer valid according to the limited-show protocol.

8.6.5 Assertion Disjunction

Wire Format Assertion Request

```
<AssertionRequest>
  <hint>
    <group>
      <condition>
        <predicate>
          <name>&PII;credentialHolder</name>
          <argument isLiteral="false">&Cert;DriversLicence</argument>
         </predicate>
      </condition>
    <evidence>
      <certification>
        <type>&Cert;Idemix </type>
      </certification>
    </evidence>
    </group>
  </hint>
  <hint>
    <group>
      <condition>
        <predicate>
          <name>&PII;credentialHolder</name>
          <argument isLiteral="false">&Cert;Passport</argument>
        </predicate>
      </condition>
    <evidence>
      <certification>
        <type>&Cert;Idemix </type>
      </certification>
    </evidence>
    </group>
  </hint>
</AssertionRequest>
```

Assertion Request Converted to SPARQL Query

```
SELECT ?cert ?cert1
WHERE
  { ?cert  Cert:evidence  ?graph .
    GRAPH ?graph
      {
        {?x  PII:credentialHolder  Cert:DriversLicence .}
          UNION
        { ?x  PII:credentialHolder  Cert:Passport .}
      }
  }
```

Assertions Just the following assertions are available.
Certification

```
<http://www.example.org/Cert#123>
      <http://www.example.org/Cert#evidence>
              <http://www.example.org/DB#graph1> .
```

Graph1

```
<http://www.example.org/DB#123>
      <http://www.example.org/PII#credentialHolder>
              <http://www.example.org/Cert#DriversLicence> .
```

Only one assertion is required to be present to satisfy the query. The proof engine will however be asked to prove the complete disjunction ($A \vee B$), not one or the other. I.e. it will be asked to prove the assertion that the user has a drivers' licence OR a passport, but not reveal which.

8.6.6 Disjunction and Conjunction

Wire Format Assertion Request

```
<AssertionRequest>
  <hint>
    <group>
      <condition>
        <predicate>
          <name>&PII;credentialHolder</name>
          <argument isLiteral="false">&Cert;DriversLicence</argument>
        </predicate>
      </condition>
      <evidence>
        <certification>
          <type>&Cert;Idemix </type>
        </certification>
      </evidence>
    </group>
    <group>
      <condition>
        <predicate>
          <name>equals</name>
          <argument isLiteral="false">&PII;ecoinValue</argument>
          <argument isLiteral="true">200</argument>
        </predicate>
      <condition/>
      <condition>
        <predicate>
          <name>greaterThan</name>
          <argument isLiteral="false">
            &PII;ecoinCurrency
          </argument>
          <argument isLiteral="false">&Cert;Euro </argument>
        </predicate>
      </condition>
      <evidence>
        <certification>
          <type>&Cert;idemix </type>
        </certification>
      </evidence>
    </group>
  </hint>
  <hint>
    <group>
      <condition>
        <predicate>
          <name>&PII;credentialHolder</name>
          <argument isLiteral="false">&Cert;Passport</argument>
        </predicate>
      </condition>
      <evidence>
        <certification>
          <type>&Cert;Idemix </type>
        </certification>
      </evidence>
      </group>
      <group>
        <condition>
          <predicate>
            <name>equals</name>
            <argument isLiteral="false">&Cert;certificateValue</argument>
            <argument isLiteral="true">200</argument>
          </predicate>
```

```
        <condition/>
        <condition>
          <predicate>
            <name>greaterThan</name>
              <argument isLiteral="false">&PII;ecoinCurrency </argument>
            <argument isLiteral="false">&Cert;Euro </argument>
          </predicate>
        </condition>
      <evidence>
        <certification>
          <type>&Cert;idemix </type>
        </certification>
      </evidence>
    </group>
  </hint>
</AssertionRequest>
```

