TECHNISCHE UNIVERSITÄT
CHEMNITZ

# *Current Trends in Web Engineering*

***Prof. Dr.-Ing. Martin Gaedke***
***Dr.-Ing. Sheeba Samuel***

Technische Universität Chemnitz

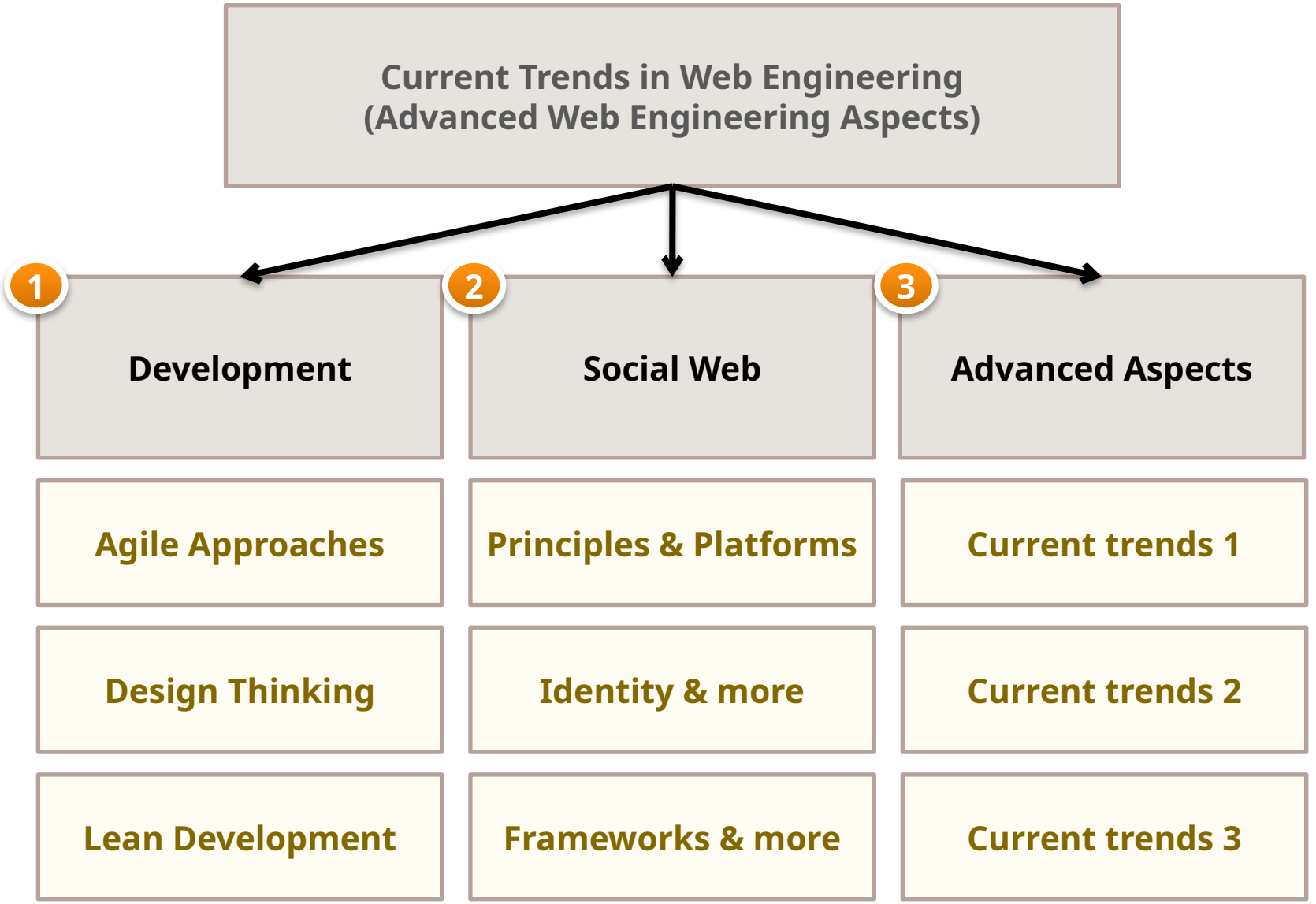
Fakultät für Informatik

Verteilte und selbstorganisierende Rechnersysteme

Part II

# Social Web

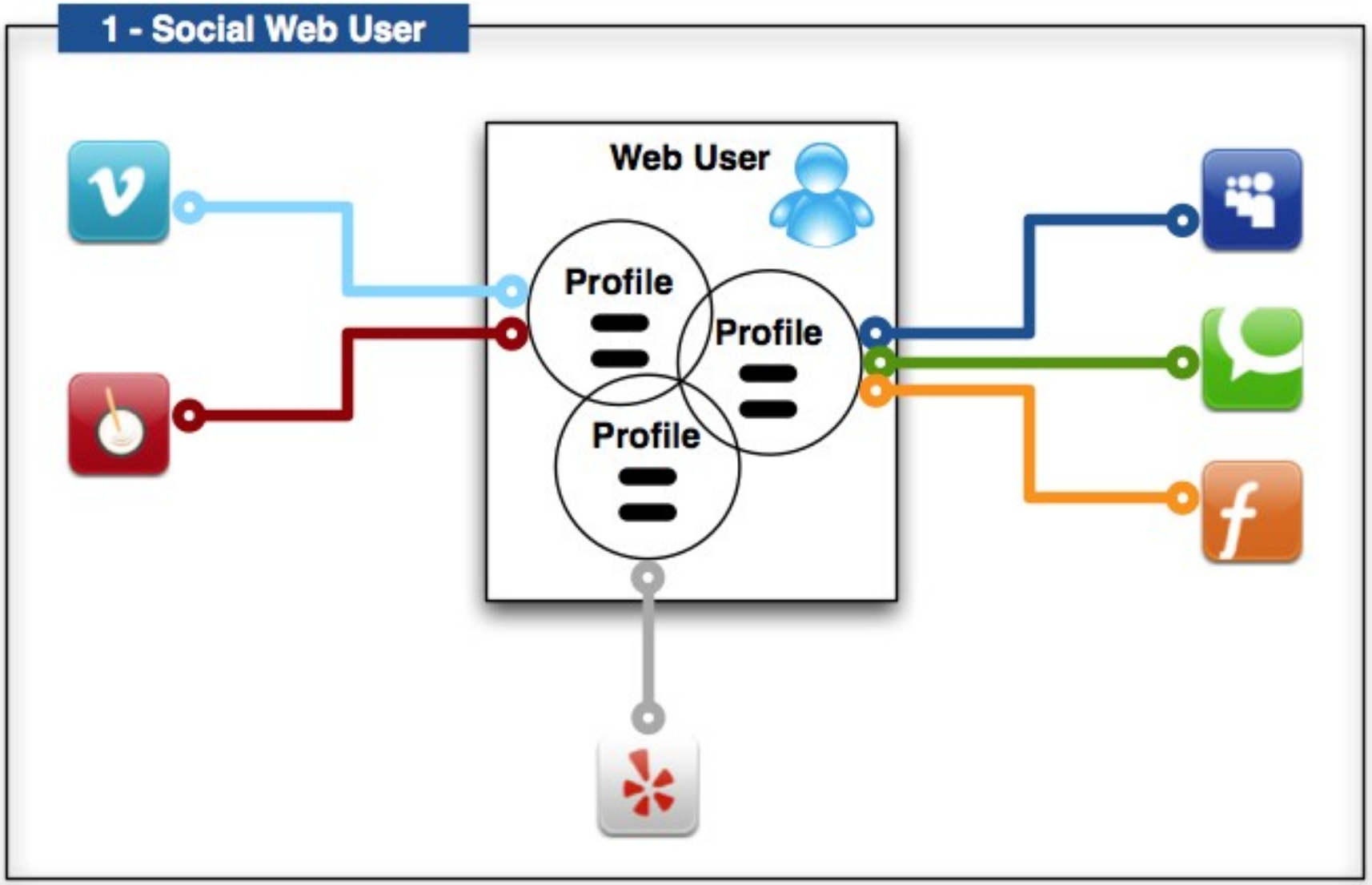
# Lecture Outline



# Social Web: Additional Materials

- The following materials provide you with an overview on important aspects of the Social Web, challenges, protocol and research.
- In the live session, we will focus on Solid
- Reading the additional materials will widen your knowledge on Social Web and deepen your understanding of Solid

# Identity and the Social Web



# Challenges

- How to represent actors in the Social Web?
- How to share data between applications?
- How to avoid identity theft?
- How to keep data up-to-date?
- How to guarantee privacy?
- …

# Terminology: Identity

- **Digital Subject** – is a person or entity from some domain, which is being described
- **Claim** – is a controversial or disputable assertion
- **Digital Identity** – a set of claims about a digital subject made either by the subject him/herself or by any other party (Digital Identity ≠ Unique Identifier)

- Example:

  - Digital Subject:
  - Claim: Bob is adult -> How to verify this claim?
  - Digital Identity: (Name: Bob, Role: ScrumMaster)

# Terminology: Identity

- **Digital Identity System** is a framework for description, management, exchange, exposure and verification of digital identities



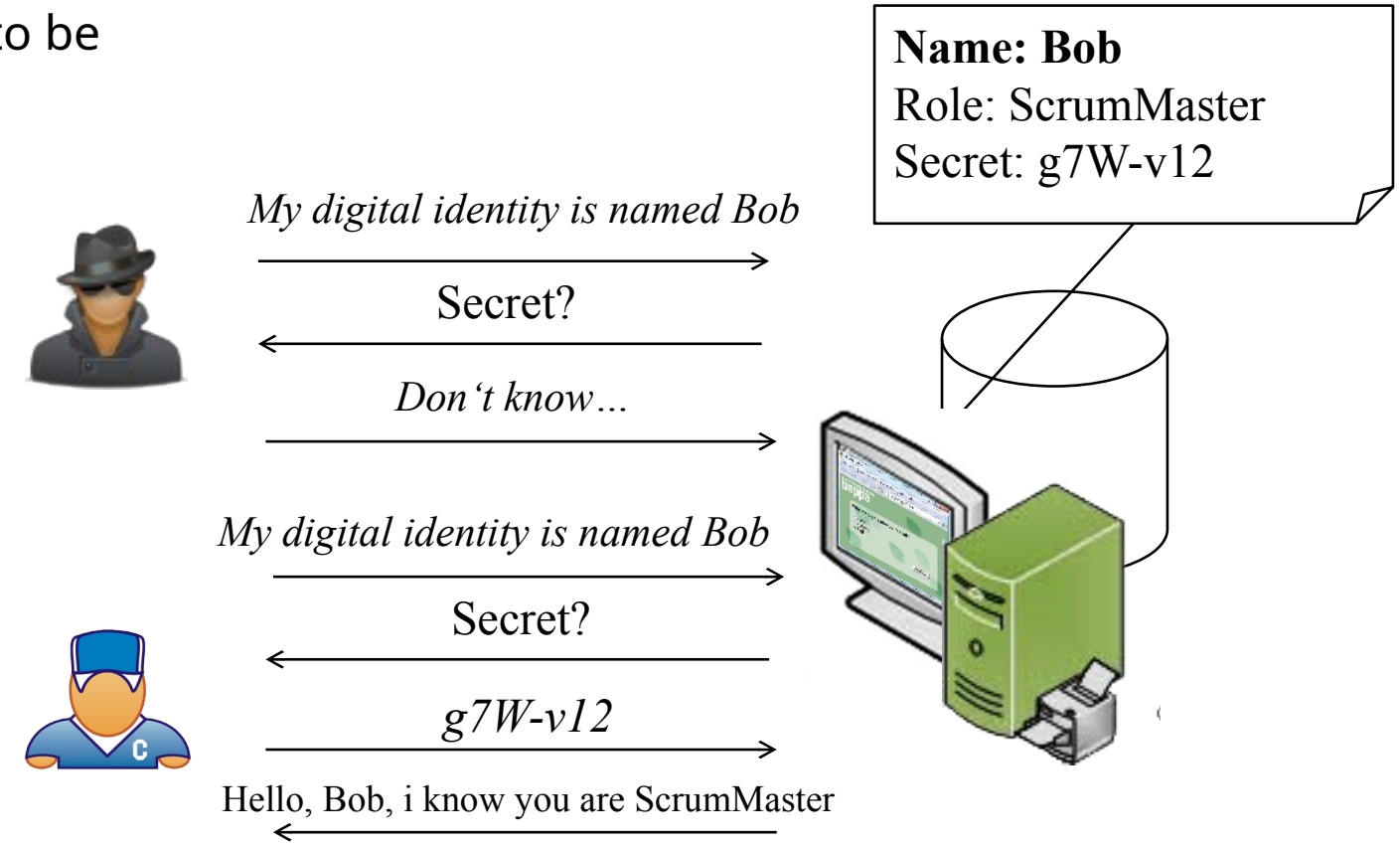






# Terminology: Authentication

- **Authentication** is the process of verification if someone is the one who he claims to be

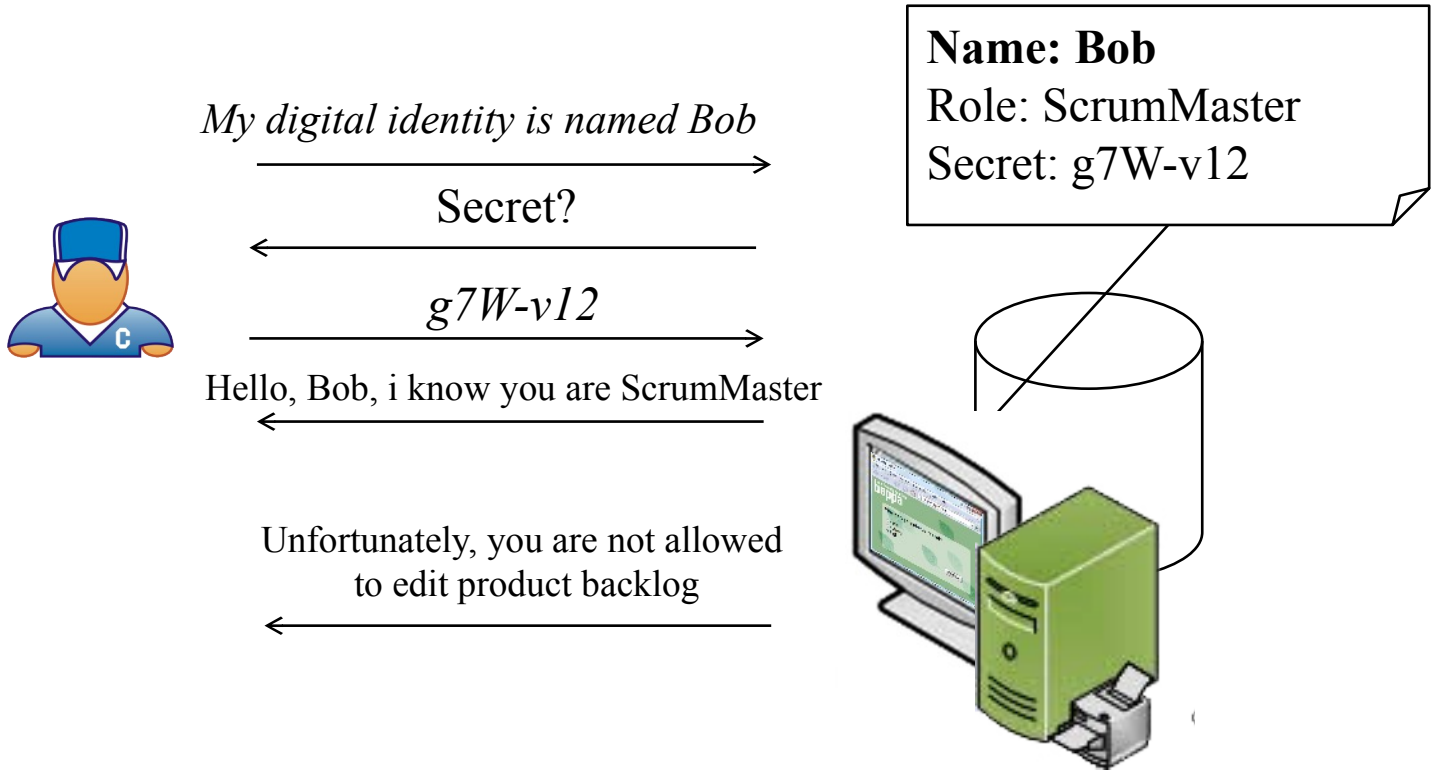


# How to verify a claim of identity possession?

- What you know: Knowledge-based
  - PIN, Passwords
  - Challenge-Response
- What you are: Biometrics
  - Fingerprint, Iris, Language, Signature
- What you have: Possession-based
  - Something, what is unable to remember and / or is stored on a medium
  - ID cards, magnet strip cards, certificates, smartcards
- Multi-factor-authentication
  - Combination of several methods
  - 2-factor: Bank card + PIN, Credit card + Signature
  - 3-factor: Password + Smartcard + Fingerprint

# Terminology: Authorization

- **Authorization** is the process of verification if someone is allowed to perform some actions



Chapter://1

# Identity Systems: An Overview

Section://1

# OpenID

# OpenID – Introduction


- Problem:
  - Cumbersome management of scattered identity data
  - Cumbersome management of usernames and passwords
- Goals:
  - Single sign-on
  - Distributed authentication system
  - Lightweight protocol

