WWW 2004, New York
Developers Day, 22 May 2004

The Sematic Web Trust Layer

Jeremy Carroll, Hewlett-Packard Labs, UK
Chris Bizer, Freie Universität Berlin, Germany

Joint work with
Pat Hayes, IHMC, USA
Patrick Stickler, Nokia, Finland

Central Questions for the Semantic Web

- How trustworthy is information found on the Semantic Web?
- How do I decide that it is trustworthy?

Agenda

1. Introduction
   - Basic Roles
   - Trust Mechanisms
   - Requirements for a Semantic Web Trust Layer
2. Publishing Information on the Semantic Web
   - Named Graphs
   - Semantic Web Publishing Vocabulary
3. Trusting Information found on the Semantic Web
   - Example Trust Policies

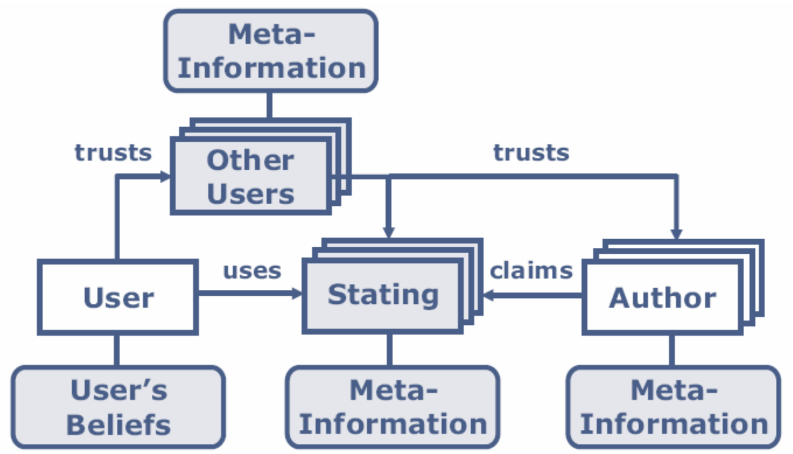
Basic Roles

- Information Providers
  - want that their information is used / believed
  - might want to state their publishing intend (assertion, quote)
  - are only willing to put a certain effort into publishing
- Information Consumers
  - want to use the information for different tasks
  - have different views of the world
  - have different subjective trust requirements
  - have different subjective preferences for certain trust mechanisms

Trust Policies

- We are using a wide range of different trust policies in everyday life:
  - We might trust Andy on restaurants but not on computers,
  - buy only from sellers on eBay who have more than 100 positive ratings,
  - regard literature as irrelevant, when it is older than 5 years,
  - trust professors on their research field,
  - believe foreign news only when they are reported by several independent sources.
- Goal: Allow a similar wide range of trust policies on the Semantic Web.

Trust Situation on the Semantic Web



Trust Mechanisms

- Reputation-based Trust Mechanisms
- Context-based Trust Mechanisms
- Content-based Trust Mechanisms

Reputation-Based Trust Mechanisms

- include rating systems and web-of-trust mechanisms
- are a well researched area
- have a general problem:
  - They require explicit and topic-specific trust ratings
  - high effort for information consumers

Context-Based Trust Mechanisms

- use background information about the information provider
  - e.g. his role in the application domain or his membership in a specific group
  - example policies: "Distrust everything a vendor says about his competitor“ or “Trust all members of organization A.”
- Information created in the information gathering process
  - publishing and retrieval date and the retrieval URL
  - information whether a signature is verifiable or not
  - example policy: “Trust all information which has been signed and is not older than a month.”

Content-Based Trust Mechanisms

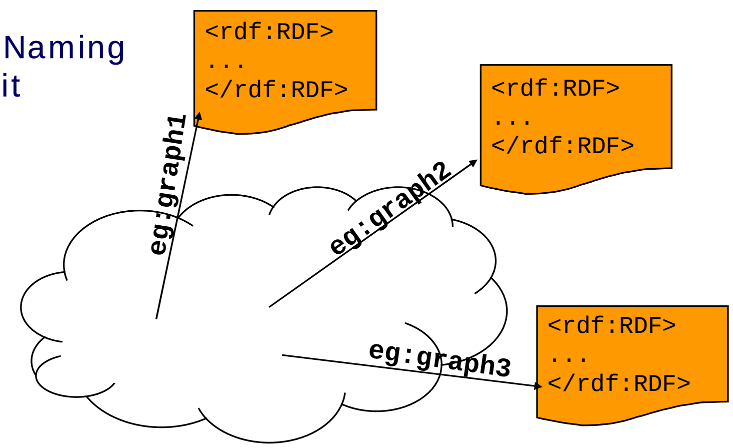
- use information content itself, together related information content published by other information providers.
- Example policies:
  - “Believe information which has been stated by at least 5 independent sources.”
  - “Distrust product prices that are more than 50% below the average price.”
  - “Distrust people claiming that Texan cows are aliens.”

