RZ 3720 (# 99730) 11/17/08
Computer Science 14 pages

Research Report

Semantic Digital Signatures

Daniela Bourges-Waldegg, Christian Hörtnagl and James Riordan

IBM Research GmbH
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

{dbw, hoe, rij}@zurich.ibm.com

Semantic Digital Signatures

Daniela Bourges Waldegg, Christian Hörtnagl, James Riordan
{dbw,hoe,rij}@zurich.ibm.com

Abstract

We discuss the need of addressing, in a uniform way, digital signatures with rich semantics, for enabling increased automation of signature processing. We present a scheme for combining digital signatures with the formal and extensible semantics of the Semantic Web, as a standard layer in applications using digital signatures. We introduce several constructions and processes towards realizing this end, a new class of attack against semantically enabled applications, and measures of avoiding this attack class. Finally, we simplify a few existing processes that use digital signatures by expressing them in terms of semantic signatures.

1. Introduction

As mathematical constructions, digital signatures provide a mechanism for creating and verifying an association between serialized content and an asymmetric key pair. These associations are used to emulate the signatures and stamps of the physical world. Ultimately, such signatures and stamps have a *meaning* within a specific context, and subsequent actions within that context are based upon this meaning.

In physical documents, the meaning of a signature being created or verified is derived implicitly (by the involved people) from the content of the document. Asserting and deriving meaning is often aided by additional qualifications of the signature, such as its placement on the page. For example, the publication clearance form for this paper has specified areas for the authors and the clearing agents to provide additional information about the place and date of the signature, the intended conference and so forth. Other forms require a handwritten assertion that the signer has read and understood the document, in addition to the signature itself. The fact that completion and signing of such a form is a process performed by humans is not problematic because it is one of many human processes performed on the document.[1] The fact that there exist no uniform way of conveying meaning is also not a major problem, as humans can adapt to interpreting signatures in new ways.

In current digital signature schemes (that is, the collection of technologies that gravitate around cryptographic signatures), the meaning of a signature is usually determined in two scales of granularity. At the coarse scale, meaning is derived from a combination of the application used, conventions concerning keys, and static usage policies. For example, in an X.509 context, it is understood that some keys are for negotiating session keys with a server, some are for signing email, some are for signing code, some are for signing (certifying) other keys, *et cetera*. At the fine scale, meaning is derived as with physical signatures: implicitly from the meaning of the content, although generally without additional qualifications, analogous to placement or structured fields. As with physical signatures, current digital signature technologies do not provide a uniform way of conveying meaning.

In applications using digital signatures, processes that require action based on the *meaning* of a signature fall into one of three categories:

1) A process driven by a human. Such processes are flexible but expensive and often error prone in all but the simplest cases. An example of this category is the signature of a PDF file performed using the Adobe signing framework [1], which allows the creator to enter a natural language description of the reason for signing; verifying the validity of the signature according to a policy requires human inspection.
2) A process that is sufficiently common that a concrete instance has been specified, standardized, and implemented. An example of this category is the application of digital signatures to associate an entity with a public key, that is certification.

1. While check clearing systems might seemingly provide a counter-example, contested signatures will be reviewed by a human. In any case, we endeavor to address the general case of signatures which might be aided by machine processing, rather that specific uses.

3) A process that, while not so common as a process of the second category, is yet automated. These tend to be implemented in an *ad hoc* and incomplete fashion, lacking formal specification (which makes sense) and standardization (which does not). Examples of this category include the myriad tools that check a signature of a file before conditionally executing or installing it.

This paper is predicated upon the thesis that these existing processes may be greatly enhanced, and their pitfalls avoided, via the addition of a formal semantics layer to traditional digital signatures, as part of standards and application programming interfaces (APIs). This particularly benefits automatic processing of digitally signed content. We propose expressing the meaning of a digital signature using Semantic Web technologies, that is, standard and well established, application-independent formalisms for representing knowledge. Our core contribution is the definition of a digital signature scheme based on this principle, with constructs that bridge between the semantic and the cryptographic world, building on top of traditional digital signatures.

In the next section we further motivate our work, by analyzing the three categories above, and the benefits of extending the traditional digital signatures they currently use with such formal semantics.

2. Foundational observations

Similar to *Greenspun's Tenth Rule*,

> "Any sufficiently complicated C or Fortran program contains an *ad hoc*, informally-specified bug-ridden slow implementation of half of Common Lisp."

any sufficiently complex application involving digital signatures contains an *ad hoc*, poorly specified, obscure, inextensible semantic specification.

In many realms, automatic processing is not currently the norm but is highly desirable. For instance, signing a written document as its author may implicitly mean that the author vouches for the correctness of the information presented, for the appropriateness of the views expressed, or for the originality of the document. By contrast, signing the same document as the copy editor raises the expectation that the signer vouches for the existence of all sources and for the fact that spelling and grammar mistakes have been removed. This category of signature applications is increasingly important in several areas such as integrity and auditability of business processes, general compliance checking and distributed command and control. In these areas, complex rule sets must be applied to large volumes of content. It is economically infeasible and error prone to manually check that the rules are being followed.

In realms where automatic processing is already the norm, the semantic specifications were designed to solve a specific problem rather than to provide a general semantic framework (which would have been out of scope). As such, they provide only sufficient capability to solve the problem as initially conceived, and tend not to be extensible. This has the effect that, as one of the following evolves

- the need that the process addresses,
- understanding of the need, or
- understanding of the process,

the deployed code and configuration base inhibit evolution of the process itself.

X.509, as an application of digital signatures, provides one of the best illustrations of this point. In X.509 it would be desirable to clarify certain attributes, to add information concerning identity evidence offered by the key owner to the certificate authority, to enable privacy features, or to expand to new domains (see 3.1). The lack of extensible semantic support in the X.509 standard means that these items can only be added via convention outside the standard. Naturally enough such non-standard conventions tend not to be globally consistent, compatible with other conventions or machine processable within the standard tooling for the format.