# OpenID – Concepts

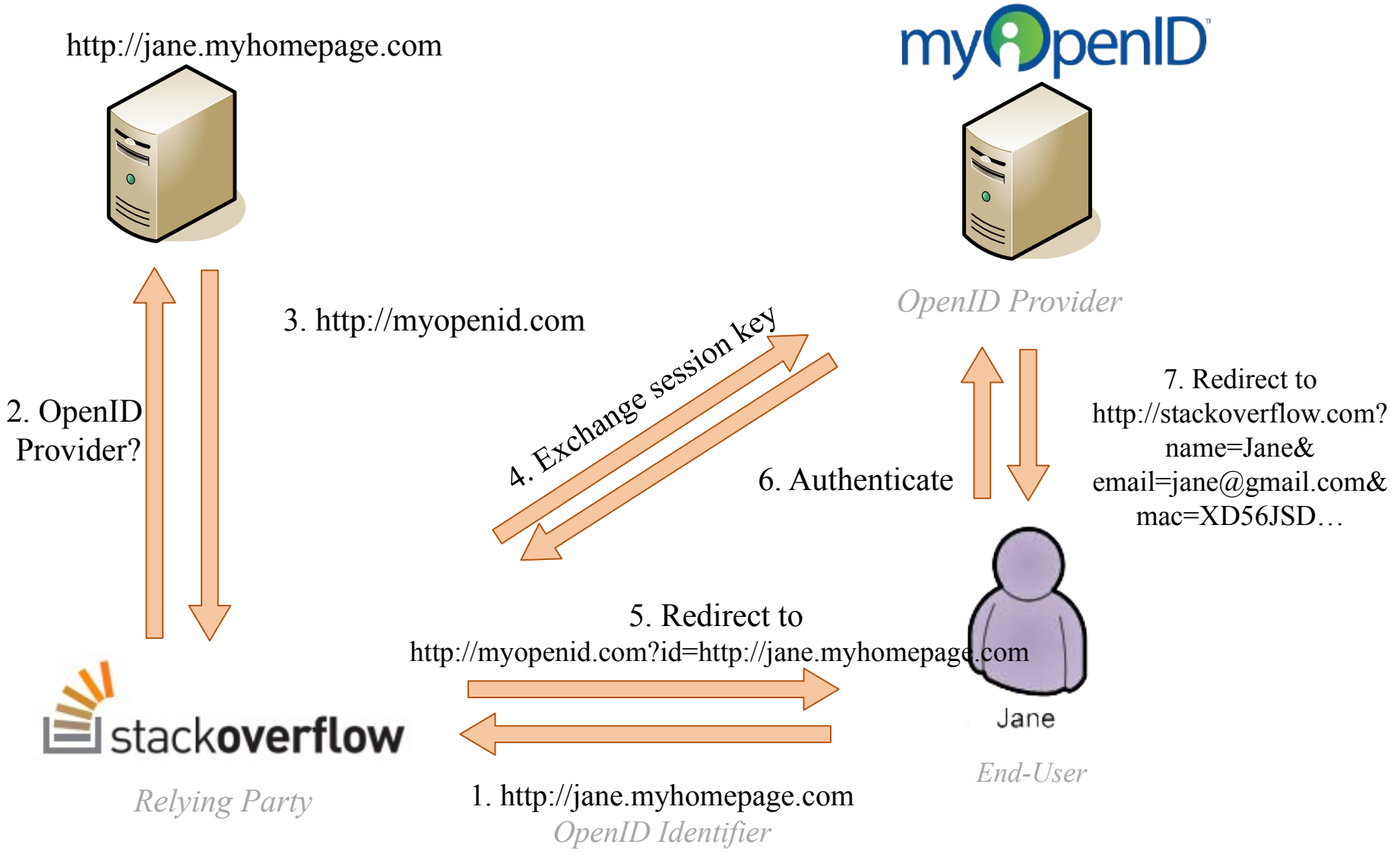
- End-User
  - Entity, which the statements are made about
- OpenID (Identifier)
  - URI or XRI of the end-user
- OpenID Provider (Identity Provider)
  - Identity management
- Relying Party
  - Third party, which wants to know the identity
- User-Agent
  - Browser of the end-user


**http://jane.myhomepage.com**

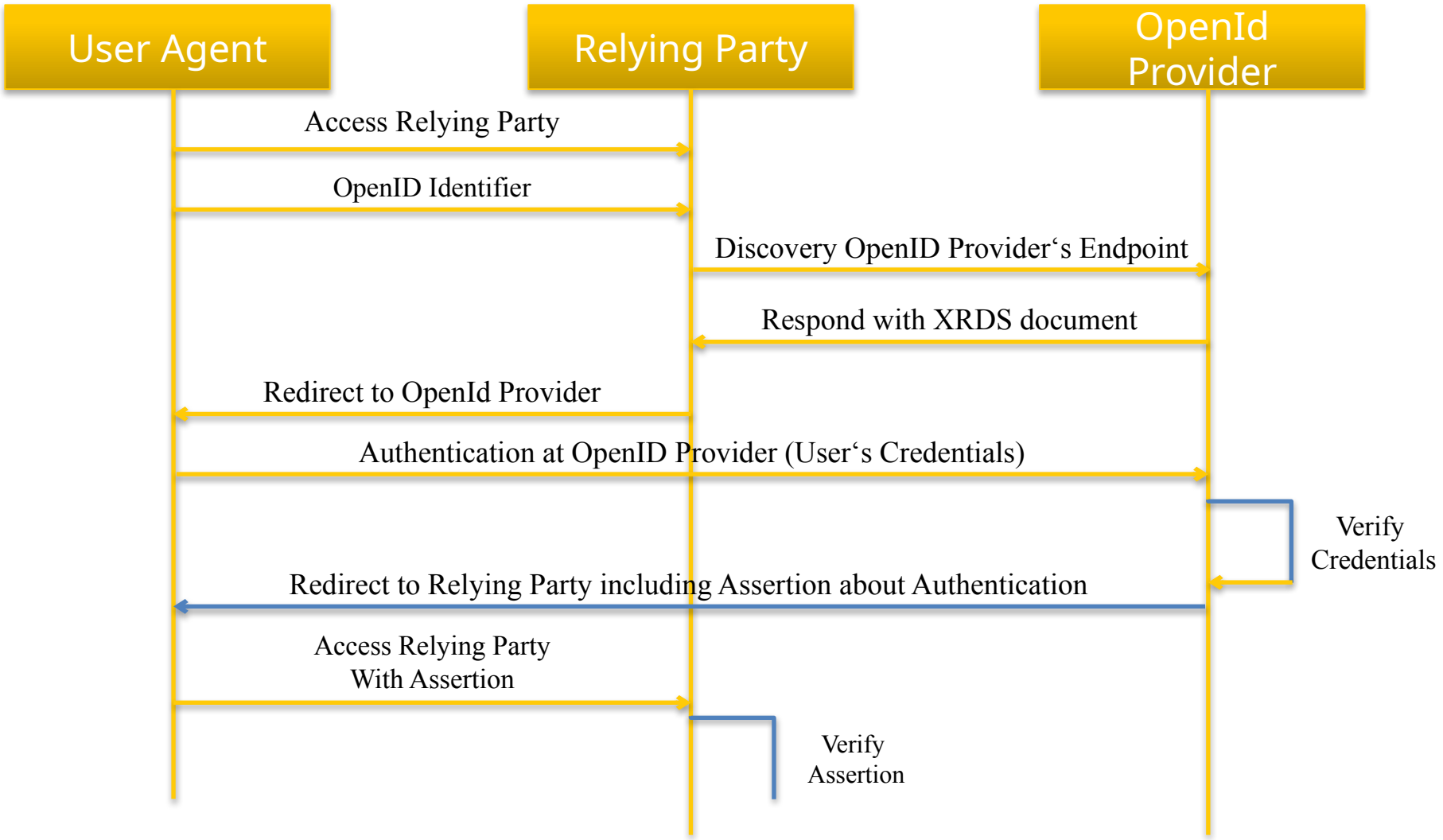
myOpenID

stackoverflow

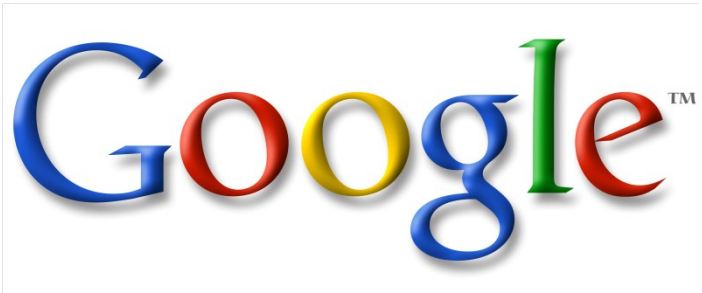
# OpenID – Idea



# OpenID – Flow



# Demo


Test pages:

- http://www.livejournal.com/identity/login.bml?type=openid
- http://slashdot.org/my/newuser
- http://stackoverflow.com/users/login
- https://verify.sxip.com/demorp/

Section://2

# Mozilla Persona

# Mozilla Persona


- Use E-Mail addresses as identity identifiers
- E-Mail provider becomes the identity provider
  - Mozilla acts as a “Fallback Identity Provider” if the original provider doesn’t support the protocol
- Goal: Easy-to-use and easy-to-implement Single Sign-On

# Mozilla Persona: Actors

- **Users:** The actual people that want to sign into websites using Persona.
- **Relying Parties (RPs):** Websites that want to let users sign in using Persona.
- **Identity Providers (IdPs):** Domains that can issue Persona-compatible identity certificates to their users.

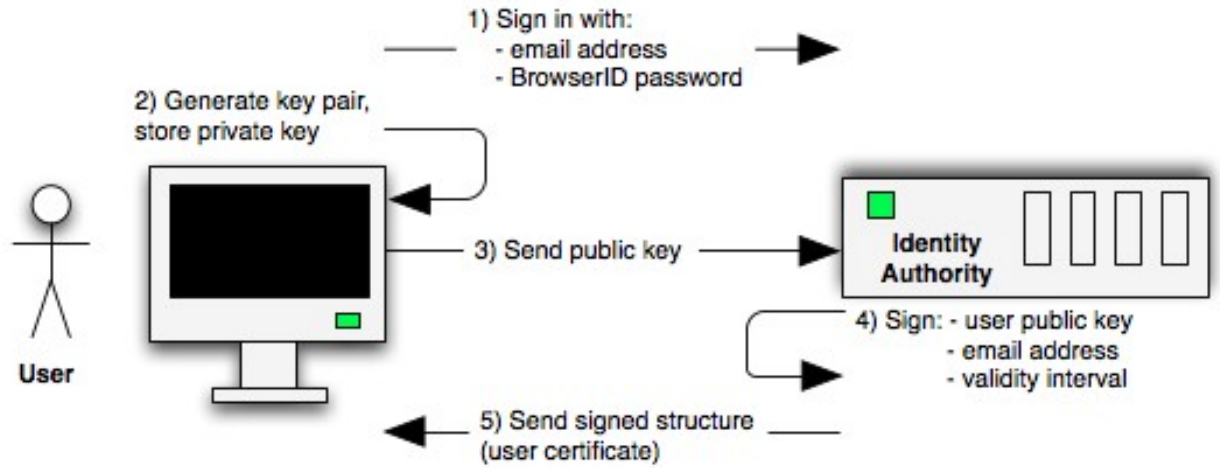
# Mozilla Persona: Idea

- User must be able to prove ownership of their preferred email address to *Relying Party*
- Browser asks the *Identity Provider* to check the ownership and issue a digital certificate confirming the possession
- User presents the certificate to the *Relying Party*, which verifies it using the public key of *Identity Provider*

- Security considerations:
  - To prove, that the certificate is not stolen, certificate includes the public key of the user. User presents both certificate and his signature to the *Relying Party*
  - *Relying Party* fetches public key of the *Identity Provider* to check if the certificate is valid

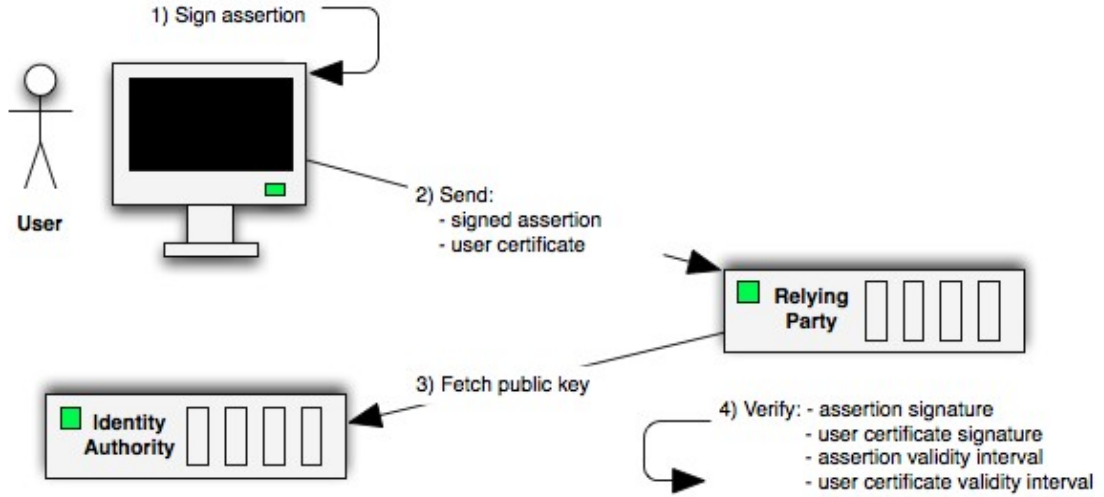
# Mozilla Persona: Protocol

1. User signs in to the *Identity Provider* with preferred e-mail
2. Browser generates public & private key to be used as proof of certificate possession
3. Browser lets *Identity Provider* create / sign a certificate with user's e-mail and the generated public key

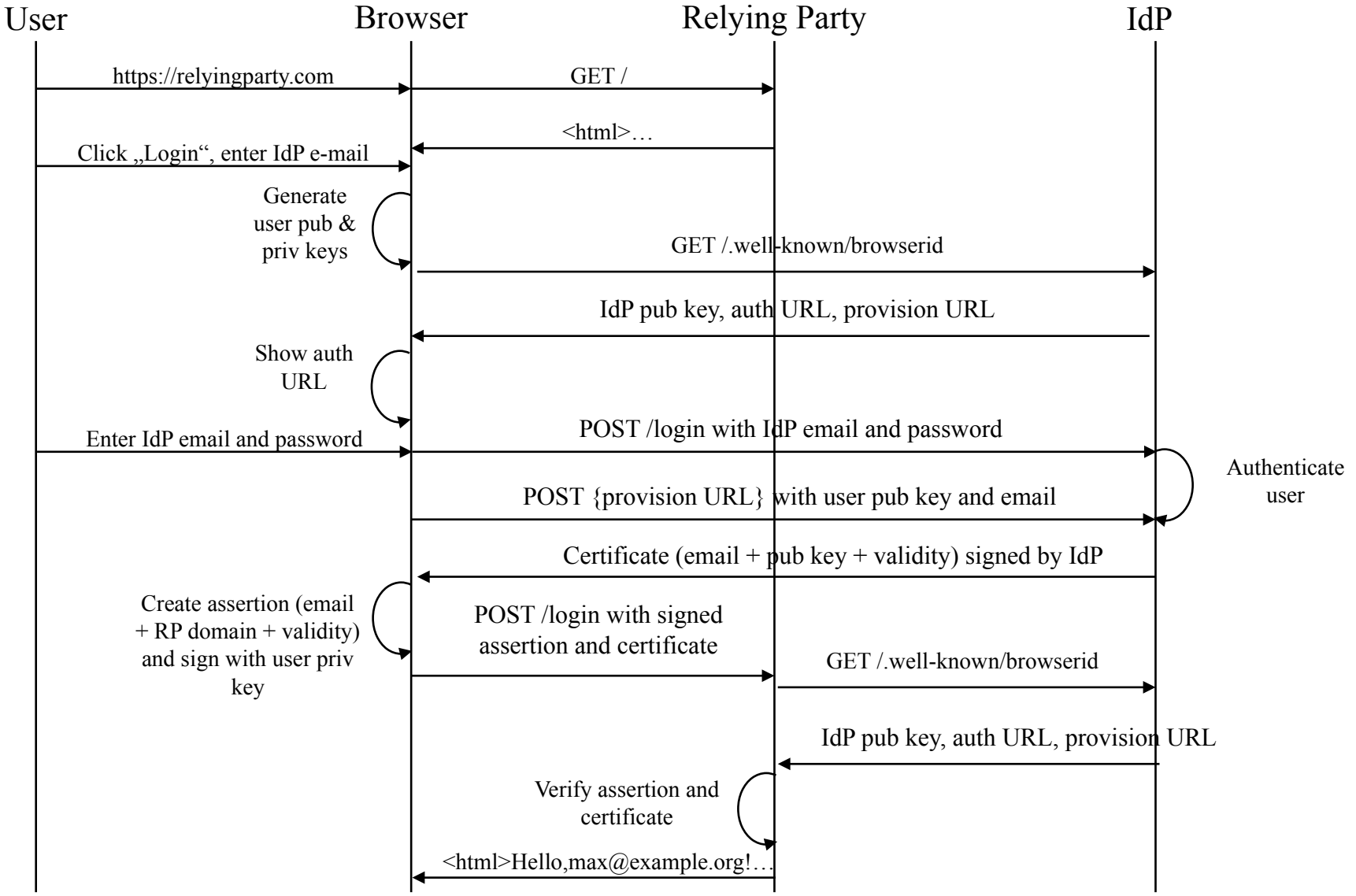


# Mozilla Persona: Protocol (2)

1. Browser creates and signs a so called “assertion” (e-mail, target relying party, expiration date) using the private key
2. Browser presents the assertion and the certificate to *Relying Party*
3. *Relying Party* verifies the assertion and the certificate



