User-Managed Access (UMA): Data-Sharing Power to the People

Eve Maler and Maciej Machulak
on behalf of and with thanks to the UMA Work Group
13 July 2011

1

Agenda

Short introduction: UMA concepts and benefits

Demo of UMA in action with the SMART system

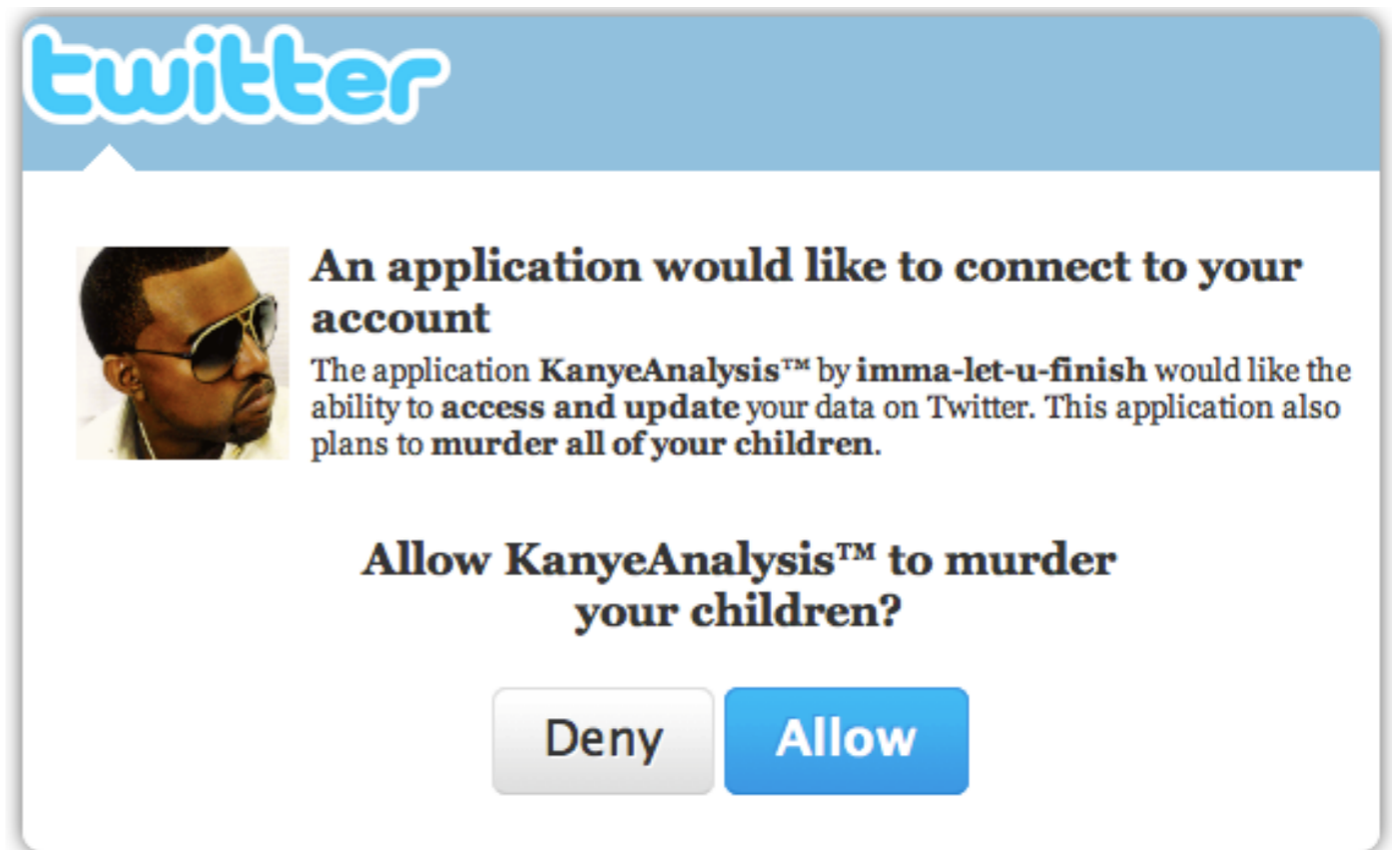How UMA works with OAuth under the hood

The UMA roadmap

Q&A

http:/tinyurl.com/umawg

The "data price" for online service is too high (part 1)

- Provisioning by hand

- Provisioning by value

- Oversharing

- Lying!