More generally, in the domain of public key infrastructure, the addition of semantics to signatures and automatic processing allows unification of various mechanisms of certification. It may be desirable to use hierarchical certification for some purposes, such as filing tax forms or contacting a bank, and, at the same time, use guerilla certification mechanisms for speaking with friends. By adding semantics to signatures, it is possible to use the same key for different purposes with different policies. A typical user account might have an X.509 certificate for signing email and accessing corporate virtual private networks, a key to log onto Lotus Notes, an ssh key for remotely accessing machines on a network, a PGP key for private correspondence, a code signing key for development, *et cetera*. One of the several reasons for this proliferation of keys, and thus an increase in the pain of key management, is that the only mechanism to separate the different uses of the keys is to use different keys. By providing semantic contextual information, we can *securely* use the same key in different contexts with separation provided by semantic differentiation. Use of such a system automatically stops chosen protocol

attacks [2] by ensuring that a signature used in one context cannot be applied in another.

Processes that are less common and well specified than certification suffer similar problems. On the positive side, because they are less common, they are less constrained by a deployed base. On the negative side, because they are less common, they are not as well studied or, generally, implemented.

In both categories, the switch to standard semantic formalisms increases the number of people who understand the format and the amount of documentation. For example, there are many more people who understand semantic web standards than the *ad hoc* semantics of X.509. There are also many more tools that support these standards, for editing, interpreting, validating and transforming semantic content (in a similar way that using XML as a data format allows developers to readily use standard parsers and other tools).

Additionally, using standard semantics in signatures allows native integration with other semantic content. In particular, one can referentially reuse existing knowledge representations (i.e. ontologies) toward specification of meaning in signatures.

The remaining paper is structured as follows. In Section 3 we further discuss elements of traditional digital signatures and point out their limitations with respect to semantic specification. In Section 4 we introduce basic building blocks and additional considerations necessary for achieving semantic signatures. In Section 5 we detail our semantic signature scheme. Section 6 outlines the realization of semantic certification using our constructs. Section 7 details an example in the compliance space, that further motivates our work in a concrete setting. Section 8 concludes the paper.

3. Traditional and semantic digital signatures

The core of digital signature schemes is asymmetric cryptography: digital signatures are defined by a well specified set of steps for producing and verifying signatures using public/private key pairs. For a signature scheme to be practical, it is also necessary to provide means of associating these mathematical objects to real-world entities such as users, and thus indirectly associating the produced signatures to these entities. Hence certification, and its associated processes, are an integral part of a digital signature scheme, as well as being an application of the digital signature scheme in the core sense. The certification application has both syntactic and semantic layers. The syntactic layer determines how to parse a signed message into the fields containing keys (or references to keys), information

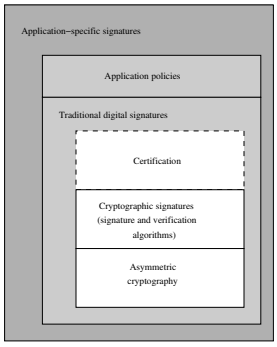

Figure 1: Traditional digital signatures in an application context

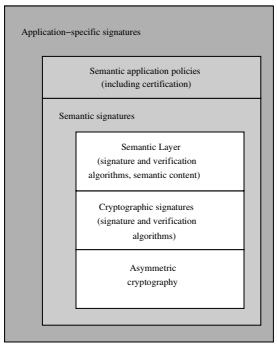

Figure 2: Semantic digital signatures

about the entities to whom the keys are associated, and whatever other metadata the particular standard supports. The most common semantic information is specification of a mechanism by which one party can attest to a binding between a key and information such as an email address or directory name.

This definition of digital signatures is depicted in Figure 1 (certification is shown as a special case, in that it is both a layer in the overall functionality, and an application of cryptographic signatures). We contrast it to Figure 2, which shows the insertion of a semantic layer above the traditional signature to produce semantic signatures; with this construction, certification becomes one of many possible policies that can be formally specified as part of the scheme and included in the signature.

Existing standards for certification include X.509 and PGP/GPG. In the following subsection, we examine X.509 in more detail. We have chosen to do so due to its widespread deployment and relative rich, if often under-specified, feature set. Conceptually, this interpretation should apply to other digital signature schemes. It should be noted that we do not expect that this widely deployed standard be replaced with a new

version based on formal semantics, but we believe that analysis of the problems of X.509 that stem from its *ad hoc* semantics is directly relevant to other, future, semantically enabled applications. As such, comments concerning X.509 should not be taken as concrete suggestions but are rather present to clarify general principles with examples from a widely considered and implemented standard.

3.1. Semantics of X.509

The X.509 standard has its lineage in the International Telecommunication Union, and as such, many of its design choices stem from the telecommunications context of the late 1980s (such as the assumption of a global X.500 directory and the strong focus on offline procedures). The PKIX workgroup of IETF is chartered to address Internet-specific aspects of X.509 usage. X.509-related standards and RFCs are vast and complex, probably due to this heavy historical baggage, and in general implementing an X.509 application is not a trivial matter (see [3] for a general discussion of X.509 inadequacies).

Certification in X.509 provides a binding between a public key and a subject expressed as a Distinguished Name, which is a hierarchical description of an entry in an X.500 directory (an extension mechanism, available since X.509v3, allows a certificate authority (CA) to bind the key to an Alternate Name such as an e-mail address or a URI). Validation of a certificate is hierarchical with a configurable collection of root authorities. With respect to semantic specification, the main problem of X.509 is that the precise meaning of this binding and the process performed by the certificate authority to attest to the binding are not within the scope of the base standard. For example,a binding between a key and the directory name "CN=Joe Doakes, DC=Hadleyburg, DC=Acme, DC=com" does not strictly imply that the owner of the key works for Acme Corp in the Hadleyburg office, or is named Joe Doakes. This is a practical manifestation of several deep problems concerning the association of names and the things to which the names refer [4].

An X.509 certificate has a well specified syntactic structure with standard fields such as issuer, subject and period of validity. In contrast to fields themselves, the values taken by the fields are significantly less well-specified. The issuer and subject fields, both of type Distinguished Name, can be composed of a wide variety of mostly optional-attributes (e.g. Country, Organizational Unit); assignment of values to these attributes varies greatly between certification authorities. For example, when issuing certificates to individuals for private use, public CAs typically set the subject's country to the CA's country (in general, CAs tend to wedge in the attributes information meaningful to them locally). There is no standard mechanism to allow clarification as to what these attributes mean, and it would be incorrect to assume any semantics from them (as their primary purpose is to provide a unique identifier for the certificate within a directory).

Profiling is another mechanism within the standard that can be viewed as serving a semantic purpose. Profiles of X.509 can be defined that place further restrictions on the contents of the certificate (e.g. what attributes are supported) and are intended for use within a specific domain. IETF PKIX defined the profile used in browsers, and others exist for banking, certain governmental uses, etc. However, the certificate itself does not indicate what profile it follows. This information is typically implicit in the application being used.