Requirements for a Semantic Web Trust Layer

- Support different types of warranties
- Use of all trust relevant information available:
  - Journalism's WWWWW: *who*, *what*, *where*, *when* and *why*
- Support different, subjective, task-specific trust policies
  - Reputation-based
  - Context-based
  - Content-based
- Keep in mind that many applications don't require total trustworthiness.

Named Graphs - Motivation

Make Naming Explicit



Named Graphs – Abstract Syntax

- **N** = (N, V, U, B, L)
  - V = U ∪ B ∪ L, URIs, blank nodes and literals
  - N is a partial function
    - domain U
    - range RDF graphs (sets of triples from V x U x V)
    - alternatively a set of pairs – each pair being a named graph
- For a named graph ng = (n, g)
  - name(ng) = n
  - rdfgraph(ng) = g
- Blank nodes **not** shared between different graphs in N

Named Graphs – Semantics

- For every ng = (n, g) in **N**
  - **I(n) = ng** in an RDF interpretation **I** which conforms with a collection of named graphs **N**
    - **Semantic Extension in terms of RDF Semantics**
- **class rdfg:Graph**
- **properties rdfg:equivalentGraph rdfg:subGraphOf**
  - **technical detail considering blank node names**

Named Graphs – Semantics 2

- A collection of named graphs **N** is not given **a single** semantics
- Semantics determined by set A of uris of accepted graphs.
- Semantics of **N** with respect to A is RDF semantics of

$$\bigcup\{\ \mathrm{rdfgraph}(N(a)) : a \in A\}$$

- Thus $2^{|N|}$ different meanings
- Trust is the problem of determining A

Named Graphs - Syntax

- RDF/XML
- TriX (HPL-2004-56, with Patrick Stickler)
- TriG (based on Turtle subset of N3)

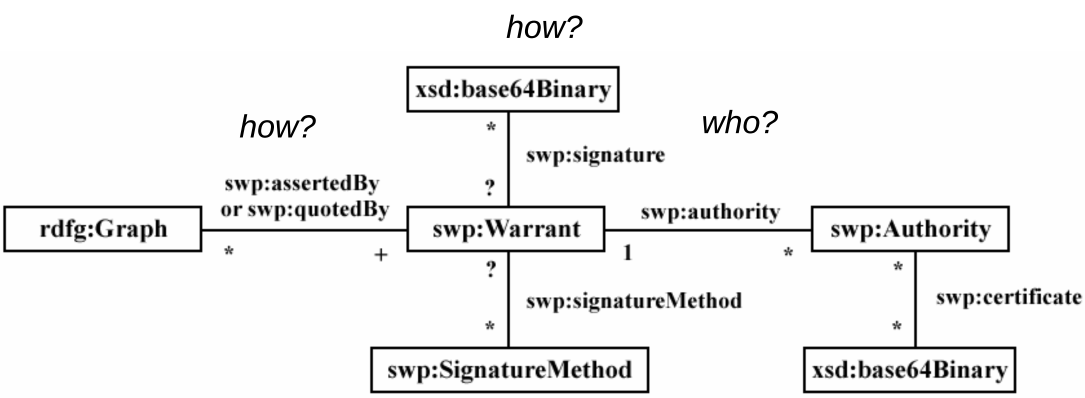
```
:G1 { _:Monica ex:name "Monica Murphy" .
       _:Monica ex:email <mailto:monica@murphy.org>.
       :G1 pr:disallowedUsage pr:Marketing }

:G2 { :G1 ex:author :Chris .
       :G1 ex:date "2003-09-03"^^xsd:date }
```

Semantic Web Publishing

- Determining which graphs we trust depends on WWWWW: *who*, *what*, *where*, *when* and *how*
- named graphs are the hooks on which we pin: *who*, *where*, *when* and *how*. Graph itself answers *what*.
- new SWP vocabulary allows for *who*, *how* (either signed or unassigned; asserted or quoted)
- DC usable for *where*, and *when*
- bootstrapping issue for *what,* and self-describing graphs
  - the meaning of the graph is given by the RDF Semantics, which means we have to (provisionally) accept the graph to determine whether to accept it.