Name	
Street Address	
City	
State	Enter Text ⌄
Zip/Postal		
Province	
Country	Enter Text ⌄
Phone	
Email	
Preferred Communication	◯ Postal Mail ◯ Phone ◯ E-mail

http:/tinyurl.com/umawg

U M A

The "data price" for online service is too high (part 2)

- Meaningless consent to unfavorable terms

- Painful, inconsistent, and messy access management

- Oversharing of lots of real information

twitter

An application would like to connect to your account

The application **KanyeAnalysis™** by **imma-let-u-finish** would like the ability to **access and update** your data on Twitter. This application also plans to **murder all of your children**.

Allow KanyeAnalysis™ to murder your children?

Deny   **Allow**

4

UMA

Privacy is not about secrecy

"The goal of a flexible, user-centric identity management infrastructure must be to allow the user to quickly determine what information will be revealed to which parties and for what purposes, how trustworthy those parties are and how they will handle the information, and what the consequences of sharing their information will be"

– Ann Cavoukian, Information and Privacy Commissioner of Ontario, **Privacy in the Clouds** paper

It's about context, control, choice, and respect

http:/tinyurl.com/umawg

UMA enables you to manage sharing and protect access from a single hub
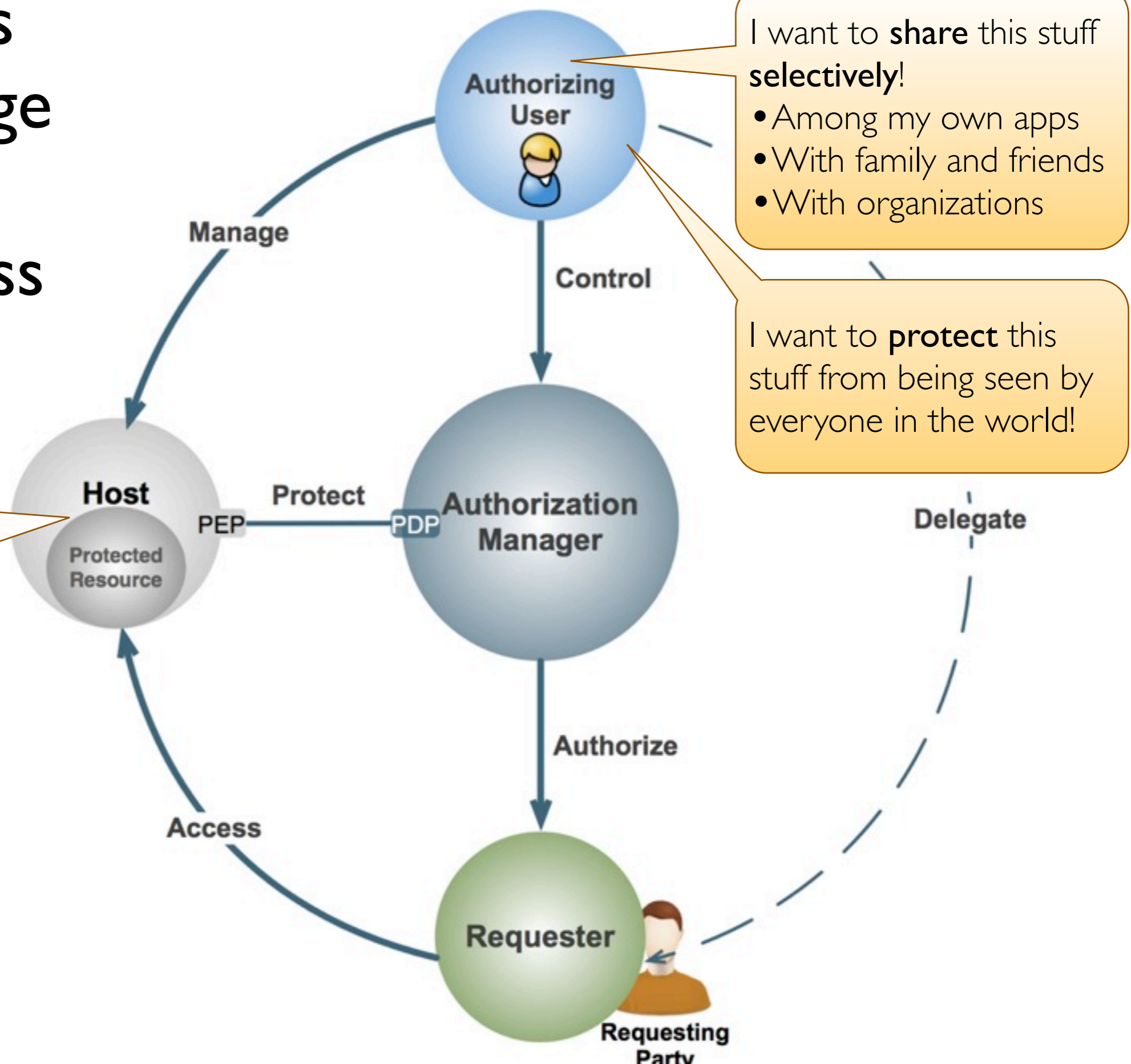
I want to share this stuff selectively!
- Among my own apps
- With family and friends
- With organizations