In addition to profiles, individual CAs define usage possibilities, which are also not encoded in the certificate itself, but are derived from the certification path. For example, Verisign defined certificate classes such as:

- Class 1 for individuals to send e-mail
- Class 2 for organizations
- Class 3 for servers and software signing

An extension mechanism introduced in X.509v3 allows vendors to place additional constraints on the certificate. An example constraint extension defined by IETF PKIX is *Key Usage*, which indicates the purpose of the key, with a value taken from a list of predefined purposes (for example: signature, encipherement, certificate signing).

These different X.509 mechanisms – attributes, profiles, certificate classes and extensions – provide *ad hoc* ways for introducing semantics in an X.509 application. The result is a certificate that is not self-contained, relying in poorly defined out of band mechanisms for semantic validation. Overall, X.509 is semantically underspecified and the underspecification is difficult to fix.

3.2. Semantic certification

The semantic aspects of an X.509 certificate and its associated processes can easily be expressed using Semantic Web technologies. Doing so we see that the advantages of semantic signatures discussed in Section 2 immediately apply to certification, for example:

- Vendors can clarify the manner in which they use terms in a standard, extensible, machine-processable fashion

- The terms exist in relationship to one another
- There are no problems with name space collision
- Developers have access to better tooling for the format

In Section 6, we elaborate on the realization of semantic certification (after having presented the details of the semantic signature scheme we propose).

4. Basic building blocks and requirements of semantic signatures

In this section we introduce the basic constructs and considerations necessary for building semantic digital signatures.

Further understanding of the paper requires basic understanding of the Semantic Web, and in particular of the Resource Description Framework (RDF), one of its core technologies. RDF provides a mechanism for making statements about resources, of the form (subject, predicate, object), with each of the elements of this triple being a URI. Other key ingredients of the Semantic Web are ontologies and reasoning. While RDF serves as a means for representing explicit knowledge, ontologies serve as a means for representing implicit knowledge, by listing a universe of terms and their relationships (hence knowledge is implicit with respect to a particular problem), and reasoning serves as a means for obtaining explicit knowledge from implicit knowledge. Because the paper is intended for a security conference, we have included as appendix a summary of the technologies from the Semantic Web that are relevant for our purposes. Note that, throughout the paper, we use the notation $\#property$ as a shorthand for properties defined in a formal ontology.

4.1. Binding of URIs

URIs [5] are at the core of the Semantic Web approach. Perhaps the most important subclass of URIs are Uniform Resource Locators (URLs), which provide names for retrievable content and are one of the pillars of the Web. URLs are fundamentally references; they indicate how and where to get content rather than actually *being* the content. Indeed, content behind a particular URL may change over time. This poses a problem when using digital signatures: at some point in the signing process a reference is included in the signed content, but typically it is the content itself which needs to be signed. For example, signing a statement saying "The content at http://en.wikipedia.org/wiki/Rsa is accurate" is problematic because the content served by the URL may change or be served differently depending upon different circumstances. Permalinks partially address the issue of evolving content. A permalink is a reference generated by an application, which uses it as a database key or as a query to retrieve a specific portion or version of content. Permalinks are application-specific: a permalink is meaningless if it is not dereferenced by the same application that generated it. Moreover, the binding between the reference and the content is not strong, so the application needs to be trusted to return the right content for a given permalink.

In order to bridge between the Semantic Web world and the cryptographic world, we require URIs that are strongly bound to the content they represent, in an application-independent way. We thus introduce the concept of BURI as a content Bound URI (respectively BURL). BURIs are based on incorporating a cryptographic digest of the content into the URI. We adopt XML namespaces and allow two types of bindings: a simple scheme and a dereferenced scheme. We note that the semantic signatures work presented in this paper is part of a larger effort [citation redacted for review] whose goal is enabling evaluable trustworthiness, hence trust, in web content, particularly HTML content. As such many of our concerns stem from this domain. This said, many of the concerns of HTML documents extend to other formats such as simple text, SVG, PDF, and GIF and JPEG images. Other formats that are proprietary, unduly complex or incompletely specified are by their nature difficult to sign: if one cannot unequivocally state that which is to be signed, it is not clear what a signature means even at the most basic level.

Before detailing the two binding schemes, we point out several practical concerns relevant to BURIs and HTML content.

Granularity is a first consideration; it is often useful to address a subsection of a document, as in citing an example or removing non-local content (such as advertising or other variable content). Additionally, enabling smaller scale granularity reduces the need for invalidating BURIs due to small changes in the content: if a change is non-local then the BURI will be valid, and a signature on it may still apply.

Alias URIs present a problem; it is common that several URIs are used to refer to essentially the same content[2] but produce different BURIs in a strict sense. We note that this is related to the problem of *Different URL Similar Text* (DUST) [6], which has been studied in the context of crawlers and annotation services.

Normalization, a necessary step prior to digest

2. Varying with banner advertisements or page counters.

calculation, is by far the most challenging problem. For example, HTML normalization may be performed either on the byte sequence returned by the server or on the already parsed DOM representation of that byte sequence. Referenced or lazily included content, such as images or third-party text, are problematic: an HTML document referencing an external image will produce the same digest even if the image changes (which is natural, but potentially contrary to the intent of a signature on the BURI). This problem may be mitigated recursively, by normalizing the content to include digests of included content, but if this is done unwisely, one risks taking a checksum of the entire Web.

Languages embedded in content, such as Javascript and XSLT, are problematic in several ways (closely related to the normalization problem). The first is that the languages may be used maliciously to alter content based upon retrieving conditions. Even if the language is available for inspection or directly included in the content, rather than by reference, determination of the code effect may be difficult or nearly impossible [7]. Potential attackers can also manipulate cache parameters and metadata of channels, such as HTTP, to generate the illusion of benign code that is actually malicious. For digital signatures to have any significance on HTML content, the signing scheme should have a *what you see is what you sign* (WYSIWYS) property; in other words, there shouldn't exist a semantic gap between the sequence of bits upon which the digital signature is computed, and the high level representation of this sequence that the user intends to sign. The aforementioned attacks compromise this property. A full discussion of trusted displays and trusted computing bases [8] is well outside the scope of this paper, but it should be noted that trustworthy display of content is extremely difficult. Even with a format as simple as text, different fonts might lead to a byte sequence being displayed "I will pay 1000€" in one place and "I will pay 1000£" in another. However, for simple cases, the problem may be addressed via normalization - by e.g. discarding all formatting and script tags. Embedded languages are further problematic as significant content is now delivered via asynchronous channels, such as is the practice with AJAX (Asynchronous JavaScript and XML) so that the initial content may only be a collection of pointers to the real content. Several widespread web content generation frameworks, such as DOJO, work in this fashion.