Semantic Web Publishing - Vocabulary



Semantic Web Publishing - Example

```
:G1 { :Monica ex:name "Monica Murphy" .
       :G1  swp:assertedBy _:w1 .
       _:w1 swp:authority _:a .
       _:a  foaf:mbox <mailto:chris@bizer.de> }

:G2 { :G1 swp:quotedBy _:w2
  _:w2 swp:signatureMethod swp:std-method-A^^xsd:anyURI .
  _:w2 swp:signature "..."^^xsd:base64Binary .
  _:w2 swp:authority _:s .
  _:s  swp:certificate "..."^^xsd:base64Binary .
  _:s  foaf:mbox <mailto:patrick.stickler@nokia.com> .
  :G2  swp:assertedBy _:w3 .
  _:w3  swp:signatureMethod swp:std-method-A^^xsd:anyURI .
  _:w3  swp:authority _:s .
  _:w3  swp:signature "..."^^xsd:base64Binary }
```


Example Policies

First one now, then some formal semantics, then Chris presents another

Example 1: Knowledge Base Integration

- Policy: Believe everything that has been
  - explicitly asserted and signed,
  - while maintaining a consistent knowledge base.

Algorithm (non-deterministic and lazy)

K is initial KB (possibly empty or not)

1. Set A := {}
2. Non-deterministically choose $n \in$ domain(N) − A, or terminate.
3. Set K0 := K $\cup$ N(*n*), provisionally assuming N(*n*). (***Bootstrapping***)
4. If K0 is inconsistent then backtrack to 2.
5. Evaluate query on OWL closure K0:
```
SELECT  ?certificate ?method ?sign
WHERE ( n swp:assertedBy ?w1 .
        ?w1 swp:authority ?s .
        ?w1 swp:signatureMethod ?method .
        ?w1 swp:signature ?sign )
      ( ?s swp:certificate ?certificate )
```
6. If **?certificate** is OK, and **?sign** is a signature according to **?method** using **?certificate**, then set K := K0 and A := A $\cup$ {*n*}, otherwise backtrack to 2.
7. Repeat from 2.

Step 4 should be evaluated lazily. Step 6 could check that authority is trusted.

Bootstrapping: Wittgenstein – On Certainty

- (as presented by Gerd Brand: *The Central Texts of Wittgenstein*)
- Doubt only exists as doubting conduct. I cannot live with doubting conduct alone. On the contrary, my normal life is for the most part a non-doubting conduct. [...] doubting behaviour exists only by reason of a non-doubting behaviour, [...] at the beginning stands not-doubting.
- Because doubt rests on what cannot be doubted [...] I cannot arrive at a genuine doubt as long as I want to doubt everything,
- A doubt without end is not even a doubt, just as to want to doubt everything means not even coming to doubt.
- Wittgenstein: "There are cases where doubt is unreasonable, but others where it seems logically impossible. And there seems no clear boundary between them." (OC 454)

Semantics of Signatures

- Extension of RDF Semantics
  - adds legal persons and their agents to domain of discourse
  - interpretation of ***swp:certificate*** restricted by identifying information in certificates
  - interpretation of ***swp:signature*** restricted by the signature method applied to the asserted or quoted graph with the certificate of the authority.
  - Note certificate validity not checked at this stage, but is part of trust policy
  - details in paper (credit Hayes and Carroll)