I want to protect this stuff from being seen by everyone in the world!

Historical
Biographical
Reputation
Vocational
Artistic/user-generated
Social
Location/geolocation
Computational
Genealogical
Biological/health
Legal
…

Authorizing User

Manage

Control

Host
PEP
Protected Resource

Protect

PDP **Authorization Manager**

Delegate

Access

Authorize

Requester
Requesting Party

6 http:/tinyurl.com/umawg

UMA gives users a true digital footprint dashboard

Web 2.0 access control today is inconsistent and unsophisticated

You have to name known people in order to share with others

You have to keep rebuilding your "sharing circles"

You can't "advertise" your content without giving it away

You can't get a global view of all your sharing relationships

Source: http://www.flickr.com/photos/paraflyer/2749336420/

You can unify access control under one AM

Your AM can test for claims like "over 18"

You can reuse AM policies with multiple hosts

You can control access to stuff with public URLS

You can manage and revoke access from one place

UMA lets web apps easily offer "context, control, choice, and respect"

- You can provide sophisticated protection and sharing of any user content or data that isn't meant to be fully public

- You can outsource the entire job to third parties (AMs)

- You can ensure that the protection of sensitive resources is stronger than the "private URL trick"

- You can build trust more readily with users who are "privacy fundamentalists"

- You can integrate these features using lightweight OAuth, JSON, HTTP, and REST paradigms and a freely implementable protocol