Assertion Request Converted to SPARQL Query

```
SELECT ?cert ?cert1
WHERE
  { ?cert  Cert:evidence  ?graph .
    GRAPH ?graph
      {   { ?x  PII:credentialHolder  Cert:driversLicence .}
        UNION
          { ?x  PII:credentialHolder  Cert:passport .}
      }
    ?cert1  Cert:evidence  ?graph1 .
    GRAPH ?graph1
      { ?x1  PII:ecoinCurrency  Cert:Euro ;
             PII:ecoinValue    ?amount ;
     rdf:type  Cert:Idemix.
        FILTER ( ?amount = 200 )
      }
  }
```

Assertions The SELECT would find an exact match for these assertions based on an RDF translation of the following RDBMS tables:

1	userID	PII:holder	Cert:passport
2	ecoin	PII:ecoinCurrency	Cert:Euro
3	ecoin	PII:ecoinValue	"200"

Graphs

GraphID	AssertionID
1	1
2	2
2	3

Certificates

EvidenceID	Key	Keyname	Value	appliesToAssertionGraph
1	aaa	bbb	AH123AREWASDAW1	1
2	aaa1	bbb1	AH123ASDQWDQDAW12	2

That is the following 2 graphs:
Default Graph

```
<http://www.example.org/Cert#123>
      <http://www.example.org/Cert#evidence>
              <http://www.example.org/DB#graph1> .
<http://www.example.org/Cert#234>
      <http://www.example.org/Cert#evidence>
              <http://www.example.org/DB#graph2> .
```

Graph1

```
<http://www.example.org/DB#234>
      <http://www.example.org/Cert#amount>
              "200"^^<http://www.w3.org/2001/XMLSchema#positiveInteger> ;
      <http://www.example.org/PII#ecoinCurrency>
              <http://www.example.org/Cert#EUR> .
```

Graph2

```
<http://www.example.org/DB#123>
      <http://www.example.org/PII#credentialHolder>
              <http://www.example.org/Cert#driversLicence> .
```

The examples in this section show that our reasoning architecture can be realized using standard methods and available tools.

9 Conclusion

We put forth a framework and architecture for assertion and metadata exchange for digital identity management. We presented both an abstract and concrete syntax and semantics for assertion requests and assertions, both comprising the actual data and metadata describing evidence for proving an assertion. We introduced a trust model describing how parties can obtain ontologies securely, thus facilitating open attribute exchange. We present an architecture for our framework for both relying parties and principals. We presented how a reasoning engine can be used to make decisions on the satisfaction of an access control policy and for supporting the decision on how to fulfill an assertion request. We have shown the real-world applicability by implementing the reasoning architecture and presenting multiple examples for the reasoning.

Our work represents a substantial improvement to the current approach to identity management in terms of having a highly flexible way of expressing assertion requests and assertions, in particular the ability of handling general trust and certification metadata. The general languages, the integration with the reasoning framework, and the model for ontology security we propose allows for open identity federations.

References

[1] Backes, M., Camenisch, J., and Sommer, D. Anonymous yet accountable access control. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society* (New York, NY, USA, 2005), ACM Press, pp. 40–46.

[2] Bangerter, E., Camenisch, J., and Lysyanskaya, A. A cryptographic framework for the controlled release of certified data. In *Twelfth International Workshop on Security Protocols 2004* (2004), Springer Verlag.

[3] Bonatti, P., and Samarati, P. A unified framework for regulating access and information release on the web. *Journal of Computer Security 10* (2002), 241–272.

[4] Business Week. Business week/harris poll: A growing threat. available at `http://businessweek.com/2000/00_12/b3673010.htm`.

[5] Camenisch, J., and Lysyanskaya, A. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In *Advances in Cryptology — EUROCRYPT 2001* (2001), B. Pfitzmann, Ed., vol. 2045 of *LNCS*, Springer Verlag, pp. 93–118.

[6] Camenisch, J., and Lysyanskaya, A. A signature scheme with efficient protocols. In *Third Conference on Security in Communication Networks* (2002), G. Persiano, Ed., vol. 2576 of *LNCS*, Springer Verlag, pp. 274–295.

[7] Camenisch, J., and Lysyanskaya, A. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology — CRYPTO 2004 (to appear)* (2004), LNCS, Springer Verlag.