# Mozilla Persona: Protocol (2)



Section://3

# OAuth

# OAuth - Introduction

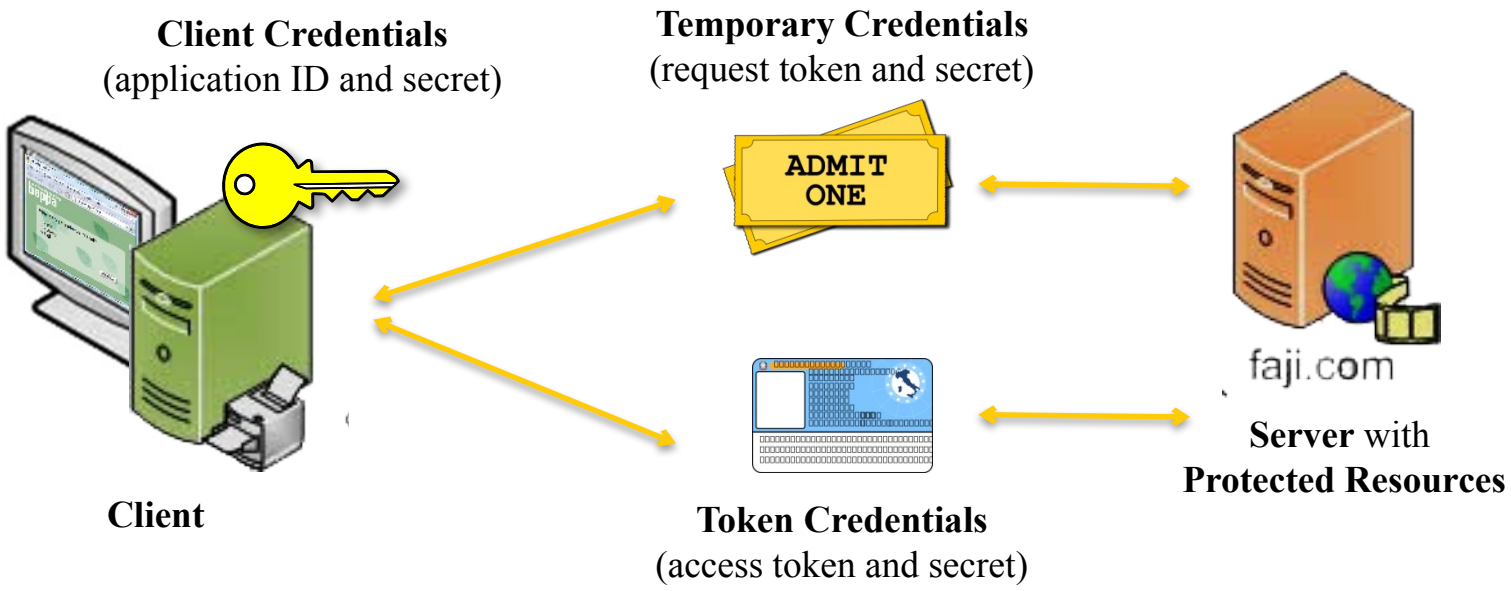

- Problem:
  - How can one allow private resource access without disclosing one's username/password?
  - How can fine-grained access be controlled?
  - How can access time be limited?
- OAuth:
  - Open standard
  - Goal is an authorization delegation protocol: Client accesses private data on behalf of the owner
  - Authorization of APIs for applications based on special tokens being issued

# OAuth – History

- Development since 2006/2007
- April 2010 - OAuth 1.0 (RFC 5849)
- October 2012 - OAuth 2.0 (RFC 6749)
  - Not backward compatible
  - Simpler than OAuth 1.0 (less cryptography)
  - SSL required!

# OAuth 1.0 Terminology


**Resource owner**



# OAuth 2.0 – Roles I

- Resource owner
  - An entity capable of granting access to a protected resource.
  - When the resource owner is a person, it is referred to as an end-user.
- Resource server
  - Hosts the protected resources

# OAuth 2.0 – Roles II

- Client
  - Application, that requests behalf of the **resource owners** and with its authorization
- Authorization server
  - Issues access token, if
    - Client is authenticated
    - resource owner grants access
  - One server can issue multiple Tokens for different **Resource servers**

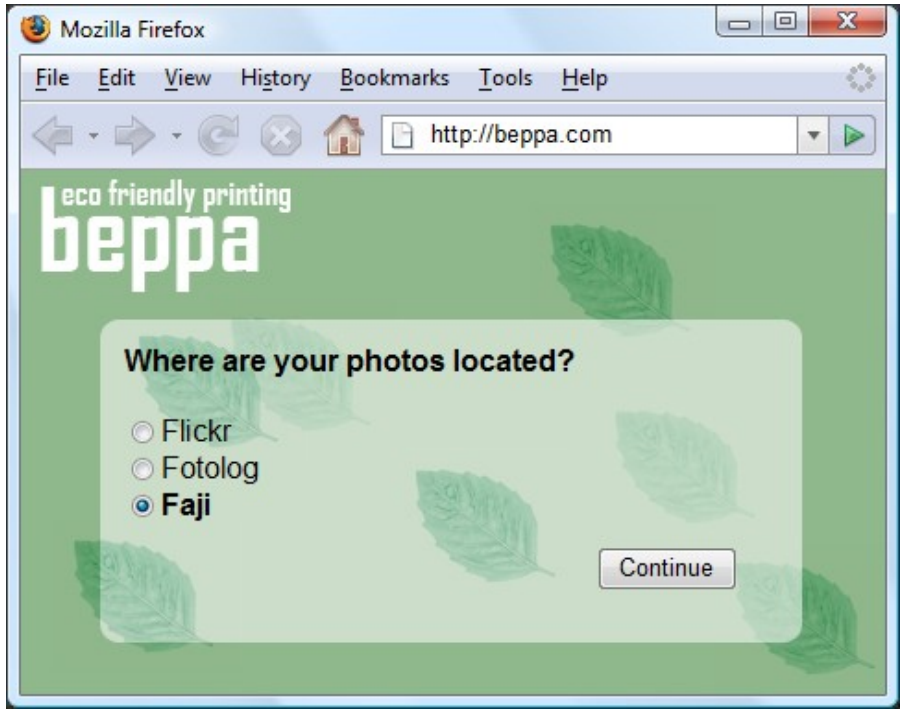
# OAuth Scenario (three-ledged): Part 1

- Janes uses http://faji.com to upload her photos


→ Jane is a **Resource Owner** and Faji.com is a **Server** with **Protected Resources**

# OAuth Scenario: Part 2

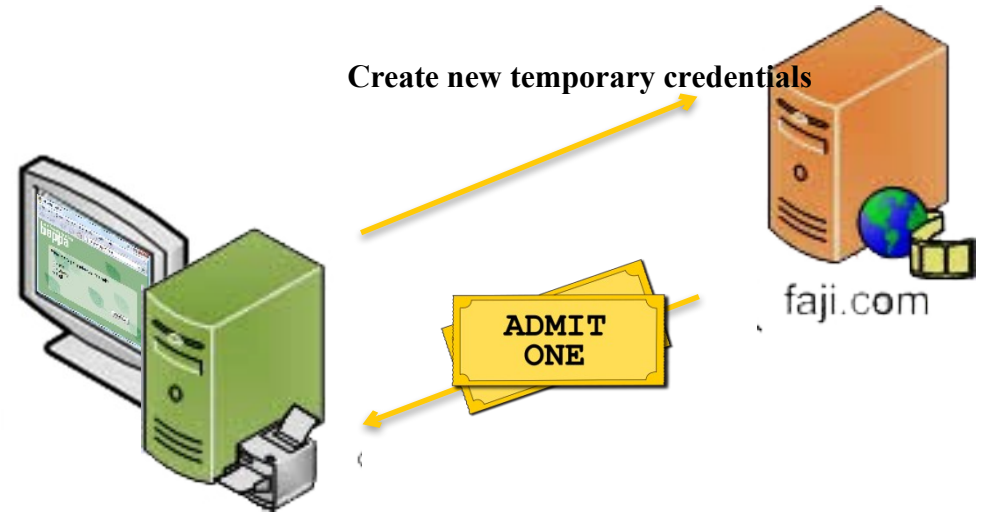
- Jane visits  beppa.com to order prints




→ Beppa.com is a **Client** and has unique **Client Credentials** assigned by Faji

# OAuth Scenario: Part 3

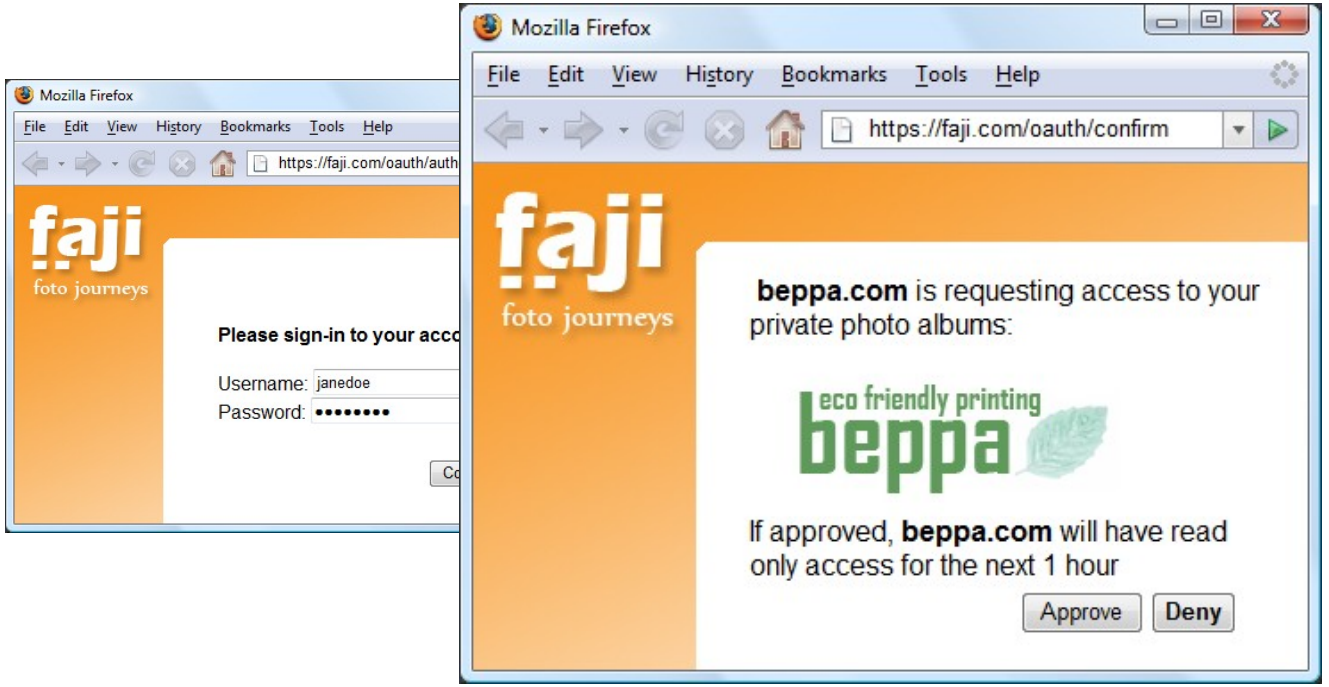
- Beppa.com asks faji.com for **temporary credentials**




- Beppa.com redirects Jane to faji.com, so she can approve the temporary credentials

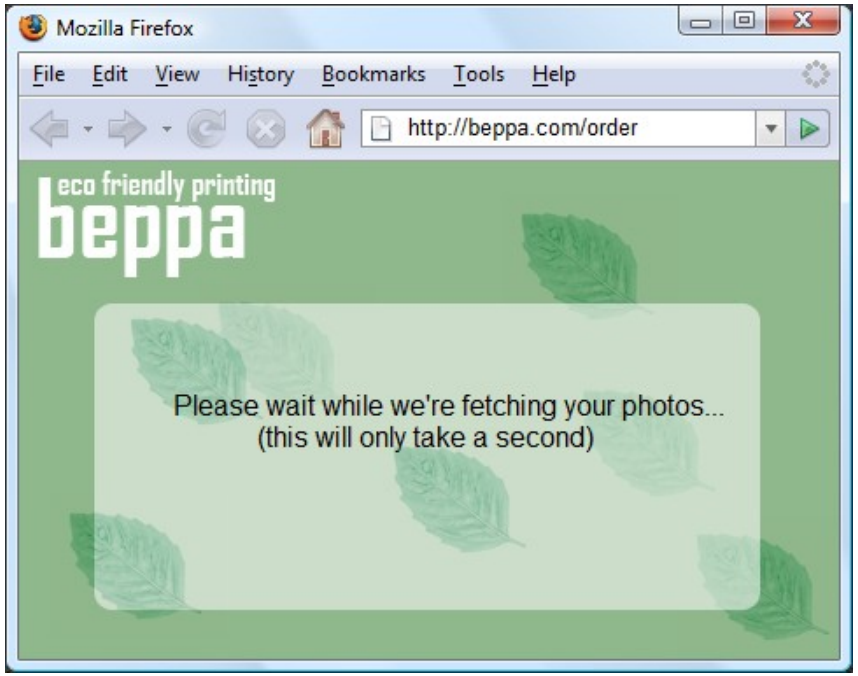
# OAuth Scenario: Part 4

- Jane signs in to faji.com and **approves the temporary credentials** (grants access to beppa.com)



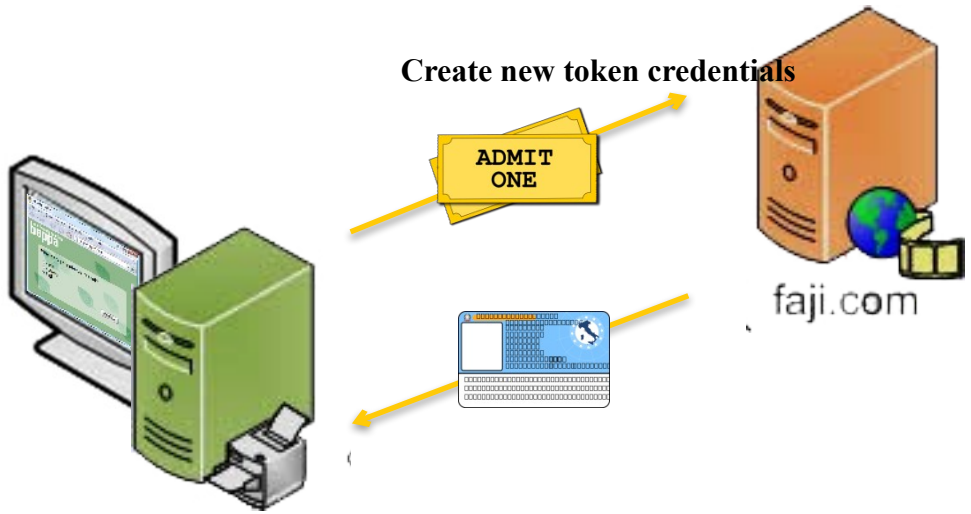
# OAuth Scenario: Part 5

- Temporary credentials of beppa.com become **authorized**
- faji.com redirects Jane back to beppa.com