4.1.1. Simple Binding. In the simple form, we add a cryptographic digest as the parameter to a URI:

```
http://www.w3.org/TR/uri-clarification?
{nsp}:sha256=e3b...855&xmlns:{nsp}=...
```

where the string `{nsp}` is an arbitrary name space prefix, in the sense of XML, specifying the namespace `http://redacted.org/2008/01/aha/`; by default we use `aha` (which stands for *ad hoc anchoring*, a concept outside of the scope of this paper). While XML namespaces are not part of the URI specification, we want to ensure that any URI can be transformed into a BURI and hence need a layer of indirection.

The preceding BURL example indicates that the content served by the URL

`http://www.w3.org/TR/uri-clarification/`

has the sha256 [9] checksum `e3b...855`.

Note that sha256 may be replaced by other cryptographic checksums. In particular, we wish to allow future algorithms, such as that which will be the winner of the upcoming NIST competition[10]. We presume that there will be an unambiguous, case insensitive short name for this digest.

We note that this scheme is limited, as it does not address fine granularity or normalization.

As a side benefit, using simple bindings eliminates many caching and distribution problems; if a system is provided the cryptographic checksum of some content and it happens to have a document matching that checksum, there is no need to retrieve the document.

4.1.2. Dereferenced Binding.. In order to enable signatures on more general web content, we build a dereferencing scheme atop the above trivial scheme. With this mechanism, a number of aspects of the referenced content (including its digest) are described in a content binding document; this document, which must be retrievable, is assigned a BURL that follows the simple scheme described above. Validating a dereferenced BURL requires two passes: validating the BURL (which requires retrieving the content binding document) and validating the content binding document (which in turn requires retrieving and validating the actual referenced content).

The content binding document has a number of components, some of which are optional:

- **permalink** – a permanent link for the referenced content (or a specific version of the referenced content), if one exists
- **logical link** – the standard link of the (mutable) content (note that a logical link may be resolved to a permalink, such as in Wikipedia)
- **fragment selection method** – the method used to subselect portions of content (XPath, XPointer,...)
- **fragment identifier** – a subselection of the content that is being referred to
- **content digest method** – the identifier of a digest method(sha1, sha256,...)

- **content digest** – the digest of the content retrieved from the permalink plus the fragment identifier (i.e. the digest of a portion of the document)
- **normalization method** – the identifier of a method of normalizing the content prior to computing the digest
- **serialization** – the identifier of how the content should be serialized to a byte sequence.

For example, the following BURL

```
https://redacted.org/burls?
{nsp}:sha256=e3b...855&xmlns:{nsp}=...
```

serves a content binding document whose digest is `e3b...855` (calculated using the digest method indicated in the BURL). This document contains the following information[3]:

```
<burl>
 <permalink>
  http://en.wikipedia.org/w/index.php?
  title=Currying&amp;oldid=118180680
 </permalink>
 <logical-link>http://en.wikipedia.org/wiki/Currying</logical-link>
 <fragment-selection-method>html</fragment-selection-method>
 <fragment-identifier>Scheme</fragment-identifier>
 <content-digest-method>md5</content-digest-method>
 <content-digest>68b...c940</content-digest>
 <normalization-method>
  https:///normalization/simplehtml_1.0
 </normalization-method>
</burl>
```

An example HTML normalization method may discard tags such as formatting information and scripts, and replace images with simple BURLS, as in `<img src="lol.jpg" .../>` with

```
<img src="lol.jpg" aha:sha256="e3b...855"
xmlns:aha="http://veracite.../aha/" .../>
```

4.2. Semantic Injection Attacks

RDF is a language for encoding information assembled from (subject, predicate, object) statements. It is most often used as a data format. The truth, accuracy and context of the statements are not in the scope of the language. The consequence is that the most straightforward encodings of information[4], fail when applied to complex semantics. The idea is best illustrated by an example: the sentence "Bob does not know that Alice signed the message M" does not imply that "Alice signed the message M" yet the natural RDF decomposition of the first sentence has a subgraph that is identical to the RDF decomposition of the sentence "Alice signed the message M".

Any application that

- has complex semantics or,
- has simple semantics but need to process general semantic input or,
- processes semantic input from potentially untrustworthy sources

is thus potentially vulnerable to attacks in which decisions are made, without contextual consideration, using semantic data whose context changes its meaning. We will call this a semantic-injection vulnerability, drawing attention to the fact that the problem is in the semantic layer. This distinguishes it from a semantic query injection attack, or indeed any semantic/non-semantic query/selection language attack, in which the problem exists in the mixing of control and data channels[11].

We note further that semantic injection concerns apply not only to the semantic data but to the schemas and ontologies that apply to the data. For example, a schema that falsely asserts that $\#seen$ is a subproperty of $\#signed$, would potentially cause a reasoner to conclude incorrectly that a document that was merely seen by a person, was also signed by that person.

Insofar as we are engineering a security mechanism in this paper, we do not want to place the burden of addressing this danger upon the designers of the ontologies that are used in the signatures (indeed, an important security maxim is to place as much of the security burden in the lower layers and do not assume that crucial mechanisms are going to be implemented at a high level). We thus need to address the problem at a lower level, but we yet wish to retain the RDF formalism and tooling.

Concretely, this means that in addition to the natural RDF interpretation of $sign_K(M)$ is the statement

$$K, \#signed, M$$

together with some (secure) mechanism of identifying this statement (and distinguishing it from other statements).

Addressing this problem via a security statement manager, that is, by disallowing statements introduced via certain channels, does not seem to be a viable solution: beyond limiting the semantic expressivity in an absolute sense, it would create immediate concrete problems by eliminating many extremely useful constructions, such as those needed by certificate authorities. It would further not protect against attacks in ontologies other than signing. For example, "Bob does not know whether the process is compliant" which has "the process is compliant" as a subgraph.