[8] Camenisch, J., Sommer, D., , and Zimmermann, R. A general certification framework with application to privacy-enhancing certificate infrastructures. In *Proceedings of SEC 2006* (2006), Springer Verlag.

[9] Carroll, J. Named graphs. `http://niap.bahialab.com/cc-scheme/`.

[10] Carroll, J. J. Signing rdf graphs. In *Proceedings of the 2nd ISWC* (2003), vol. 2870 of *LNCS*, Springer.

[11] Carroll, J. J. Signing rdf graphs. Tech. Rep. HPL-2003-142, Hewlett Packard Research, 2003.

[12] Carroll, J. J., Bizer, C., Hayes, P., and Stickler, P. Named graphs, provenance and trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web* (New York, NY, USA, 2005), ACM Press, pp. 613–622.

[13] Common criteria. `http://niap.bahialab.com/cc-scheme/`.

[14] Cranor, L. F., Reagle, J., and Ackerman, M. Beyond concern: Understanding net users' attitudes about online privacy. Tech. Rep. Research Technical Report TR 99.4.3, AT&T Labs, April 1999.

[15] Dean, M., Schreiber, G., and , eds. Owl web ontology language reference. W3C Recommendation, February 2004.

[16] (ed.), T. B.-L. Notation 3. `http://www.w3.org/DesignIssues/Notation3`.

[17] European Parliament. Directive 95/46/EC of the European Parliament and of the council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. In *Official Journal L* (1995), vol. 281.

[18] Forrester. Online consumers fearful of privacy violations, October 1999. http://www.forrester.com/ER/Press/Release/0,1769,177,FF.html.

[19] Gehrmann, C., and Mats Nä slund, e. Ecrypt yearly report on algorithms and keysizes. Tech. Rep. D.SPA.16, Ecrypt Network of Excellence, 2005. `http://www.ecrypt.eu.org/documents/D.SPA.16-1.0.pdf`.

[20] Golbeck, J., and Parsia, B. Trust network-based filtering of aggregated claims. *Int. J. Metadata, Semantics and Ontologies 1*, 1 (2006).

[21] Golbeck, J., Parsia, B., and Hendler, J. A. Trust networks on the semantic web. In *Procedings of the CIA 2003* (2003), pp. 238–249.

[22] Manola, F., Miller, E., and ed. RDF Primer. Recommendation, World Wide Web Consortium, February 2004. `http://www.w3.org/TR/rdf-primer`.

[23] National Institute of Standards and Technology. Digital Signature Standard (DSS). Federal Information Processing Standards Publication 186-2, 2000.

[24] Palen, L., and Dourish, P. Unpacking "privacy" for a networked world. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2003), ACM Press, pp. 129–136.

[25] Pfitzmann, A., and Hansen, M. Anonymity, unlinkability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology. Tech. rep. available at `http://www.freehaven.net/anonbib/cache/terminology.pdf`.

[26] Prud'hommeaux, E., and Seaborne, A. Sparql query language for rdf. W3C Working Draft.

[27] Rivest, R., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, 2 (Feb. 1978), 120–126.

[28] Sheehan, K. Toward a typology of internet users and online privacy concerns. *The Information Society 18* (2002).

[29] Wang, X., Yin, Y. L., and Yu, H. Collision search attacks on sha1, February 2005. available from `http://theory.csail.mit.edu/~yiqun/shanote.pdf`.

[30] Whitten, A., and Tygar, J. Why johnny can't encrypt: A usability evaluation of pgp 5.0. In *Proceedings of the 9th USENIX Security Symposium, August 1999* (1999).

A Overview of the Ontology

This section sketches the main concepts of our ontologies for expressing certification properties. We note that the ontologies are not complete real-world-capable ontologies, but convey the main ideas of our approach.