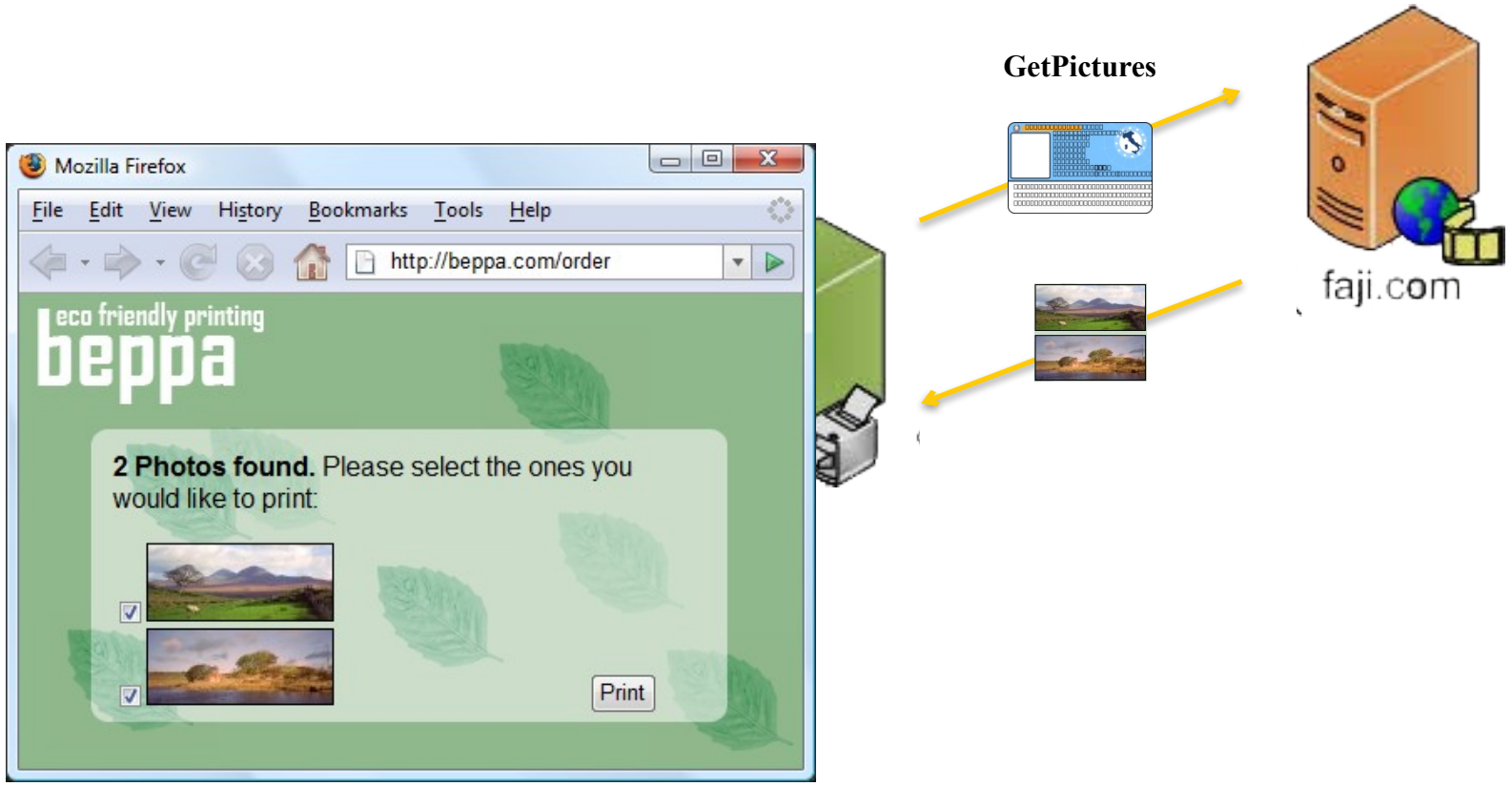
# OAuth Scenario: Part 6

- Beppa.com exchanges temporary credentials for **token credentials**

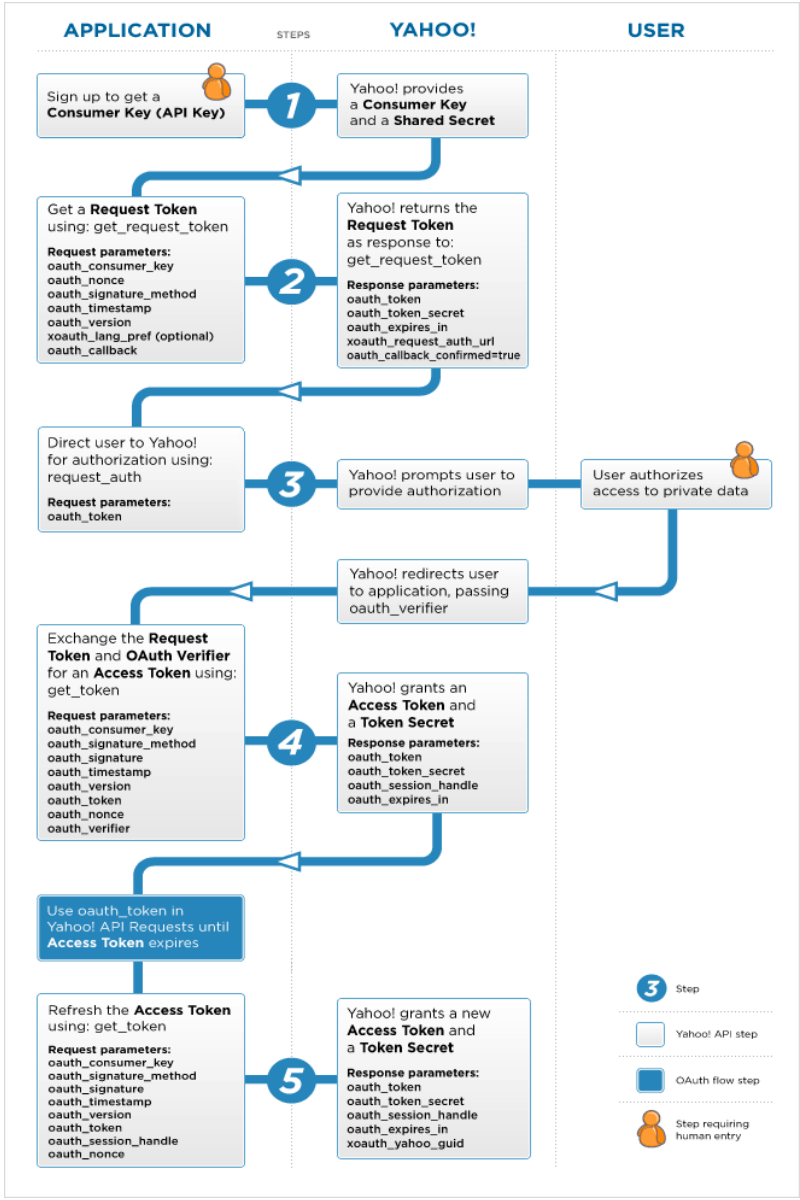


# OAuth Scenario: Part 6

- Beppa.com uses token credentials to fetch Janes photos

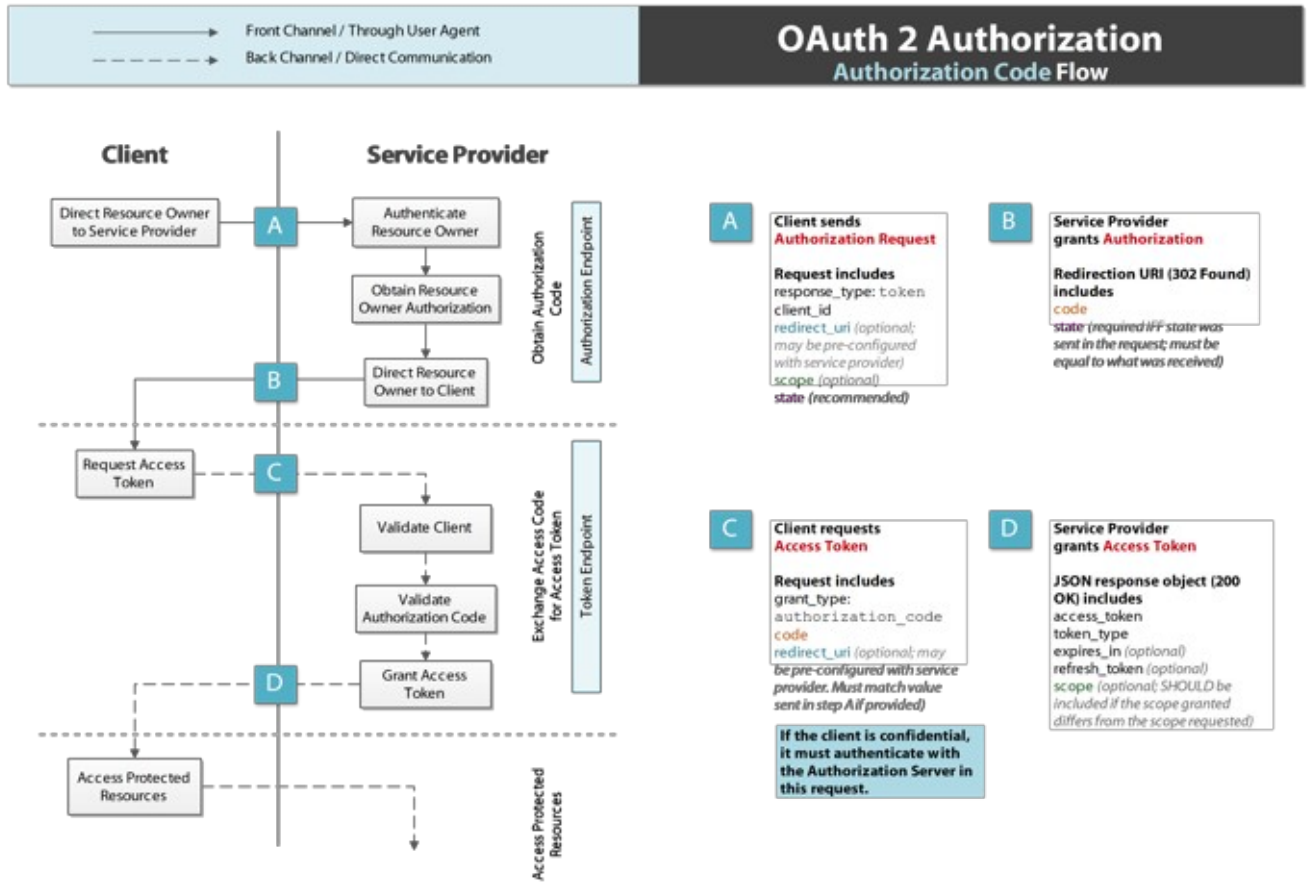


# OAuth 1.0 Example Authorization Flow



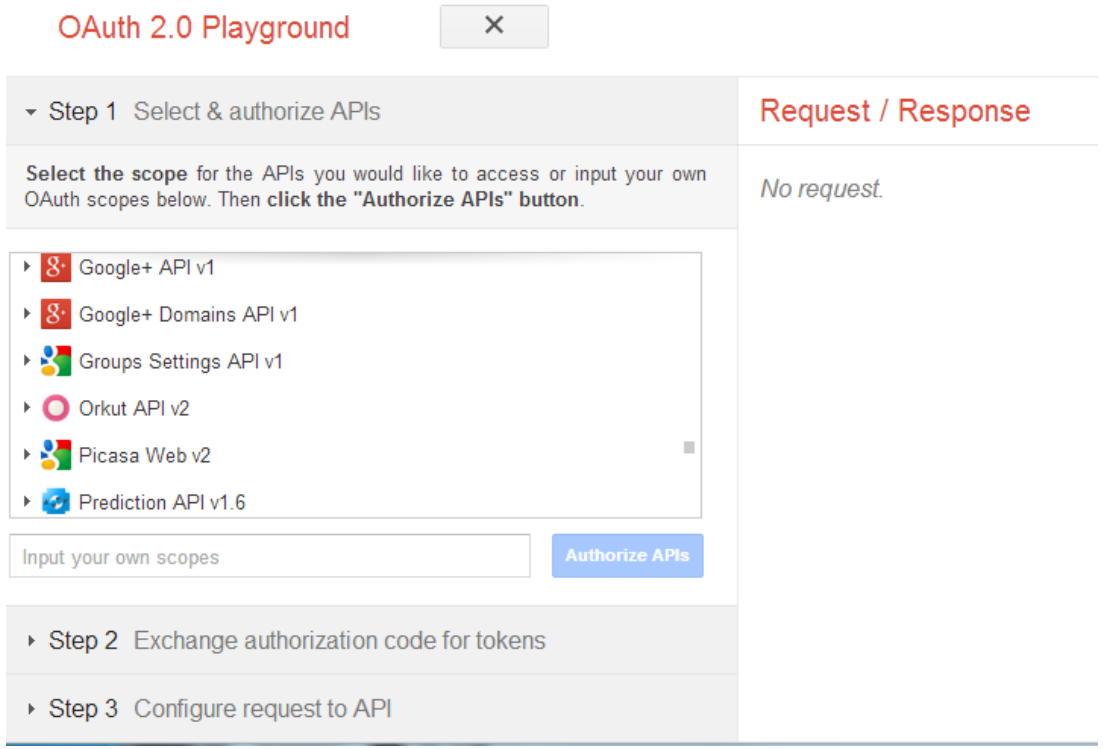
# Oauth 2.0 vs 1.0

- Better support for desktop and mobile applications
- Less cryptography, more TLS



# Demo

- Google Playground:
  - https://developers.google.com/oauthplayground/



Section://4

# WebID and WAC

# WebID – Introduction

- Open standard for identity and authentication
  - W3C initiative
  - Currently in development
  - http://www.w3.org/2005/Incubator/webid/spec/
- Ownership-based authentication
  - Enables users to authenticate via certificates
  - No username/password required
- Two specifications
  - WebID – Web Identity and Discovery
  - WebID-TLS – WebID Authentication over TLS

# WebID – Web Identity and Discovery

- WebID is a universal identification mechanism
  - Is distributed,
  - Is openly extensible,
  - Enables persons to control their identity
  - Allows creating a Web of Trust (WoT)
- WebID makes use of *three artifacts* to establish basis for
  - Identification
  - Discovery
  - Management of user profile data
  - Exploitation of user profile data

# WebID – Artifacts

- WebID URI
- WebID Certificate
- WebID Profile

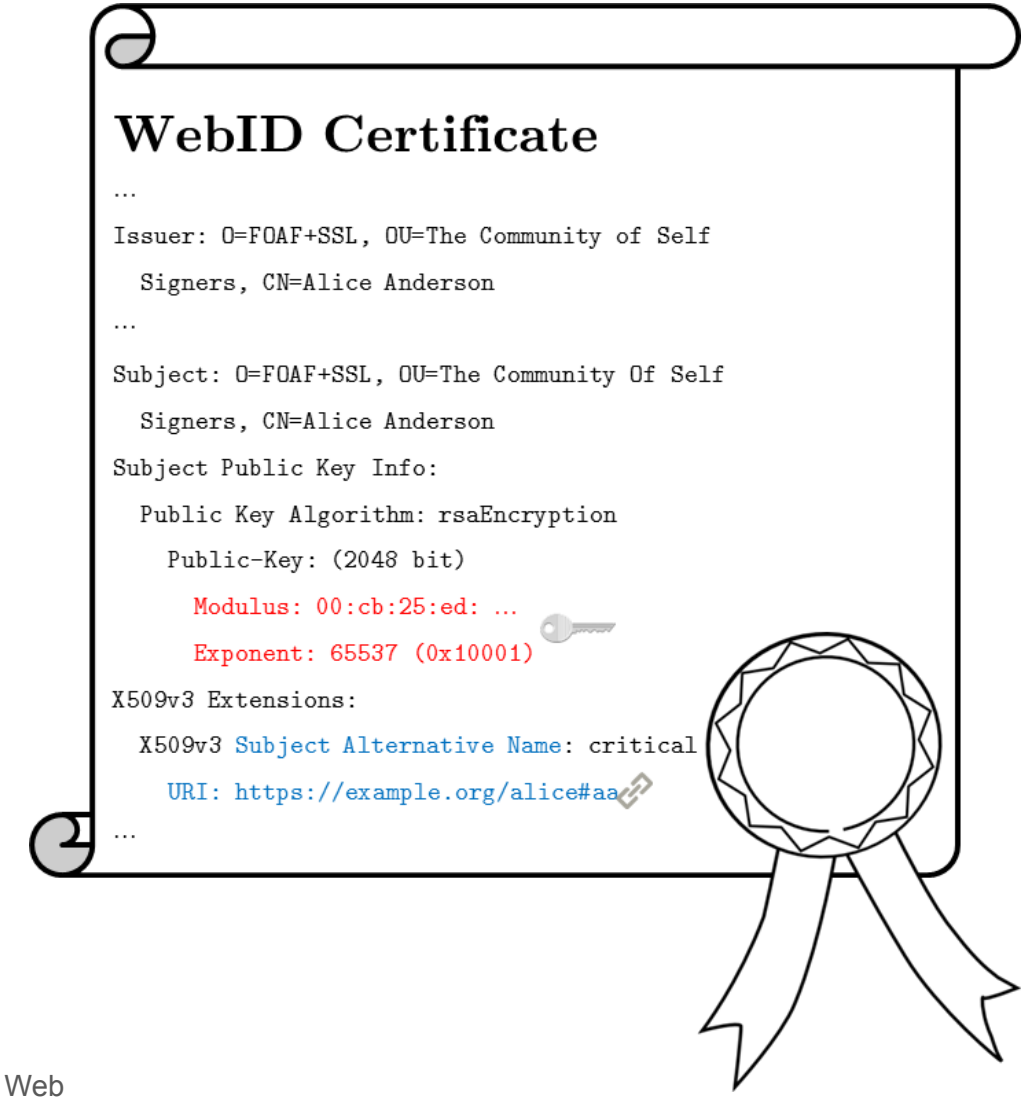
# WebID URI

- WebID URI refers to an agent
  - In most cases, agent is a user
  - Agent can also be robot or group
  - Any entity to be identified
- WebID URI is an identifier
  - Cf. username, e-mail address in other identity systems
- WebID URI links to a resource containing further data about the identity owner
  - Resource is called WebID Profile

# WebID Certificate

- WebID Certificate is typical X509.v3 client certificate with
  - Subject name
  - Issuer name
  - Public key
  - Signature
  - Validity
  - Subject Alternative Name property stores WebID URI
- Private key is stored on client
- Users allowed to self-sign certificate (recommended)
- Signing by certificate authority (CA) is also possible

# WebID Certificate



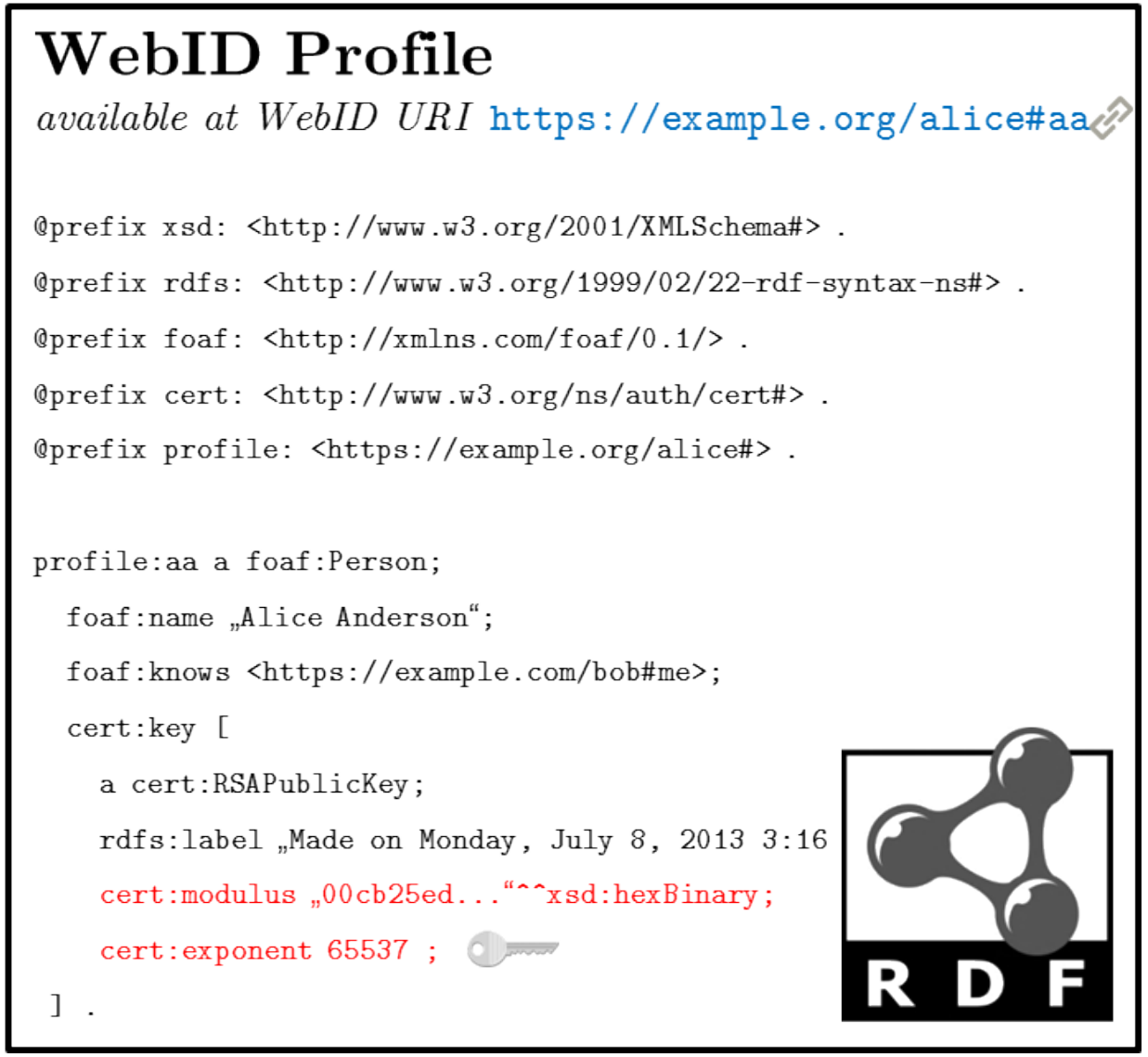
# WebID Profile

- Repository of identity owner's personal information
- Personal information described using Linked Data
  - Flexible
  - Extensible
  - Machine-readable
- WebID profile is also used to store public keys