An approach of stating that a statement is true, does not work due to the fact that all of the statements are *in band*. An attacker knowing of this mechanism could create a message of the form "Bob does not know that it is true that Alice signed the message M".

We can, however, address this problem using reification, an RDF mechanism by which statements can refer to other statements [12], together with a *secret* URL

3. We present this content binding in XML for illustration purposes; XML is one possible data format, with many drawbacks as is discussed later.

4. Indeed exemplary in texts and tutorials.

V. Then $sign_K(M)$ may thus be securely represented as

$$V, \#attests, (K, \#signed, M)$$

An attacker cannot maliciously introduce this statement into our store via "Bob does not know that V attests that Alice signed the message M" because the V is secret.

To generalize this scheme, we can generate a family of secrets V_n by appending a counter to V. Now for a given semantic message M we can import $sign_K(M)$ as

$$V_n, \#attests, S$$

for all base statements, that is those which appear directly rather than via reification, $S \in M$ and

$$V_n, \#concerns, RS$$

for all other $S \in M$, with the schema constraint that $\#attests$ is a subproperty of $\#concerns$.

This secure indirection layer allows meaningful expression and distinction of reified statements, thereby avoiding semantic injection attacks while permitting similarly structured valid semantic constructions such as key certification.

4.3. Data Formats

There are two distinct problems related to data formats.

The first is that signing a directed graph, roughly the underlying data model of RDF, is potentially quite difficult [13] due to the difficulty of graph equivalence [14]. This difficulty would manifest itself if the verifier were to start with a collection of RDF statements (a graph), and need to know if a signature applies (thus requiring a graph equivalence computation). For our purposes, the verifier instead starts with a particular serialization of a graph, verifies the core digital signature, and then transforms it into the reified model "signer signed graph".

The second, more pedestrian, difficulty is the end representation. The most natural choice to format a combination of RDF and the binding information is XML. Unfortunately, XML is not a particularly easy format to sign for a number of reasons. The most problematic of these is that support for mixed-content implies the need for a schema when parsing; even with a schema, the map between the DOM model of a document and its serialization may not be a strict bijection due to white space.

XML Signature [15] is a W3C recommendation that defines both an XML syntax for specifying parameters

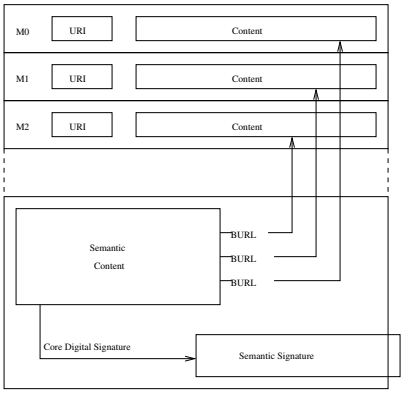

Figure 3: Compound message

for digital signatures (asymmetric algorithm, cryptographic digest, *et cetera*) as well as a means of XML canonicalization using XML Schema. The latter aspect makes it a viable choice for canonically serializing that data which will comprise a semantic digital signature. Another possibility could be to replace the standard XML serialization with a more data friendly representation such as a canonicalized list of SAX [16] calls.

5. Semantic digital signatures

Semantic signatures are defined by signing and verification algorithms that make use of BURIs and address the concerns stated in Sections 4.2 and 4.3. For producing a semantic signature, this semantic layer creates a message that follows a prescribed data schema, and is the input to the underlying core signature mechanism.

Generally, with digital signatures, there is the question of whether the content signed is bundled with the signature. For our purposes, we have addressed this by using the BURI construction together with a container format. The resulting compound message has the structure depicted in Figure 3, in which the content is optional. Each serialized base message is labelled with a URI which will be referred to in the semantic content via bound reference. In contrast to traditional signatures, we have opted to allow multiple content blocks for two reasons. Firstly, we can envision settings in which semantic processing requires access to different data sources within the content for semantic processing. Secondly, nearly all container formats, such as S/MIME, allow multiple blocks. Insofar as we wish to use standards it does not make sense to limit these standards. We mention explicitly that the content required for indirected BURIs is assumed to come from content blocks before being loaded over the network.

The semantic content has the following structure:

- semantic armor
  - set A of assertions
  - set of ontologies O_A, from which the assertions take their terms
  - a list of content BURIs, indirectly content, to which the statements apply
  - identification of suggested W of world beliefs
  - identification of suggested reasoner r
- semantic armor

wherein the semantic armor is merely a standard statement that the content of the signature should be interpreted as a semantic signature and not otherwise. We have placed it at both the beginning and end to prevent attacks that somehow might be based on parse order.

5.1. Assertions

The assertions are a collection of RDF statements encoding the meaning of the signature. They are bound to the content to which the signature applies using the BURI mechanism.

5.2. Ontologies

As with a standard Semantic Web application, ontologies determine the universe of terms used in the statements and their relationships. In the case of semantic signatures, the ontologies must be known to the verifier to prevent semantic injection. Note that ontologies (and their specific versions) would typically be included via BURI.

5.3. List of content BURIs

We explicitly include a list of BURIs for the content to which the signature applies, rather than implicitly deriving it from the list of assertions, as a security mechanism to ensure that the signers' intentions apply only to the desired content (rather than to incidentally referenced content).

5.4. World Beliefs

World Beliefs are statements that are believed by the verifier. These beliefs may be used in the verification of signatures, determination of compliance or other automated processes. One can think of these as a generalization of the pool of certificate authorities that a traditional signature scheme might have. Example world beliefs are:

- a particular key, identified by BURI, may be used to certify other keys (potentially with well-defined specification of key capabilities),
- a particular key, identified by BURI, is associated to a particular Distinguished Name,
- a particular condition needed for compliance has been satisfied,
- a particular individual, identified by Distinguished Name, is authorized to make the assertion that another individual, identified by Distinguished Name, works for the Acme Corporation in the Hadleyburg office.

These are, in short, the initial assumptions that are needed to seed belief.

While the signing party cannot set the verifier's world beliefs, the field W is a suggested set the signer suggests is needed to make the signature meaningful. This set can also be used to improve the speed with which a statement is verified, by limiting semantic queries in cases where the suggested world beliefs are a subset of actual world beliefs. An example is a BURI identifying a certificate authority's root key included in a certificate produced by that authority. It is naturally important that the verifier does not simply accept the suggestions of the signer.

5.5. Reasoner

We have included a field for the sort of reasoner used in the semantic signature. As with world beliefs, this field is used as a suggestion from signer to the validator about what validation is appropriate or most effective.