Self-Asserting Graphs as Performatives

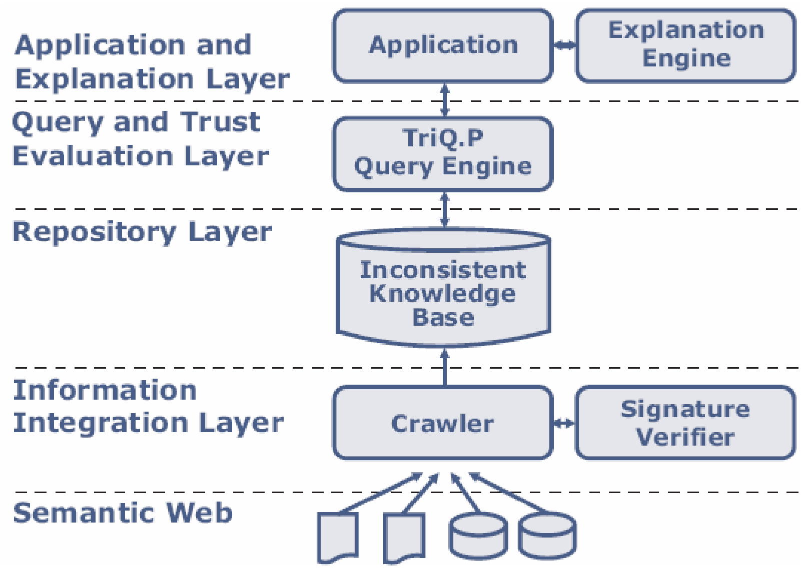
```
:G1 { :Monica ex:name "Monica Murphy" .
    :G1  swp:assertedBy _:w1 .
   _:w1 swp:authority _:a .
   _:a  foaf:mbox <mailto:chris@bizer.de> }
```

- **Performatives: e.g. “I do”, “I promise to pay ...”**
- **Asymmetric semantics: means more to first person, (as social convention).**
- **When Chris interprets :G1 then it is necessarily true, (subject to signatures or other verification)**
- **Other people might not trust Chris**
- **Details in paper (Hayes)**

Example 2: The TriQL.P Trust Architecture

- TriQL.P is a query language, that allows the formulation of trust policies within queries
  - similar to RDQL
  - uses graph patterns
  - supports set operations and different ranking mechanisms
  - returns justification trees together with the query results
- Justification trees
  - provide explanations why data should be trusted
  - can be used to implement Tim Berners- Lee’s “Oh, yeah?” button.

Architecture



Example

- Application domain: Skill management
- Query: Retrieve all persons with the skill "Programming“.
- Query specific, context-based Trust Policy:
  Use only claims by people who have an affiliation to at least 2 projects involving “Programming”.

TriQL.P Query

```
SELECT ?person
WHERE ?graph (?person km:skill km:Programming .
              ?person rdf:type km:Person )
             (?graph swp:assertedBy ?warrant .
              ?warrant swp:authority ?author )
             (?author km:affiliation ?project )
             (?project rdf:type km:Project .
              ?project km:topic km:Programming )
AND COUNT(?project) >= 2
USING km FOR <http://www.example.org/vocabulary#>
      rdf FOR <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
      swp FOR <http://www.w3.org/2004/03/trix/swp-1/>
```


Query Result and Justification Tree

?person = http://www.example.org#Monica

Claimed in	Justification Bindings
ex:Graph1	?graph (?person km:skill km:Programming . ?person rdf:type km:Person) ?graph = ex:Graph1 ?person = ex:Monica
ex:Graph2	(?graph swp:assertedBy ?warrant . ?warrant swp:authority ?author) ?graph = ex:Graph1 ?warrant = ex:Warrant2 ?author = ex:Chris
ex:Graph3	(?author km:affiliation ?project) ?author = ex:Chris ?project = ex:projectInterVal
ex:Graph3	(?author km:affiliation ?project) ?author = ex:Chris ?project = ex:projectKnowledgeNet
ex:Graph4	(?project rdf:type km:Project . ?project km:topic km:Programming) ?project = ex:projectInterVal
ex:Graph5	(?project rdf:type km:Project . ?project km:topic km:Programming) ?project = ex:projectKnowledgeNet

Summary

- Highlighted the need for subjective, task-specific trust policies
- Proposed using context- and content-based trust mechanisms, beside of reputation-based mechanisms
- Proposed extending RDF to Named Graphs
- Proposed the Semantic Web Publishing Vocabulary

Key References and Credits

- http://www.w3.org/2004/03/trix/
  - Named Graph Website
  - Links to TriX, TriG, TriQL, RDFQ Specs
- http://www.hpl.hp.com/techreports/2004/HPL-2004-57.html
  - *Named Graphs, Provenance and Trust*
    Carroll, Bizer, Pat Hayes and Patrick Stickler
- http://www.hpl.hp.com/techreports/2004/HPL-2004-56.html
  - *TriX: RDF Triples in XML*
  - update of initial named graphs paper by Carroll and Stickler
- http://www.wiwiss.fu-berlin.de/suhl/bizer/TriQLP
  - More information about the TriQL.P trust architecture
- All papers, specs and sites are early versions, this is work-in-progress

Thanks :-)