# WebID Profile (2)

- Various representation of a WebID profile
  - Turtle (has to be supported)
  - N3, RDF/XML, RDFa etc.
- Use of FOAF as a vocabulary for personal data
  - Allows creating decentralized social network
  - http://xmlns.com/foaf/spec/
- Cert ontology for describing cryptographic information
- Use of any other RDF vocabulary, e.g., PIM ontology for describing extended contact data

# WebID Profile



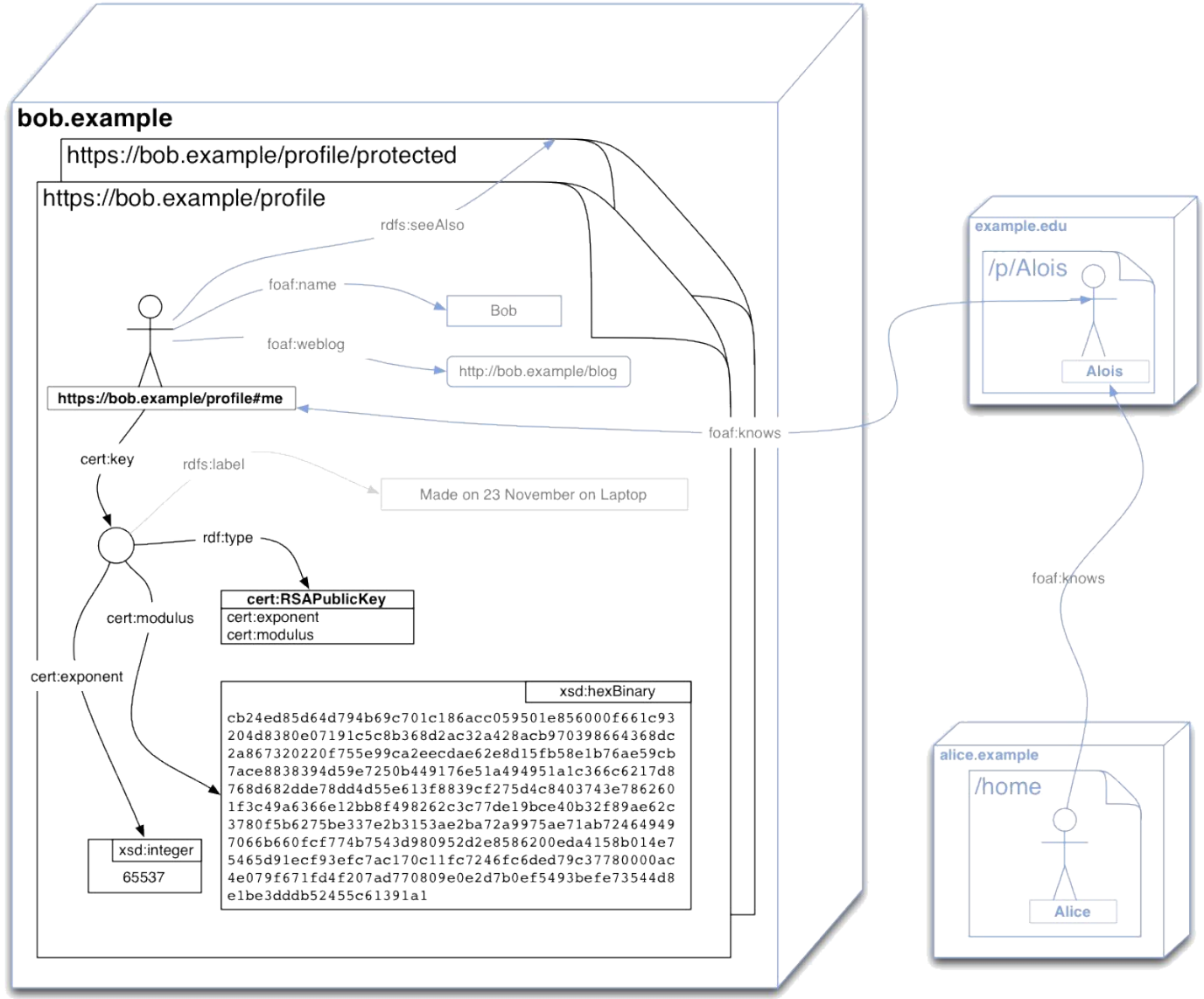
**WebID Profile**

*available at WebID URI* https://example.org/alice#aa

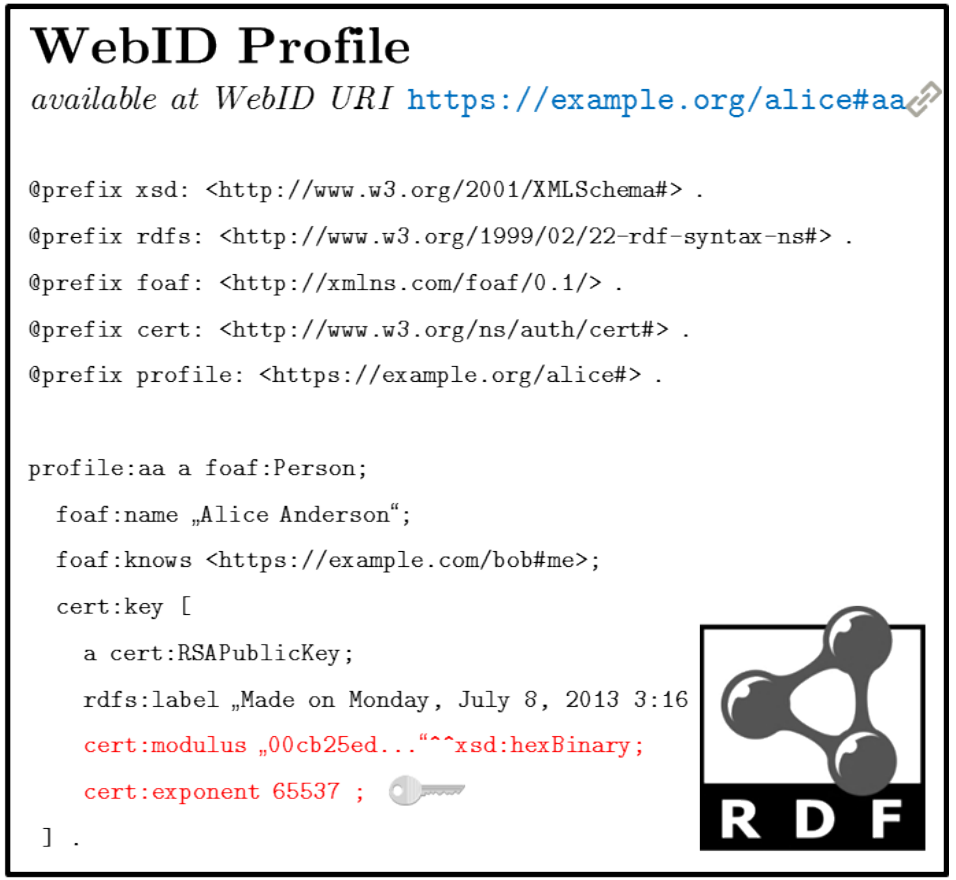
```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix cert: <http://www.w3.org/ns/auth/cert#> .
@prefix profile: <https://example.org/alice#> .

profile:aa a foaf:Person;
  foaf:name „Alice Anderson“;
  foaf:knows <https://example.com/bob#me>;
  cert:key [
    a cert:RSAPublicKey;
    rdfs:label „Made on Monday, July 8, 2013 3:16
    cert:modulus „00cb25ed...“^^xsd:hexBinary;
    cert:exponent 65537 ;
 ] .
```

RDF

# WebID Profile - Graph



# WebID – Artifacts

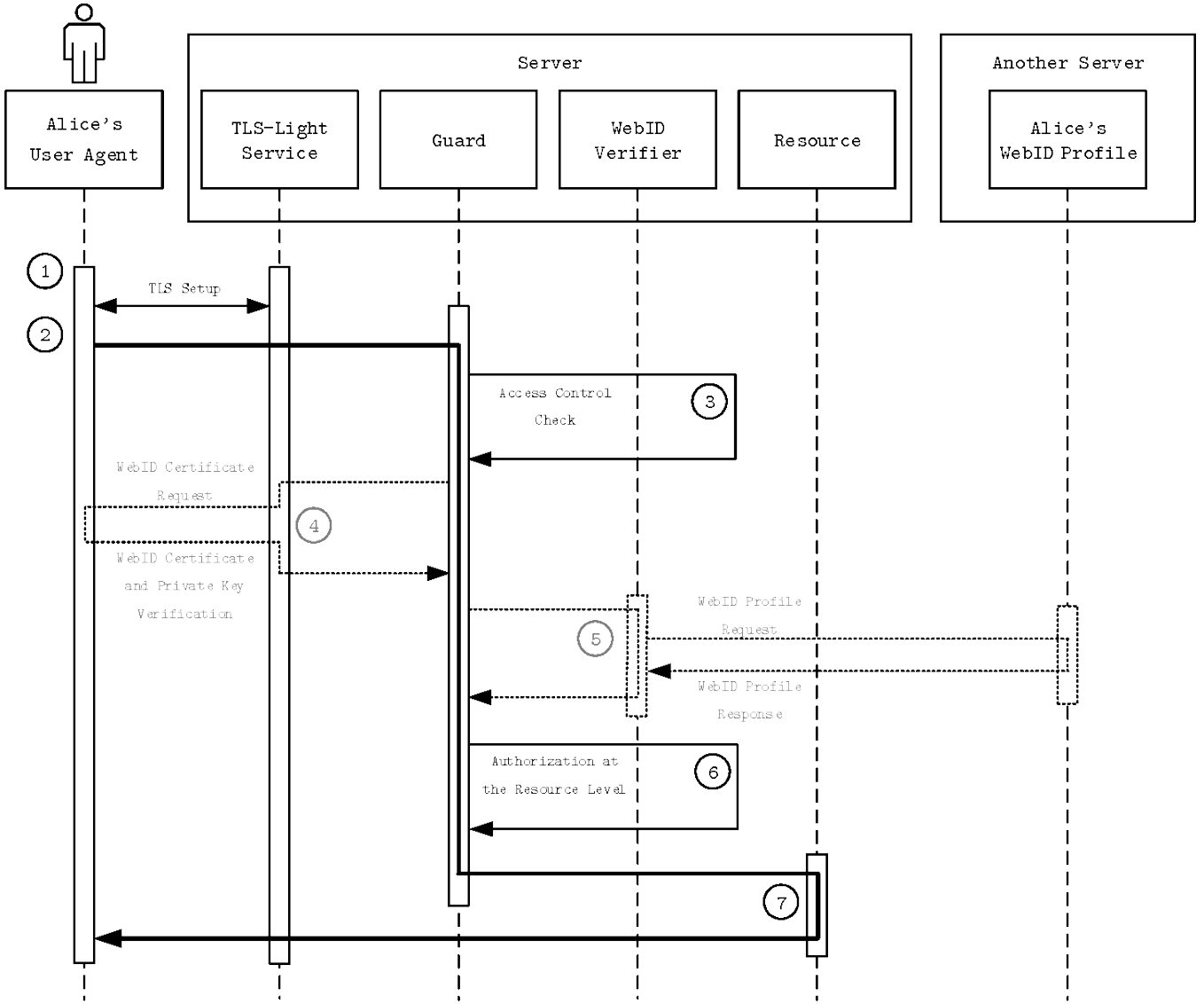
# WebID – Use of Profile Data

- Discovery as all personal data available as RDF triples
- SPARQL to query for information
- Linking to other concepts and related data instead of creating duplicates
  - Establishing social connections via foaf:knows entries
- WebID profile to store individual preferences
  - Can be used for tailored customizations and, thus, user experience improvements

# WebID-TLS – WebID Authentication over TLS

- Protocol enables secure and user-friendly authentication
  - Public key cryptography
  - No need to remember complex passwords or choose redundant, insecure ones → client certificate
- Makes use of
  - HTTPS/TLS
  - Client Certificate (WebID Certificate)
  - SPARQL

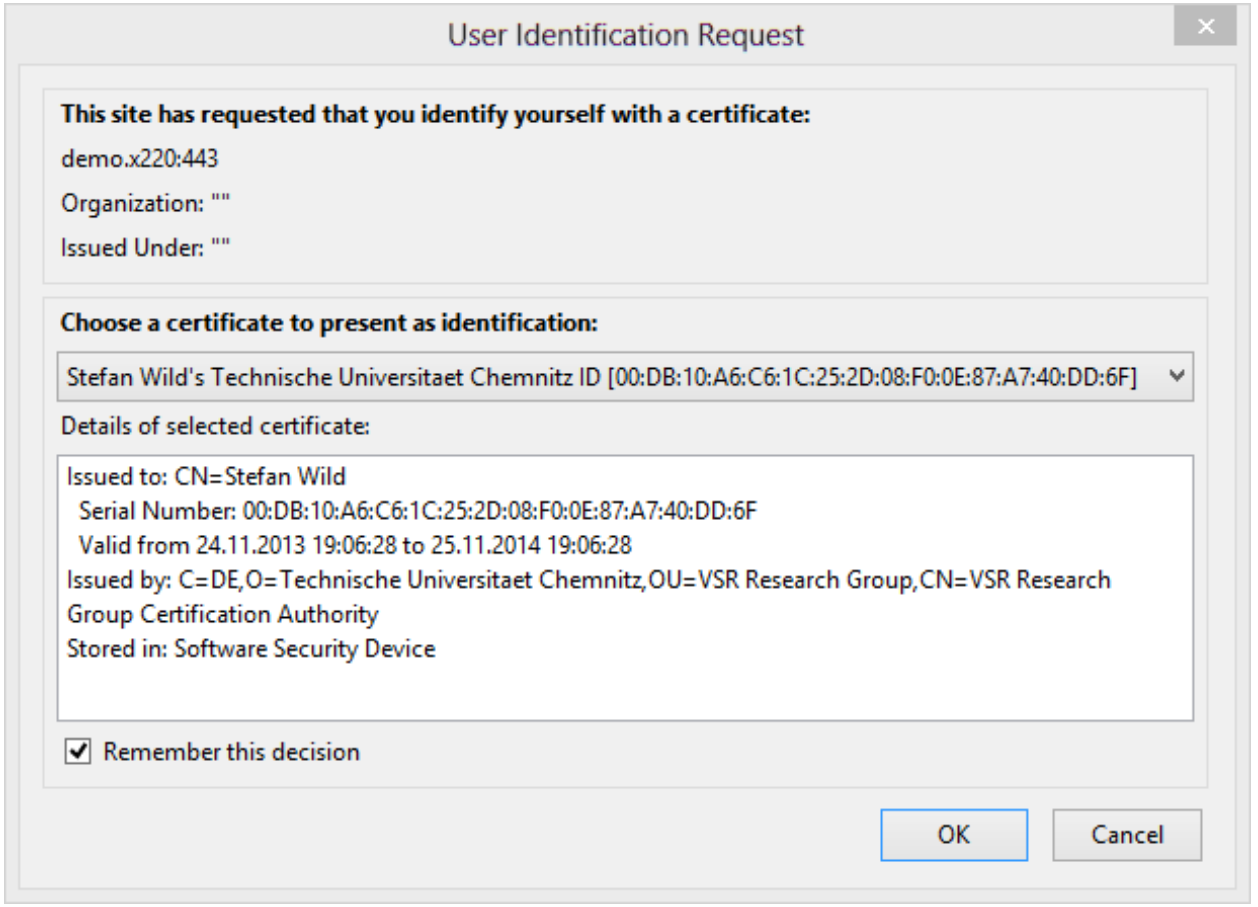
# WebID-TLS – Authentication Sequence



# WebID-TLS – TLS Protocol

1. Client sends preferences
2. Server sends preferences, server certificate, and ***client certificate request***
3. Client authenticates server and sends enc. pre-master secret and ***client certificate***, and signed challenge data
4. Server authenticates client and decrypts pre-master secret
5. Client & server generate master secret from pre-master secret, create session key and inform each other

# WebID-TLS – Certificate Selection Dialog



# WebID-TLS – SPARQL-Query for Verification

```
PREFIX : <http://www.w3.org/ns/auth/cert#>
PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>
ASK {
  ?webid :key [
    :modulus ?mod;
    :exponent ?exp;
  ] .
}
```

```
PREFIX : <http://www.w3.org/ns/auth/cert#>
PREFIX xsd:
<http://www.w3.org/2001/XMLSchema#>
ASK {
  <https://bob.example/profile#me> :key [
    :modulus "cb24...91a1"^^xsd:hexBinary;
    :exponent 65537;
  ] .
}
```

# Web Access Control (WAC)

- Decentralized access control system
  - Agent and resource are separated from each other
- All involved entities are identified by URIs
  - Agents (WebID URIs)
  - Resources
  - Access modes
- WAC vocabulary

  defines access rights in

  a machine-readable way
- http://www.w3.org/wiki/WebAcces

```
@prefix acl:
<http://www.w3.org/ns/auth/acl#>.

[
 a acl:Authorization;
 acl:accessTo <https://server.org/first>,
              <https://server.org/second>,
              <https://server.org/third>;
 acl:agent
<https://example.org/profiles/pan#pp>;
 acl:mode acl:Append, acl:Control, acl:Write
]
```

# WAC to Protect WebID Profiles

- WAC only allows resource-based access control
- All WebID profile data available as RDF triples
  - Relevant RDF triples to be protected can be outsourced
  - Separate resource containing set of RDF triples
  - Link to newly created resource from within WebID profile
  - Create appropriate WAC access control setting for this resource
- Fine-grained access control is hard to setup and to maintain

# WebID Identity Provider

- Users are enabled to be their own identity providers, but
  - Knowledge about WebID required
  - URI-addressable Web storage needed
    - Trust in Web storage provider
    - Protection of Web storage / system against attackers
  - Certificate generation not that easy for end users
    - WebID URI needs to be added(!)
  - WebID profile data creation / modification is difficult

# WebID Identity Provider (2)

- Third-party WebID identity provider
  - Simplified creation of WebID artifacts
  - No prior knowledge about WebID needed
  - But where to generate key pair and certificate?