Beyond this there are a number of fundamental questions about the nature of signatures and semantics:

- If I sign "A implies B and A" have I implicitly signed B?
- If I sign "A implies B and C" and C obviously implies A, have I implicitly signed B?
- If I sign a statement S and a reasoner, am I implicitly signing all implications of the reasoner applied to S?
- If I sign a statement S and a reasoner, am I implicitly singing all implications of any correct reasoner applied to S?

Which are *not* intended to be addressed with this field.

5.6. Signing and verification

5.6.1. Signing. Semantic signing prepares and combines the data elements described in the previous

section. The signer:

1) Collects, cryptographically hashes and names a content block or collection of content blocks, as to generate a collection of BURIs (they are BURIs rather than BURLs insofar as there is no guarantee that content will be served by the specified URI),
2) generates semantic statements concerning the meaning of the signature. These statements are cryptographically bound to the content by use of the generated BURIs,
3) assembles the semantic statements together with (bound) references to the ontologies in which the statements take their meaning, and suggested reasoner and world beliefs, placing them inside the semantic armor,
4) passes this assembly to a traditional digital signature scheme.

The signature thus obtained is adjoined to the named base content to form a compound message. If it is the signer's intention that the content be delivered via alternate channels, such as downloading a BURL, the content can simply be omitted.

5.6.2. Verification. Traditional public key verification results in a binary result `(valid, invalid)` indicating that a particular binding exists between a key and content. Conceptually, semantic signatures result in either `invalid` or a collection of reified statements representing "X said Statement1, Statement2,..." that concern and are cryptographically bound to a collection of content blocks.

As with traditional signatures, the verification process resembles the signing process. The verifier:

1) Parses the compound message, cryptographically hashes each base content message, and checks to ensure that they match the BURIs. Failure to match results in `invalid`,
2) verifies the signature on the semantic content, using the underlying signature validation procedure. Failure to validate results in `invalid`,
3) checks that the formatting of the semantic content complies to the syntax of semantic signature specification. Failure to comply to the format results in `invalid`,
4) ensures that the specified ontologies are known and acceptable. An unknown ontology results in `invalid`,
5) the following are returned to the semantic verifier:
   a) semantic content parsed into a collection of statements together with the generated security support statements for the semantic content statements (Section 4.2)
   b) the suggested world beliefs as a collection of statements (these have no security support statements),
   c) an identifier of the suggested reasoner.

6. Realization of semantic certification

In this section we examine the realization of certification with the concepts and constructs we have introduced. A full ontology of X.509 is outside the scope of this paper. We thus present an outline of the semantic expressions of certification and explain how it can be expanded. We define the following data as expressed by BURIs:

- $\#certkey$ is the key of a trusted certificate authority
- $\#dname$ is a Distinguished Name
- $\#key$ is a key associated to the Distinguished Name
- A standard X.509 ontology defining properties $\#certifies$, $\#associated$, $\#key\_for$, $\#all\_associations$, *et cetera*

The most basic semantic certification has

- Assertions:
  - $S_0 := (\#dname\, \#associated\, \#key)$
  - $\#certkey\, \#certifies\, (S_0)$,
- Ontology: bound reference to a standard X.509 ontology
- Reasoner: an X.509 reasoner
- List of BURIs: $\#dname$, $\#key$
- World beliefs: $(\#certkey\, \#key\_for\, \#all\_associations)$

which, through validation, gets transformed into the statements:

- $S_1 := (\#dname\, \#associated\, \#key))$
- $V_n\, \#attests\, (\#certkey\, \#certifies\, S_1)$
- $V_n\, \#concerns\, (\#dname\, \#associated\, \#key)$

which the reasoner transforms into:

- $V_m\, \#attests\, (\#dname\, \#associated\, \#key)$

This illustrates a basic certification scenario. The advantages of using semantic signatures over an *ad hoc* semantic specification are immediate. In particular,

- The terms in the system are meaningful and carry relations to one another due to their definition in a formal ontology.
- The system is extensible. For example, the email certificate class can be represented as the assertion
  - $S_2 := (\#dname\, \#associated\, \#key)$
  - $S_3 := (S_2\, \#key\_for\, \#email)$

  - $\#certkey\,\#certifies\,S_3$
- We can similarly, standardly, and meaningfully express such qualifiers as the conditions under which the CA made the certification, new classes and their relation to old classes, arbitrary restrictions (profiles), usage policies, *et cetera*.

7. Compliance Scenario

To illustrate further usage of semantic signatures, we discuss a concrete example of compliance checking from the pharmaceutical industry. The United States Food and Drug Administration (FDA) guidelines dictate that a certain collection of tests must be done on a drug before it is vetted for human testing. When requesting approval for a new drug, pharmaceutical companies have to demonstrate they have followed such guidelines, generally by submitting documentation of their processes in paper form. The FDA accepts digital records and signatures in place of their physical counterparts if the conditions specified by Title 21 Part 11 of the Code of Federal Regulations [17] are met by the company's systems and processes[5]. Among these conditions, the regulation states that all signed electronic records must contain information identifying the signer, date and time of signature execution and meaning of the signature (examples such as review, approval, responsibility or authorship are cited in the regulation, but this depends on the particular approval being sought). Note that the regulation does not mandate any particular format for this information. While 21 CFR Part 11 enables the use of digital signatures when submitting paperwork for approval by the FDA, the lack of formal semantics within digital signatures means that determination of compliance is a human matter.

Our scenario is based on the following simplified policy, abstracted from FDA guidelines: in order to ensure that the tests are done correctly, they must be done by two different parties, a primary tester and a verifying tester, each of who is authorized to perform the tests.

Formalizing this, we need four *different* parties:

- A**:** Primary tester,
- B**:** Verification tester,
- C**:** Testing Certification Authority,
- D**:** Key Certification Authority.

We assume that the roles of A and B are not symmetric as to allow richer semantics; the actual details are not important for our purposes.

5. 21 CFR Part 11 is contested at many levels in the industry, as the specification is too wide and open for interpretation; a new regulation is underway.

We assume that each party x has a public/private key pair $K(x)$ that is used in compliance with 21 CFR Part 11; for our purposes, the use of the public or private components of $K(\cdot)$ is contextually clear. We translate compliance into a collection of signatures, assertions, and rules as described in the following subsection.