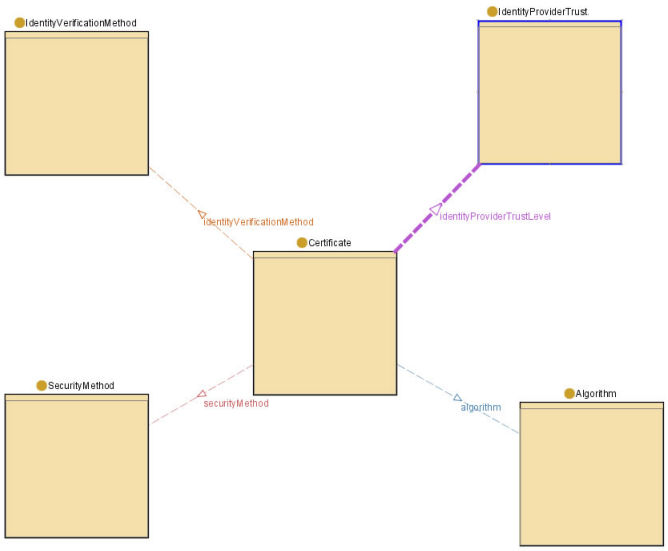


Figure 1: Certificate ontology – high-level concepts.

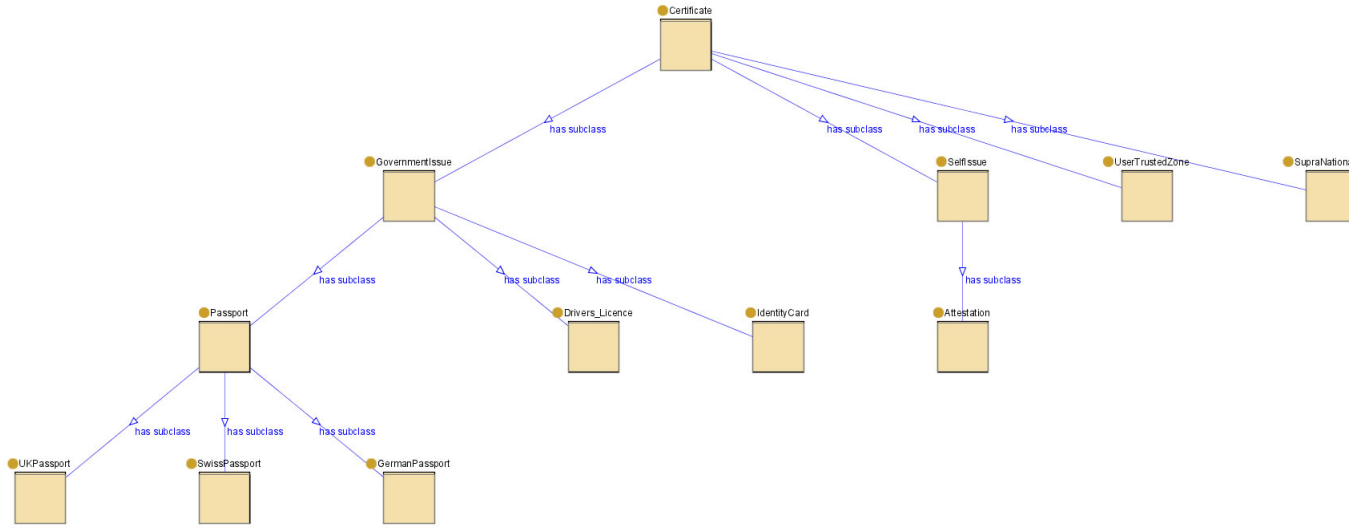


Figure 2: Ontology for certificates.