# Challenges

- Browser integration needs to be improved
  - Logging out is complicated
  - Graphical representation of certificate selection dialog is not always ideal
- WebID certificate loss
  - How to claim that you are the identity owner

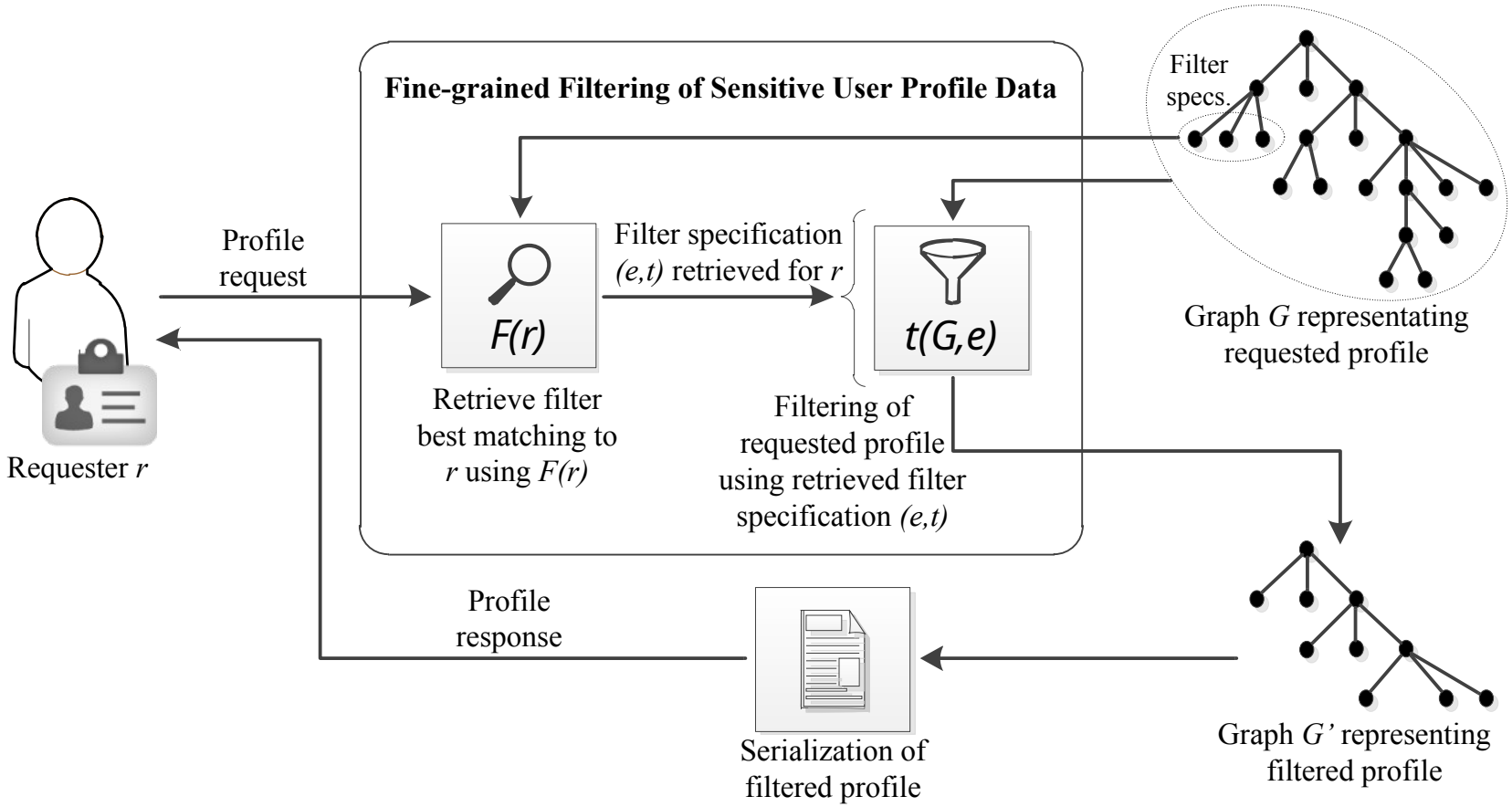
# WebID - Conclusion

- Users are in control of their identity data
  - Storing identity data at a self-defined place
- Machine-readable, flexible, extensible user profiles
  - Simplified discovery, identification, and query
  - Exploitation of user profile data
  - Management, e.g., social connections via foaf:knows
- Ownership-based authentication using certificates
- Reuse of existing infrastructure / technologies
  - HTTPS, TLS, X.509 client certificates
  - RDF, FOAF
  - SPARQL

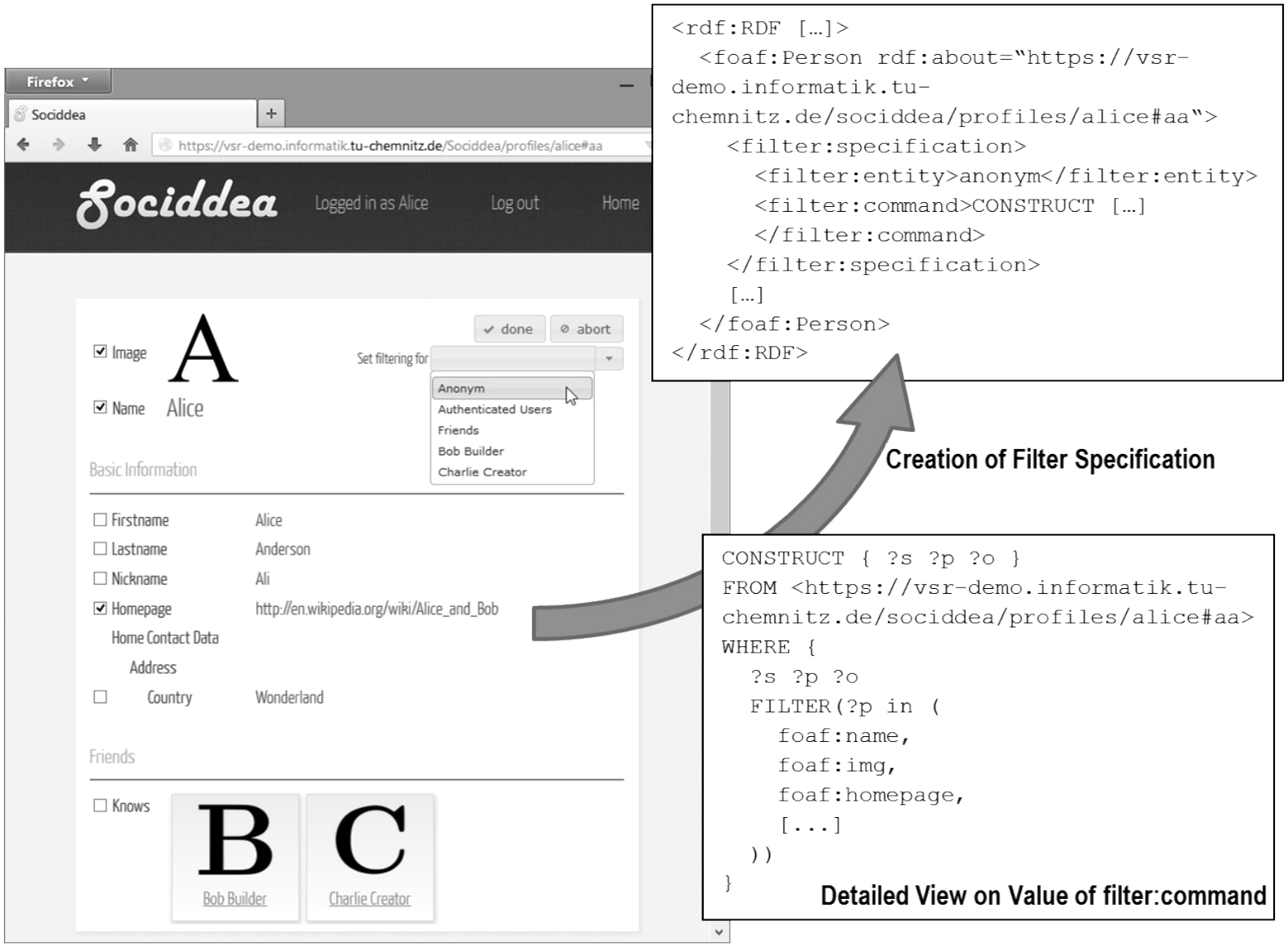
# *Current Research Questions*

- Customized Views on WebID Profiles
- Access Delegation in the Context of WebID
- Integrity Protection of WebID Profiles
- Context-aware WebID Certificate Creation
- Use of WebID for Semantic Web Services
- Identity Bridging Involving WebID
- Exploitation of WebID Profile Data

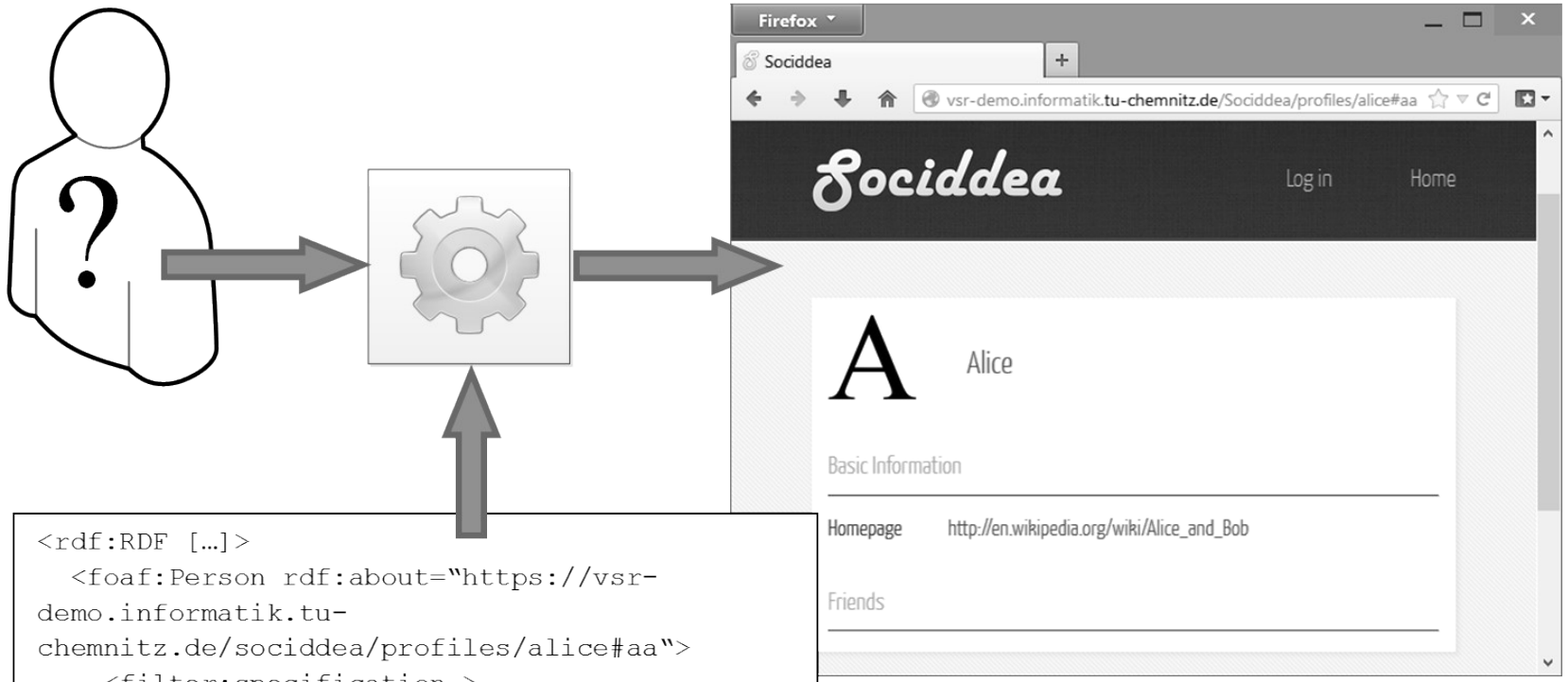
# *Research:* Views on WebID Profiles



# *Research:* Views on WebID Profiles

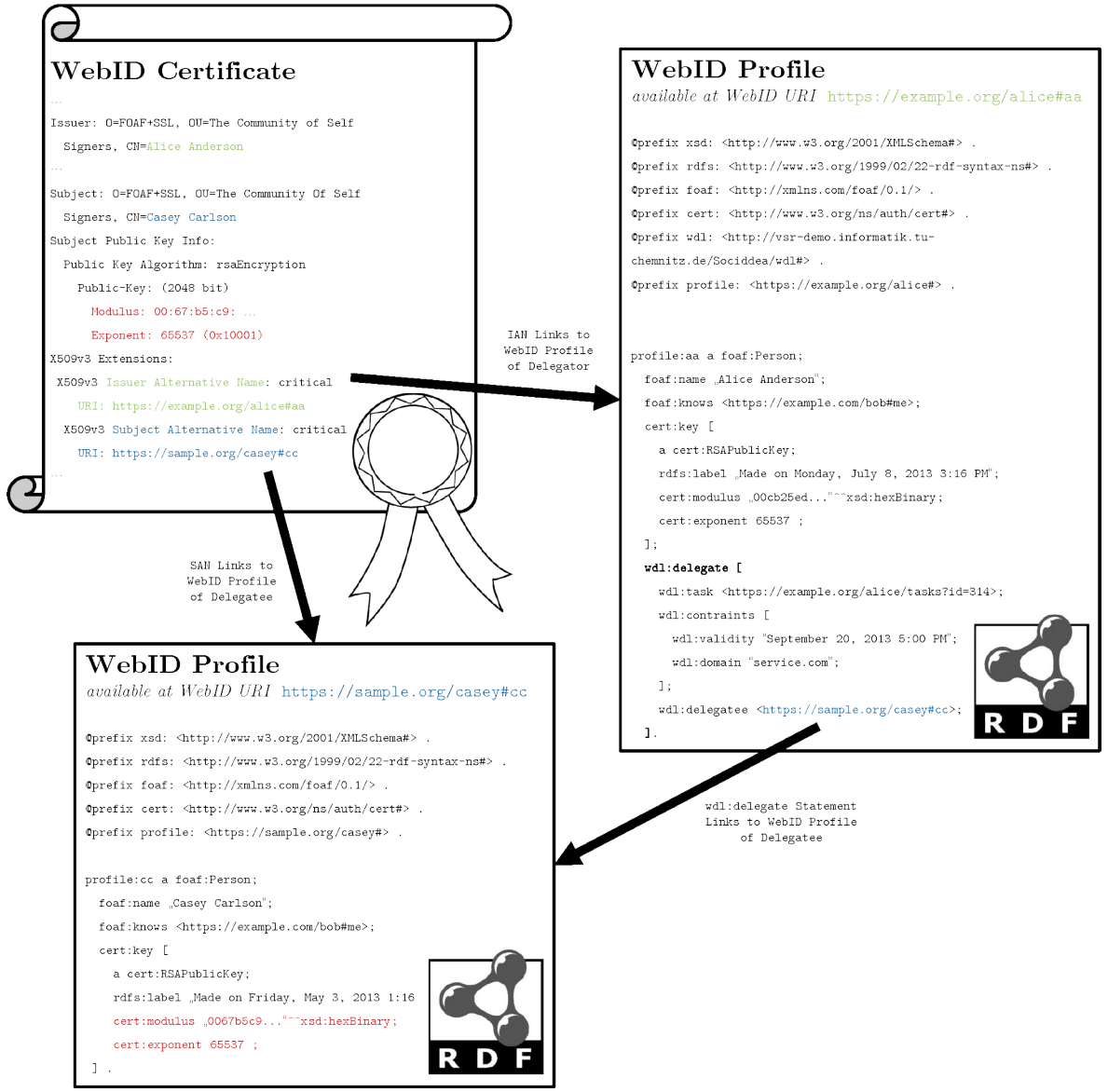


# *Research:* Views on WebID Profiles




```
<rdf:RDF […]>
  <foaf:Person rdf:about="https://vsr-
demo.informatik.tu-
chemnitz.de/sociddea/profiles/alice#aa">
    <filter:specification >
      <filter:entity>anonym</filter:entity>
      <filter:command>CONSTRUCT […]
      </filter:command>
    </filter:specification>
    […]
  </foaf:Person>
</rdf:RDF>
```