7.1. Rules

- There is some drug Ψ,
- Evidence Σ demonstrates that drug Ψ satisfies test T,
- A, B, C, and D are distinct entities,
- $K(A)$ signs "I produced evidence Σ that showsΨ passes T",
- $K(B)$ signs "I verified evidence Σ that showsΨ passes T",
- $K(C)$ signs "A is certified to run the primary test",
- $K(C)$ signs "B is certified to run the verification test",
- for $x \in \{A, B, C\}$, $K(D)$ signs "the key $K(x)$ is associated to the directory name x"

Note that the roles of C, the testing certification authority, and D, the key certification authority, are a matter of policy configuration (in other words, a world belief).

7.2. Realization of scenario

Translating these rules into a semantic signature scenario begins with the creation of an ontology for the drug testing guideline being satisfied, and using this ontology in union with a certification ontology. An RDF graph representing the scenario in terms of semantic signatures is shown in Figure 4. We have that A semantically signs:

- Assertions:
  - $S_4 := (\#\Sigma\ \#shows\ (\#\Psi\ \#satisfies\ T))$
  - $\#A\,\#primary\ S_4$
- Ontology: bound reference to standard certification and drug testing ontologies
- Reasoner: a certification reasoner
- List of BURIs: $\#\psi$, $\#T$, $\#\Sigma$
- World beliefs: $(\#certkey\,\#key\_for\,\#all\_associations)$

while B semantically signs a similar message with the sole difference in assertions:

- Assertions:
  - $S_4 := (\#\Sigma\ \#verifies\ (\#\Psi\ \#satisfies\ T))$
  - $\#A\,\#verifies\ S_4$

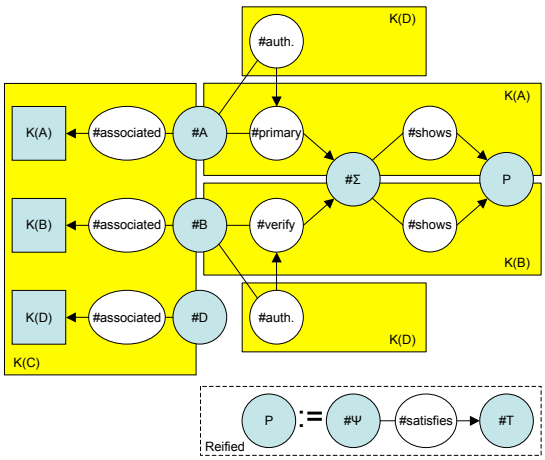

Figure 4: Representation of scenario as an RDF graph with signed statements.

C semantically signs two messages

- Assertions:
  - $\#A\,\#authorized\,\#primary$
  - $\#B\,\#authorized\,\#verifies$

Validation of the signatures results in a collection of statements. Querying of these statements, bearing in mind that statements must be reified against semantic injection, directly translates back to the rules of compliance [Section 7.1] thus providing demonstrable proof of compliance.

7.3. Observations

In the area of compliance checking it is important to audit signed artifacts and check who was involved in which step (and signed what). It is important to know in which capacity each person was involved in a given process. Often this information needs to be encoded explicitly inside the signature in such a way that it can be taken into account in an expanded verification process. There are obvious *ad hoc* ways for achieving this in a particular situation, but we are interested in approaching the problem in a more general sense, accommodating the fact that it would be unrealistic to assume that up-front agreement on an immutable set of capacities could be achieved and that those would be used by everybody (e.g. anywhere on the Internet) without ambiguity and overlap.

Two further observations can be made related to concrete applications. Firstly, that different parties carry different obligations and they can have quite important real-life consequences e.g. in a legal sense, but they are often only implicitly conveyed. Secondly, that in the physical world, signatures are relatively rare: nobody expects the copy editor to sign off on physical documents, and signatures are only required if certain minimum obligations, that could become relevant in legal disputes, apply. With digital signatures, the whole signing process is automated and can be performed on behalf of a user without any noticeable effort; a typical IT user has information signed on many occasions during a normal day of use. For instance the simple step of accessing a corporate network over a Virtual Private Network entails digital signatures, and so of course does online shopping. The parallels between the physical world and the real world end when considering this explosion of signed instruments, and further automation becomes a necessity.

8. Concluding remarks

We have presented a high-level view of a mechanism for enriching digital signatures with formal semantics. In doing so, we have presented a number of constructions that are of use in general applications using semantic technologies. We have further introduced a new class of attack against applications using RDF, the basis of the Semantic Web. Full specification of semantic digital signatures requires significant further syntactic and programming interface development. This engineering effort is underway as part of a general effort to merge security mechanisms with semantics toward automated compliance checking.

References

[1] Adobe, "PDF Security http://www.adobe.com/products/pdfs/AdobePDFSecurityGuide-c.pdf," 2005.

[2] J. Kelsey, B. Schneier, and D. Wagner, "Protocol interactions and the chosen protocol attack," in *In Proc. 1997 Security Protocols Workshop*, pp. 91–104, Springer-Verlag, 1997.

[3] P. Gutman, "Pki: It's not dead, just resting," *Computer*, vol. 35, no. 8, pp. 41–49, 2002.

[4] S. A. Kripke, *Naming and Necessity*. Harvard University Press, 1980.

[5] W. URI Planning Interest Group, "Uris, urls, and urns: Clarifications and recommendations 1.0," 2001.

[6] Z. Bar-yossef, I. Keidar, and U. Schonfeld, "Do not crawl in the DUST: different URLs with similar text," in *In Proc. 15th WWW*, pp. 1015–1016, ACM Press, 2006.

[7] J. Riordan and B. Schneier, "Environmental Key Generation Towards Clueless Agents," *LECTURE NOTES IN COMPUTER SCIENCE*, pp. 15–24, 1998.

[8] R. C. Summers, *Secure computing: threats and safeguards*. Hightstown, NJ, USA: McGraw-Hill, Inc., 1997.

[9] National Institute of Standards and Technology, "Announcing the SECURE HASH STANDARD," 2002.

[10] National Institute of Standards and Technology, "Announcing the Development of New Hash Algorithm(s) for the Revision of Federal Information Processing Standard (FIPS) 180-2, Secure Hash Standard," *The Federal Register*, vol. 72, no. 14, pp. 2861–2863, 2007.

[11] T. Pietraszek, C. V, and E. Berghe, "Defending against injection attacks through context-sensitive string evaluation," in *In Recent Advances in Intrusion Detection (RAID*, 2005.