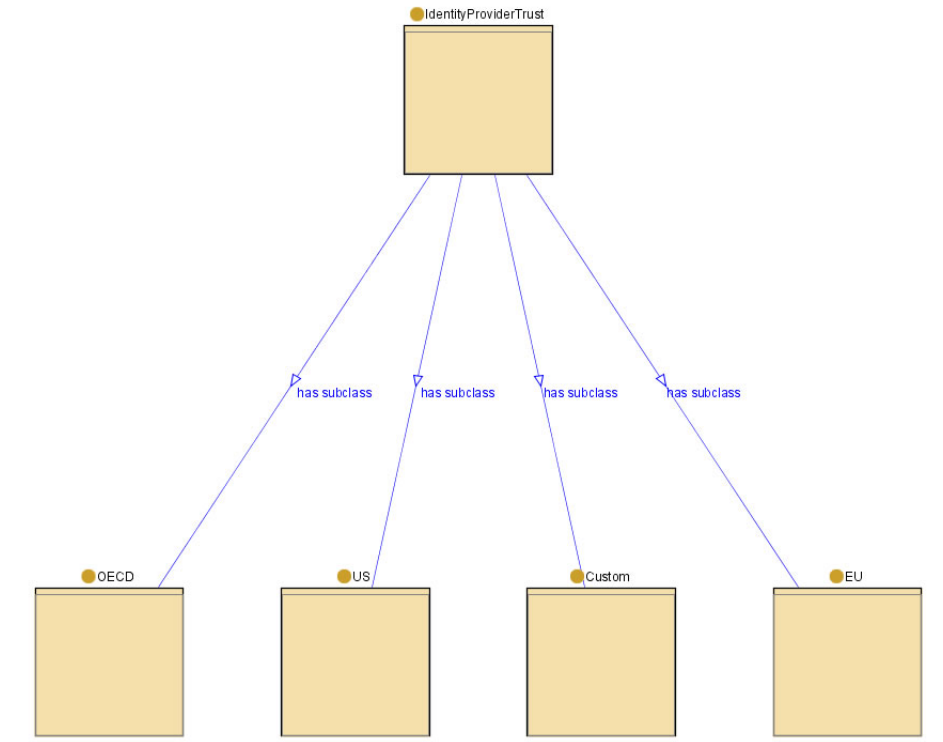


Figure 3: Identity provider trust.

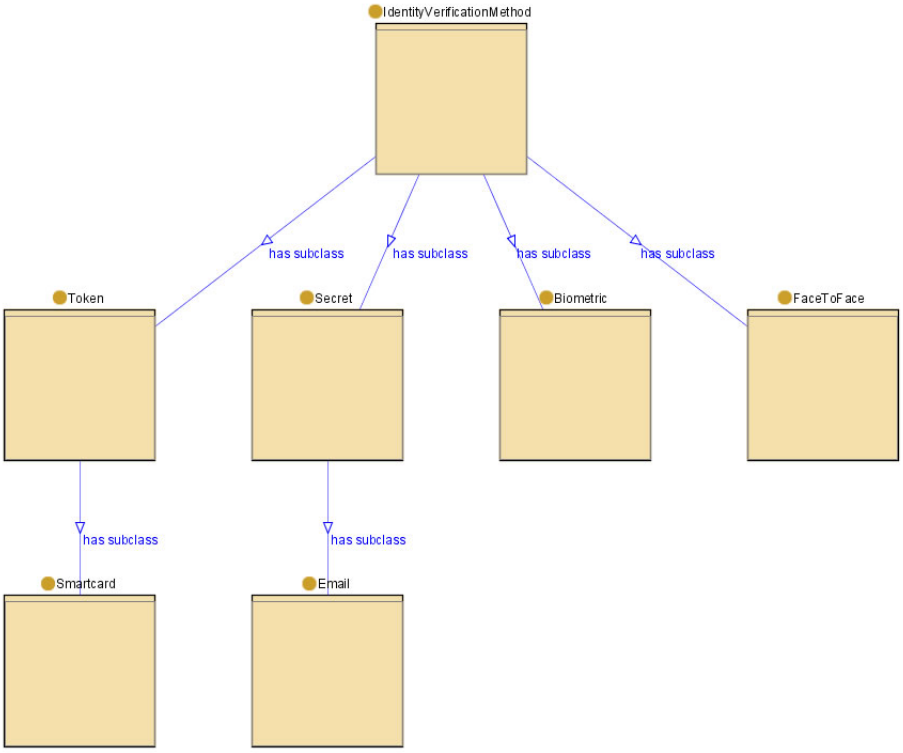


Figure 4: Identity verification methods.