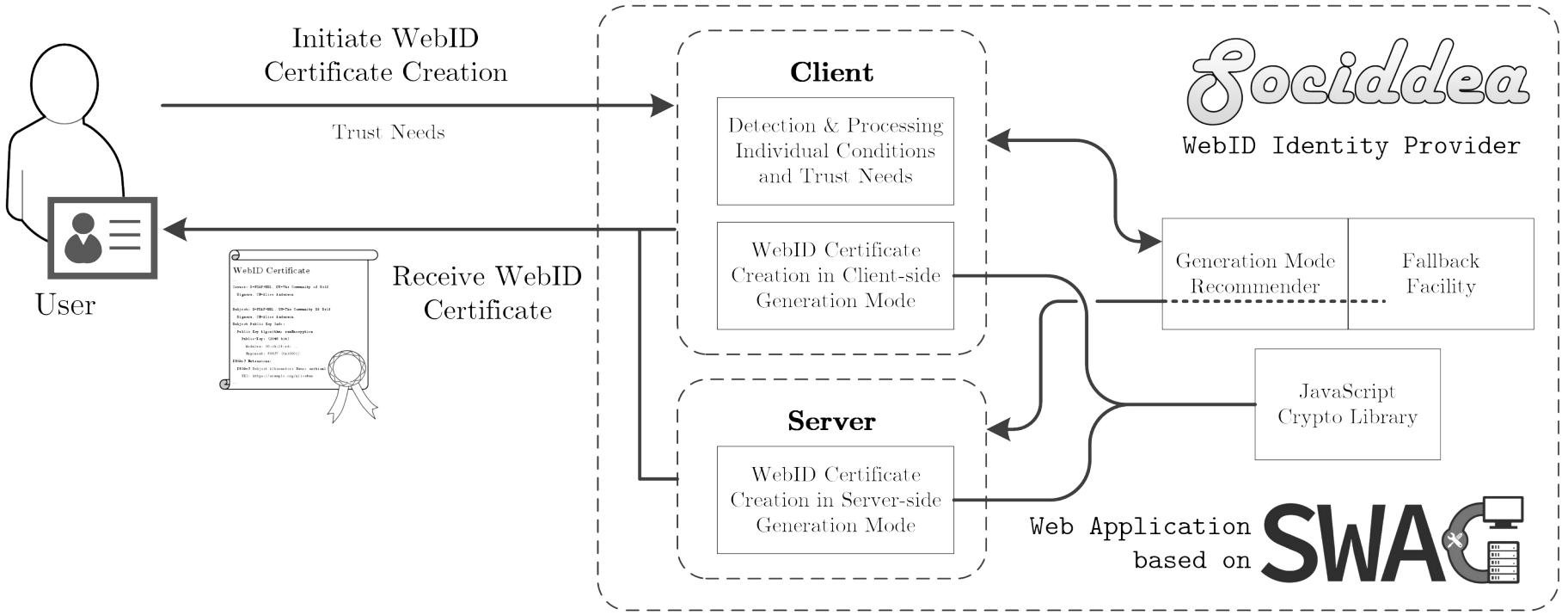
# *Research:* Access Delegation with WebID



# *Research:* Integrity Protection of Profiles

- Problem: WebID profile is not integrity-protected by default
  - Attackers could manipulate specific RDF triples, e.g., adding a wanted criminal as social connection
  - Attackers could steal identity
  - Persistent and transient attacks
- Goal: Enable users to host their WebID profile anywhere without having to fear data tampering or identity theft
  - Make tampering detectable during authentication
  - Allow profile requesters to verify integrity without prior knowledge

# *Research:* Context-aware Certificate Creation



# *Research:* WebID for Semantic Web Services

- Goal: Apply the principle of WebID profiles as repositories for personal information to Web services
  - Semantic interface description (query, discovery etc.)
  - Use foaf:knows to model connections between services
  - Utilize WAC to control access in a highly distributed environment

# *Research:* Identity Bridging Involving WebID

- Problem: While there are diverse digital identity systems in use, users want to authenticate only with their WebID certificates
  - How to bridge between different digital identity systems?
  - How to map personal information inside WebID profile?
  - How to store proof of identity in a secure way?
  - What to do when no adequate identity is available on the “other side of the bridge”?

# *Research:* Exploitation of WebID Profile Data

- Goal: Make use of decentralized social online network

- Areas:
  - Secure message exchange
  - Gathering profile data from social online networks
  - Synchronization of profile data
  - Skill matching / task assignment

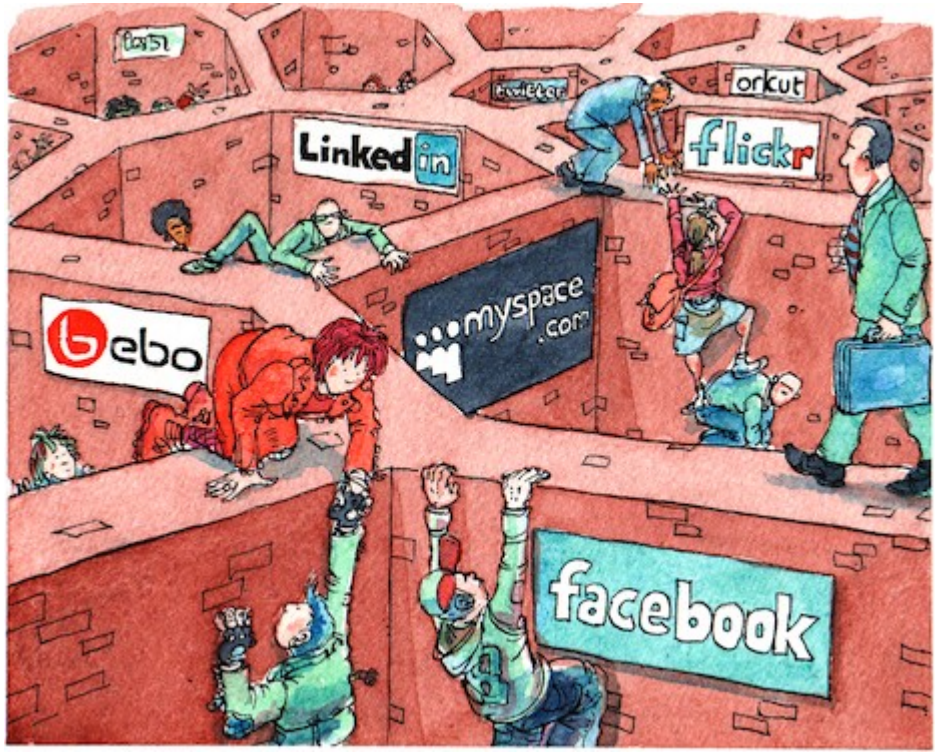
Chapter://2

# Social Web Protocols

# Challenges

- How to represent actors in the Social Web?
- How to share data between applications?
- How to avoid identity theft?
- How to keep data up-to-date?
- How to guarantee privacy?
- …

# The Problem of Walled Gardens




Source: Social Networking Sites as Walled Gardens by David Simonds

# The Problem of Walled Gardens

- **Portability:**
  - The user's data still remain in the control of the user
  - Let users share their information stored in social networks with other useful applications
- **Identity**
  - Problem of reusing passwords
  - Using new sites requires the users to re-find their friends

# The Problem of Walled Gardens

- **Linkability**
  - Inform users about content they are tagged in without being a member of the social network where the content is shared
- **Privacy**
  - Control how the users‘ information is viewed by others in different contexts
  - Preventing or undoing of data discolsures by others about oneself

# Terminology


User


Identity
(1 representation of User)