[12] F. Manola, E. Miller, *et al.*, "RDF Primer," *W3C Recommendation*, vol. 10, 2004.

[13] Jeremy J. Carroll, "Signing RDF Graphs," 2003.

[14] Béla Bollobás, *Modern Graph Theory*. Springer, 2002.

[15] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon, "Xml signature syntax and processing (second edition)," 2008.

[16] "Simple api for xml http://www.saxproject.org/," 2008.

[17] "Title 21 Code of Federal Regulations (21 CFR Part 11) Electronic Records; Electronic Signatures," 2000.

[18] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, May 21 2001.

[19] D. Beckett, "Expressing simple dublin core in rdf/xml," 2002.

[20] F. Baader and W. Nutt, *The Description Logic Handbook*, ch. Basic Description Logics. Cambridge University Press, 2003.

Appendix

Semantic Web Summary

The Semantic Web [18] is an extension of the existing World Wide Web, which allows adding representations of formal knowledge in a format that facilitates automatic reasoning and other meaningful manipulations of web content by automatic software.

Early proponents argued that human-readable information that currently forms the bulk of web content is only suitable for some low-level processing tasks, such as formatting and display, because the World Wide Web was first designed with only human readers in mind. By this argument more advanced applications strictly required a major upgrade of the web's core infrastructure of the kind envisioned for the Semantic Web.

The considerable and ever-growing volume of human-readable (and, from the point of view of high-level processing, initially unstructured) web content has meanwhile allowed statistical methods, such as those used in search engines, gain considerable traction. These for the most part do not require new data formats and infrastructure, and hence attractive applications that specifically depend on the World Wide Web as a whole evolving into the Semantic Web have become less apparent. This may be a major reason why wide-scale adoption on a global footing is not apparently imminent at this time.

However, the ingredient technologies are at a mature level and constitute open standards; for the time being they can also be considered for use in single applications that require knowledge representation, in particular when this knowledge concerns web-based resources. This appendix briefly discusses three core ingredients, namely RDF, OWL (a loose acronym for Web Ontology Language) , and reasoning. RDF serves as a means for representing explicit knowledge, OWL serves as a means for representing implicit knowledge (implicit with respect to a particular problem), and reasoning serves as a means for obtaining explicit knowledge from implicit knowledge.

Resource Description Framework

The Resource Description Framework (RDF) provides a mechanism for representing knowledge about web-based and other resources. Resources are identified by Uniform Resource Identifiers (URIs, [5]), and can therefore comprise all information on the web (with URIs indicating network locations) as well as all other entities that can be assigned such unique identifiers (e.g. encoding geographic locations inside URIs is straightforward).

RDF is not concerned with what is stored *behind* URIs (e.g. with the content of corresponding web pages), but only with information *about* resources. This metadata consists of a set of statements (mini-sentences) of a single form, each comprising a subject,

predicate, and object, in the form of a URI each (blank subjects and literal object values are out of scope of this brief overview.) At a conceptual level, RDF is attractively simple, because it solely relies on statements in the uniform form of 3-tuples. Some of this simplicity becomes less apparent when the RDF graph is expressed in a concrete serial syntax, such as XML in the case of RDF/XML serialization.

For making a statement about a resource, one can formulate a triple that has the respective resource URI as its subject. The semantics of what is represented in this way depends on what predicate and object are used. These resources are also represented by URIs, which in turn and recursively can be described by further statements. The union of all related statements forms a directed graph, whose nodes correspond to all subjects and objects and whose vertex labels correspond to all predicates.

The transitive closure of all edges leading outwards from a node corresponds to the entire meaning of the entity named by the resource URI, as far as RDF knowledge representation is concerned.

Ontologies

The knowledge representation mechanism adopted for the Semantic Web accounts for the distributed nature of the world wide web. In particular, since content creation occurs potentially in parallel at many places, times, and jurisdictions, it is in practice unavoidable that equal concepts will be referred to under different URIs by different authors (e.g. who may form semantically equivalent names using different languages and arriving at different URIs), and described in terms of different relationships.

Adopting URIs for naming resources has therefore two direct advantages. First, there are existing practices for managing the entire name space of possible URIs in non-overlapping administrative domains. (Domain Names partly constitute URIs.) Second, there is a direct mapping of URIs that represent web-based resources to a suitable transfer protocol (HTTP Hypertext Transfer Protocol).

OWL is a type language for expressing rules for reasoning about RDF data. It encodes implicit assumptions that underlie the description of certain resources e.g. in terms of their expected relationships and known differences. Concepts are resources that characterize common sets of individual resources according to those properties. For the purpose of convenience and reuse, concept definitions are grouped into ontologies. (Unlike with object-oriented type systems, concept membership of an individual resource is just one property among others, and is dynamically inferred as well.) Some ontologies of specialized concepts, such as for describing publishing information [19] already exist.

OWL offers three dialects with increasing expressiveness, and correspondingly increasing difficulty of reasoning and hence computational cost. Expressiveness comes from a range of available concept-forming primitives that allow the description of new concepts in relation to existing ones according to Description Logic (DL). For example, there are primitives for expressing disjointness between concepts, or cardinality of individuals in certain relations to a concept. OWL Lite allows a subset of concept-forming primitives from DL and is suitable for building simple hierarchical taxonomies. OWL DL allows all DL concept-forming primitives and is still computationally complete and decidable. OWL Full further reduces the constraints on the use of these primitives, but it implies serious computational obstacles under state-of-the-art theory and technology.

Reasoners

The underlying theoretic formalism of the Semantic Web is Description Logic, which emerged from predecessor systems such as frame-based systems by adding more precise formal logic. In terms of expressivity and decidability Description Logic corresponds to a subset of First Order Logic [20].

By giving unique names to concepts (RDF) and by expressing their differences in exact terms (OWL) logical reasoning can occur. The basic supported reasoning task checks whether one concept subsumes another, i.e. whether all individuals belonging to one concept implicitly also belong to another. (Other supported reasoning tasks, such as satisfiability, can be viewed as specializations thereof.)

Conceptually, the inputs to reasoning are RDF statements (vertices in a graph) and a set of ontologies (OWL), and the output are additional and qualified vertices. Unlike the initial ones, which can be considered unqualified because of the axiomatic nature of the initial RDF statements, those added by reasoning are qualified by what implicit knowledge was considered by the reasoner.