Profile Attribtue


Social Connection


Social Group


Social Interaction

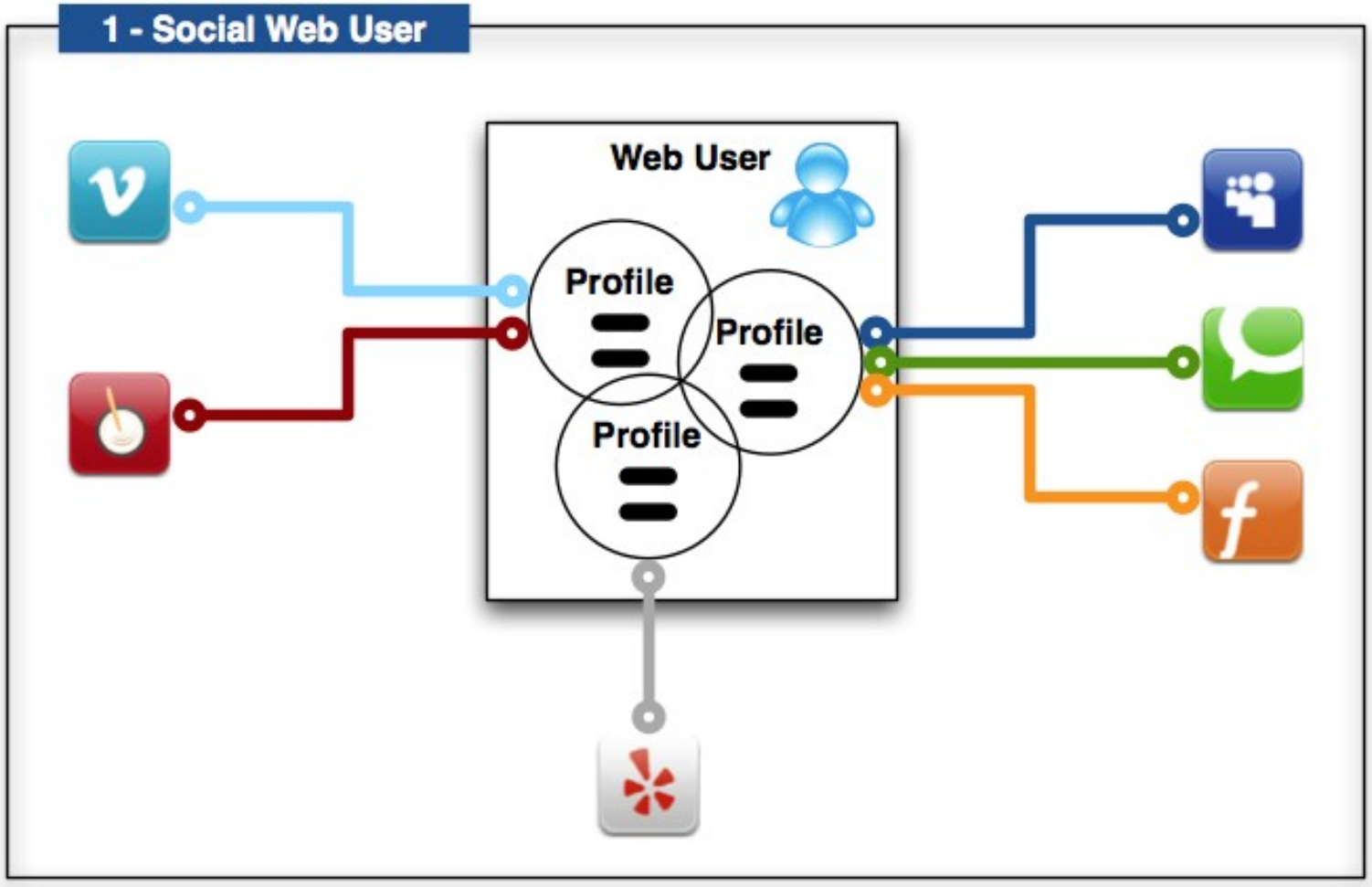

Social Platforms


Distributed Social Graph


Social Applications

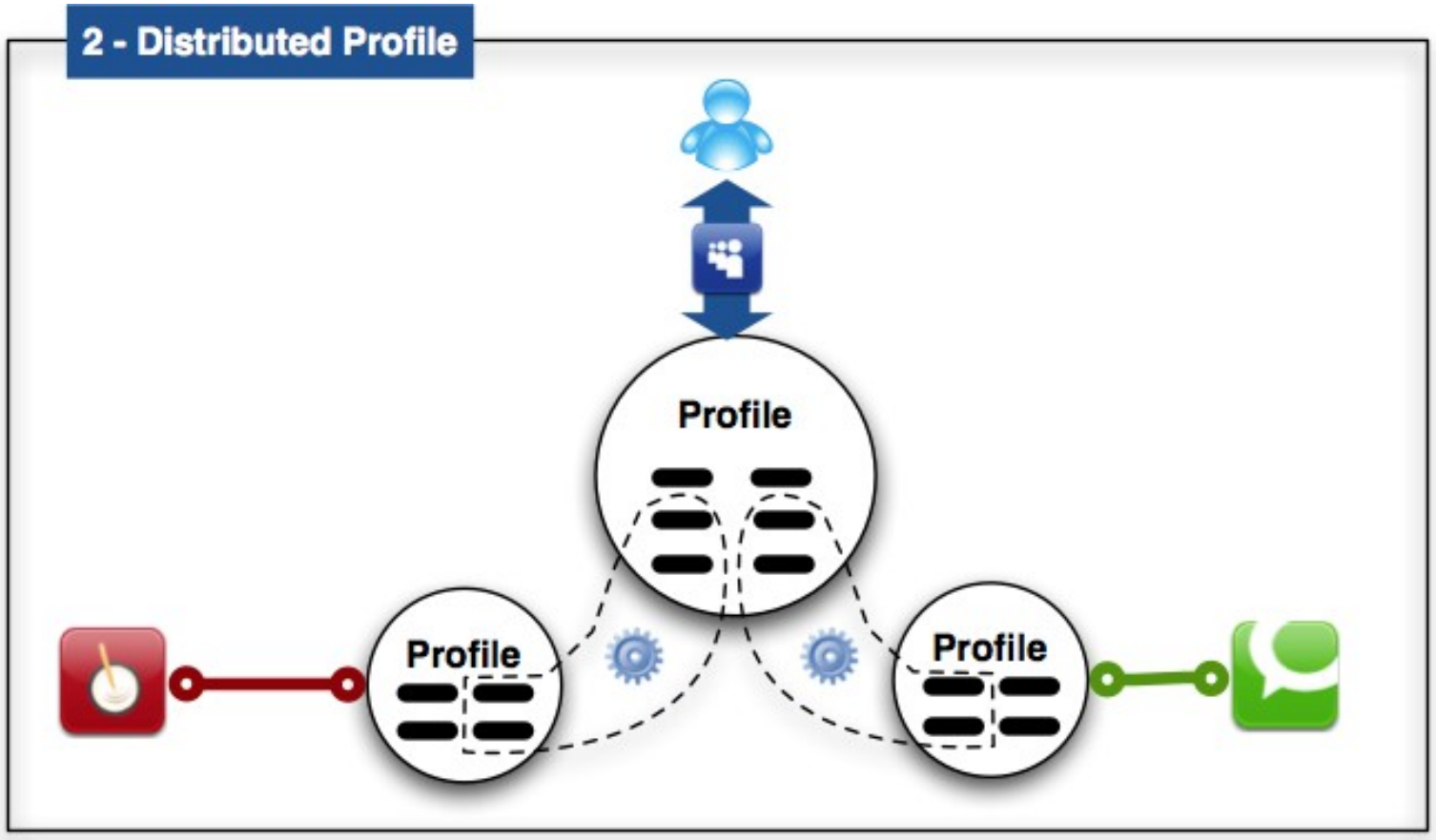

Profile Association

# Social Web User and Profiles



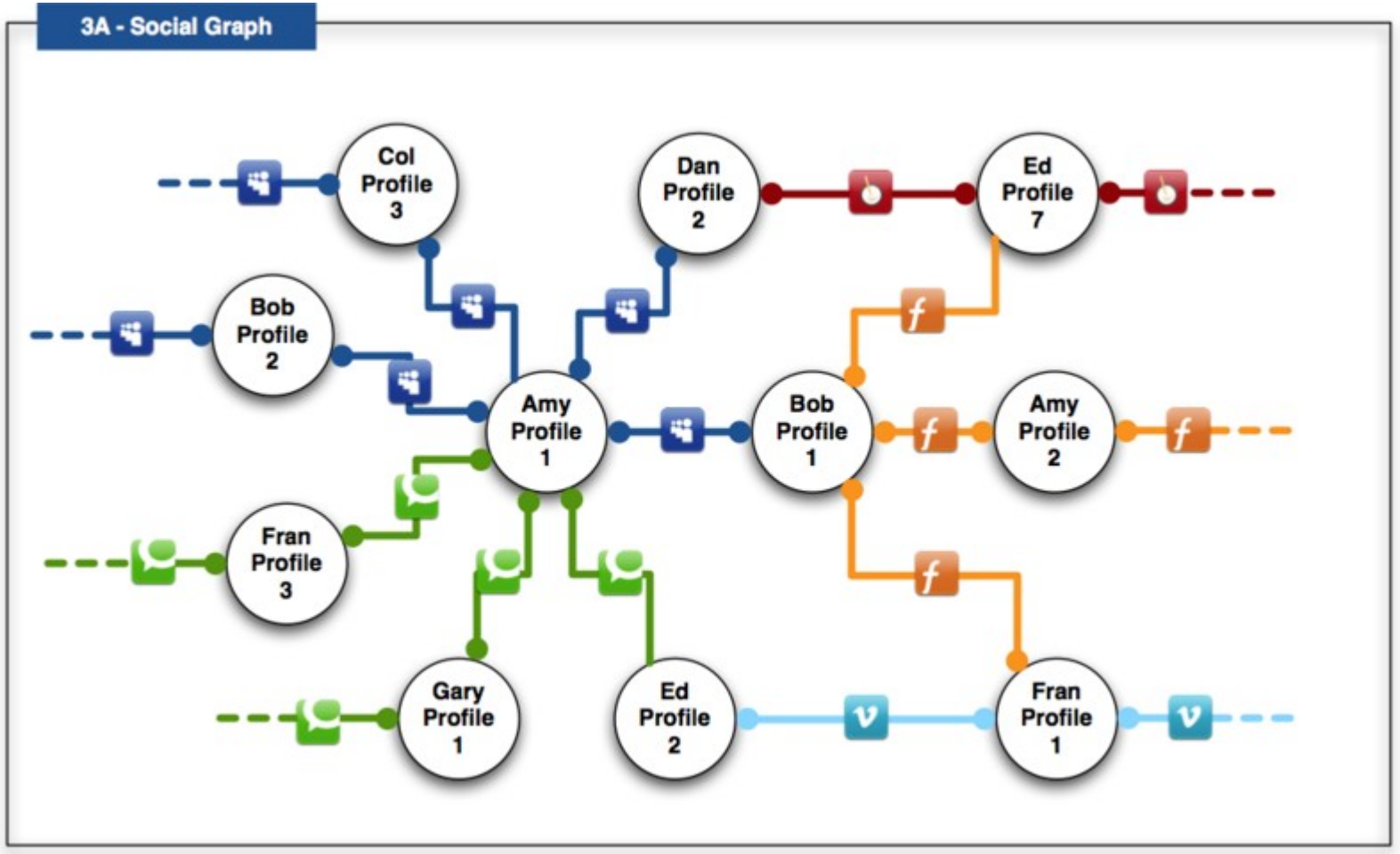

Source: http://www.w3.org/2005/Incubator/socialweb/wiki/FinalReport

# Single Distributed Social Graph



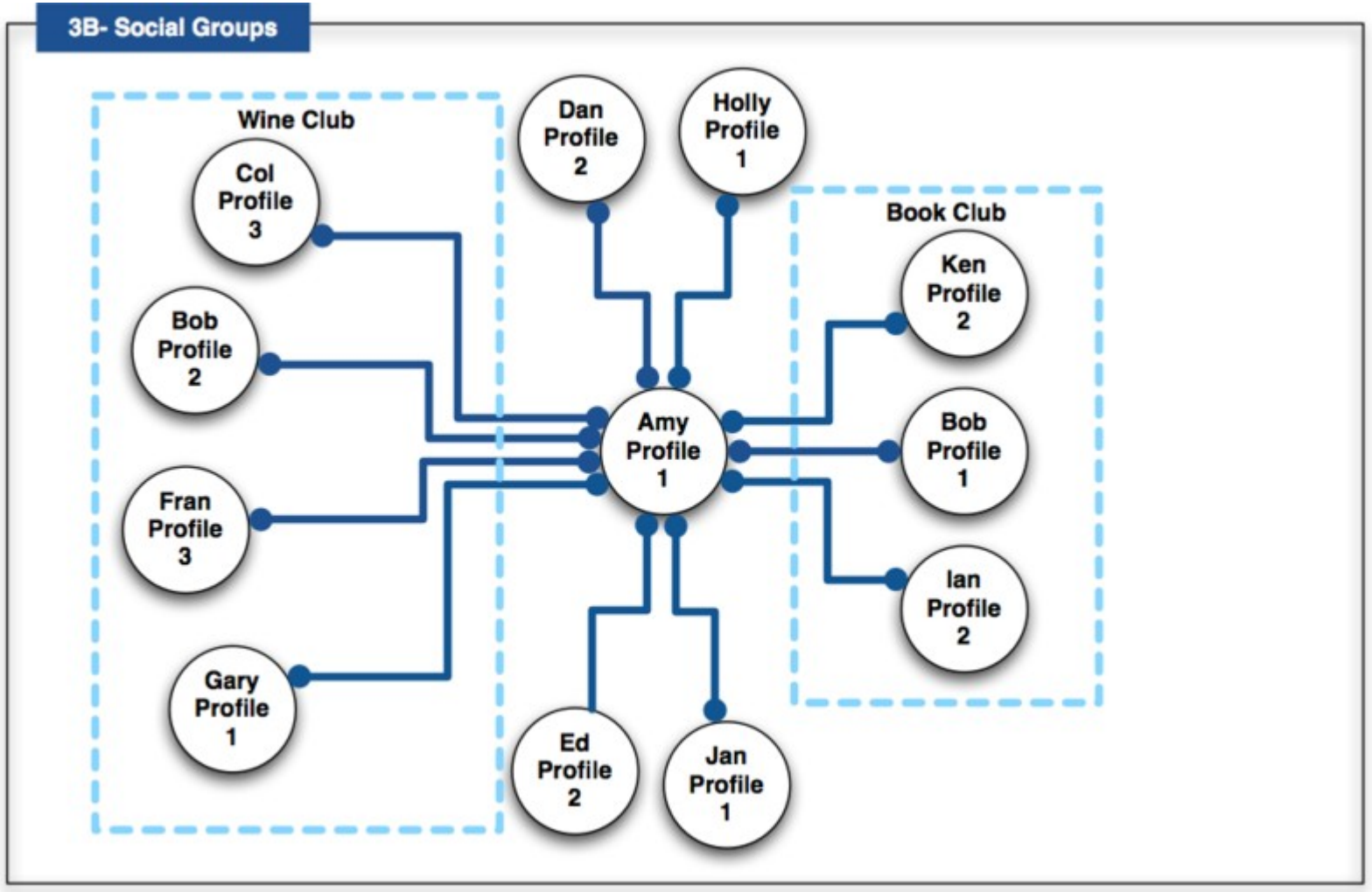

Source: http://www.w3.org/2005/Incubator/socialweb/wiki/FinalReport

# Multiple Distributed Social Graphs



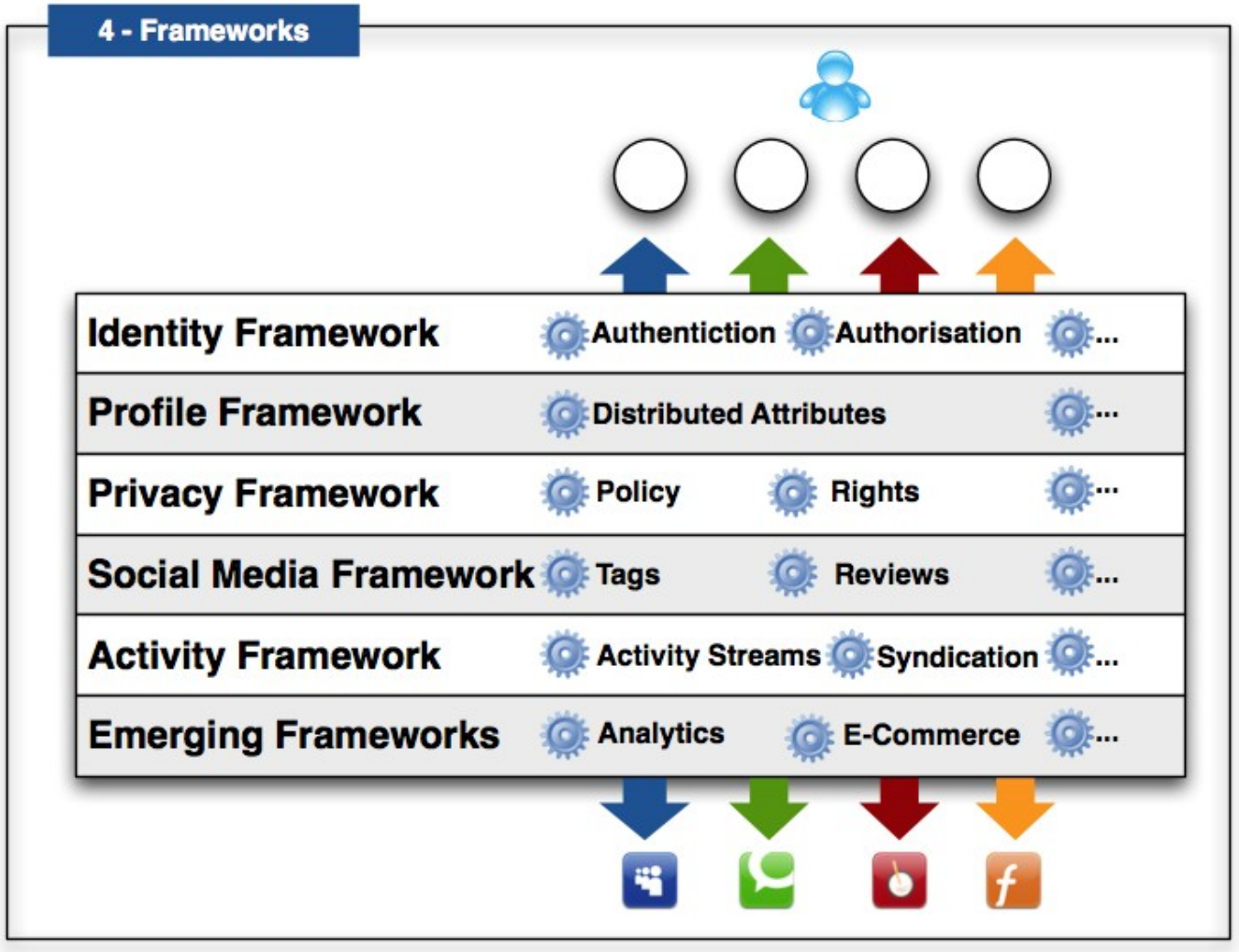

Source: http://www.w3.org/2005/Incubator/socialweb/wiki/FinalReportHenryArt

# Multiple Distributed Social Graphs




Source: http://www.w3.org/2005/Incubator/socialweb/wiki/FinalReport

# Multiple Distributed Social Graphs




Source: http://www.w3.org/2005/Incubator/socialweb/wiki/FinalReport

# Identity

- Problem: Usernames and Passwords are Insecure
- Use-case: No more passwords (or only one)
- Discussed earlier in this lecture:
  - Cf. OpenID, Oauth, WebID, etc.

# Profiles

- Problem: Can Not Describe Yourself
  - User profiles often constrained how to describe users
  - Manually re-find friends
  - Different names on different sites and anonymity on other sites
- Use-case: Keep Your Profile and Friends Across Networks
  - Authentication using Single-Sign-On at a new site
  - Re-using an existing profile by adding a few new fields in the profile
  - Automatic discovery of friends
  - Complete download of the profile for further use

# Profile Standards

- XRD (Extensible Resource Descriptor)
  - The XRD format provides this for arbitrary resources via the use of types and typed links describing URIs (URI templates) given in the XML format that can then be queried by a user-agent.
- VCard (cf. XML-Lecture)
- FOAF (cf. prev. session)

# Profile Standards

- PortableContacts
  - It provides a common access pattern and contact scheme as well as authentication and authorization requirements for access to private contact information.
- OpenSocial
  - collection of Javascript APIs, controlled by the OpenSocial Foundation, that allow Google Gadgets (a proprietary portable Javascript application) to access profile data, as well as other necessary tasks such as persistence and data exchange

# Social Media

- Problem: Fined for Consuming Social Media
  - How to (re-)use social media without breaking the content's copyright
  - User want to be well-informed about the social media they consume
- Use-case: Safely Drag-and-Drop Social Media Across Multiple Platforms
  - Easily adding Creative Commons license to social media (e.g. Micropayments for commercial used pictures)
  - Removing content from the original sites as well as the sites which are resharing the content

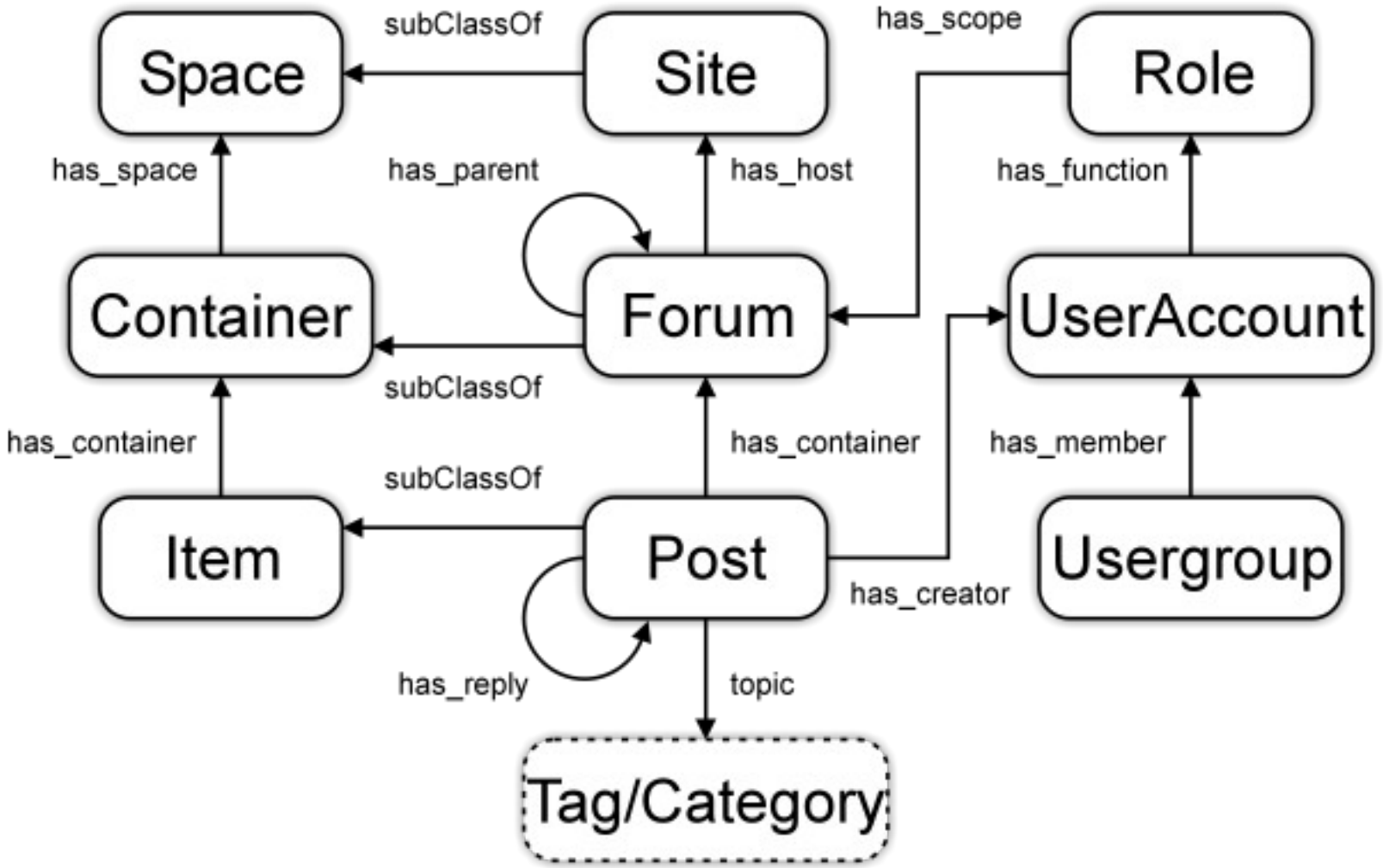
# Social Media Standards

- Tagging
- Microformats (cf. XML-Lecture)
- Open Graph Protocol
  - is a metadata vocabulary for describing documents and (indirectly) their topics
- Payswarm
  - Micropayment for social media

# Social Media Standards

- OExchange
  - is a OWF-licensed specification for users sharing rich content over the Web using URIs between social sites
- The Semantic Web (cf. XML-Lecture)
- SIOC
  - aims at developing a standard vocabulary for representing user-generated content on the Web, using Semantic Web technologies

# SIOC




Source: http://sioc-project.org/ontology

# SIOC

- Example at http://sioc-project.org/ontology#sec-example

# Privacy

- Problem: Violation of Privacy
  - Social media are spread across multiple social networking sites and accessed by all sorts of people
- Use-case: Your Own Terms of Service
  - Defining what to share with whom
  - Alert conflicts between own defined sharing propertys and these which are defined by the service (e.g. Alice share a photo under Creative Commons with attribution, but sharing something on some service means it becomes property of this service)

# Privacy Standards

- P3P
  - Platform for Privacy Preferences
  - allows website operators to express their data collection, use, sharing, and retention practices in a machine-readable format
- POWDER
  - Protocol for Web Description Resources
- AIR
  - AMORD in RDF
  - a policy language and features a basic proof-level to define policies in a machine-readable manner

# Privacy Standards

- XACML
  - eXtensible Access Control Markup Language
  - Declarative policy language for access control
- Rule Interchange Format (RIF)
  - The W3C RIF Recommendation is a format to exchange rules between rule engines that operates over both XML and RDF data
- Mozilla Privacy Icons
  - takes a simple icon-based approach inspired by Creative Commons

# Mozilla Privacy Icons

- Retention period
  - User’s data will be deleted before X months


- Third-party use
  - Intended Use Only

  - Limited Re-use

# Mozilla Privacy Icons

- Ad networks
  - No ad-share
  - Ad-share with opt-out
- Law enforcement
  - Statutory process
  - Transparent process

# Activity

- Problem: Can not Integrate Conversations
  - Social media can circulated across multiple social networking sites
  - No standard way to update and integrate back distributed comments back to their original source
- Use-Case: Real-time Collaboration
  - Update of friends‘ activities across multiple socoial networking sotes

# Activity Standards

- XMPP
  - Extensible Messaging and Presence Protocol
  - is an IETF RFC for the near real-time transfer of XML data
- Atom and Pubsubhubbub
  - Pubsubhubbub provides a „push“ architecture for the HTTP-based Web
  - Atom addresses syndication of Web content using XML

# Activity Standards

- ActivityStreams
  - Atom and JSON serialization for activity streams such as status updates in popular seocial networking sites
  - Defines a vocabulary for activities
- Salmon Protocol
  - While Pubsubhubbub deals with publishing content to multiple subscribers, Salmon defines a protocol how to provide interactions „upstream“ to the original content
- OStatus
  - is a a "meta-specification" for sending status updates to people in a federated Social Web

# ActivityStreams

- Actor *verb* Object [ Target ]

- Activity Base Schema
  - Defines vocabulary for Activity Streams
- JSON Activity Streams
  - Defines serialization of Activity Streams in JSON
- Atom Activity Streams
  - Defines serialization of Activity Streams in Atom

# ActivityStreams

```
{
 "published": "2011-02-10T15:04:55Z",
 "actor": {
  "objectType" : "person",
  "id": "tag:example.org,2011:jane"
 },
 "verb": "share",
 "object" : {
  "objectType":"activity",
  "title": "John posted a photo",
  "id": "tag:example.org,2011:abc123",
  "verb": "post",
  "actor": {
   "objectType":"person",
   "id":"tag:example.org,2011:john"
  },
  "object": {
   "objectType":"photo",
   "url":"http://example.org/album/my_fluffy_cat.jpg"
  }
 }
}
```

# Salmon Protocol