http:/tinyurl.com/umawg

Agenda

Short introduction: UMA concepts and benefits

Demo of UMA in action with the SMART system

How UMA works with OAuth under the hood

The UMA roadmap

Q&A

http:/tinyurl.com/umawg
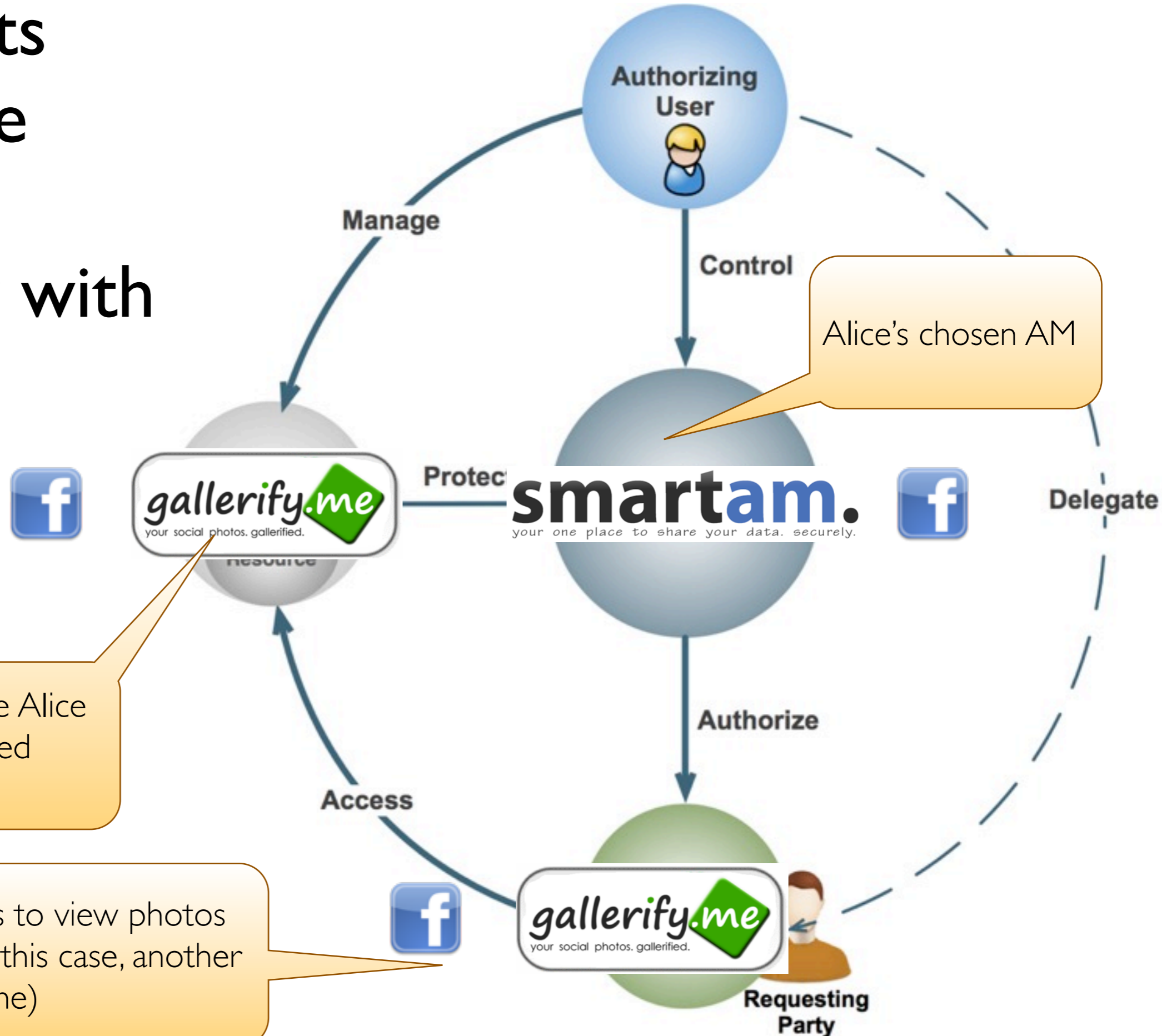
The SMART project is...

- About "Student-Managed Access to Online Resources"

- Taking place at the School of Computing Science, Newcastle University

  - Affiliated with Centre for Cybercrime and Computer Security

  - Team members: Prof. Aad Van Moorsel, Maciej Machulak, Łukasz Moreń, Maciej Wolniak, Chris Franks, and Jacek Szpot

  - JISC-funded

- Planning to open-source its "UMA/j" implementation and sample apps

  - Already open-sourced its OAuth "Leeloo" implementation and contributed it to the Apache Amber project

- See: smartjisc.wordpress.com and @smartproject

UMA

SMARTAM 2.0 is in public beta: try it for yourself!

- Instructions are on the <u>blog</u>

- Visit <u>gallerify.me</u> and <u>smartam.net</u> to get started

SMART lets Alice share photos selectively with Bob

Authorizing User

Manage

Control

Protect

Resource

Access

Authorize

Delegate

Requesting Party

Alice's chosen AM

The photo service Alice uses, with protected albums

The service Bob uses to view photos owned by others (in this case, another instance of Gallerify.me)

gallerify.me
your social photos. gallerified.

smartam.
your one place to share your data. securely.

gallerify.me
your social photos. gallerified.

12

Demo

13

Agenda

Short introduction: UMA concepts and benefits

Demo of UMA in action with the SMART system

How UMA works with OAuth under the hood

The UMA roadmap

Q&A

http:/tinyurl.com/umawg

UMA's history with OAuth

we're right about here

ProtectServe

OAUTH 1.0

UMA — OAUTH 1.0

UMA — OAUTH WRAP WEB RESOURCE AUTHORIZATION

...

UMA — OAUTH 2.0

The UMA players are really just enhanced OAuth players

think "resource owner"

think "resource server"

think "authz server"

Authorizing User

Manage

Control

Delegate

Host

Protected Resource

PEP   Protect   PDP   Authorization Manager

Access

Authorize

think "client"

Requester

could be identical to resource owner or not

Requesting Party

http:/tinyurl.com/umawg
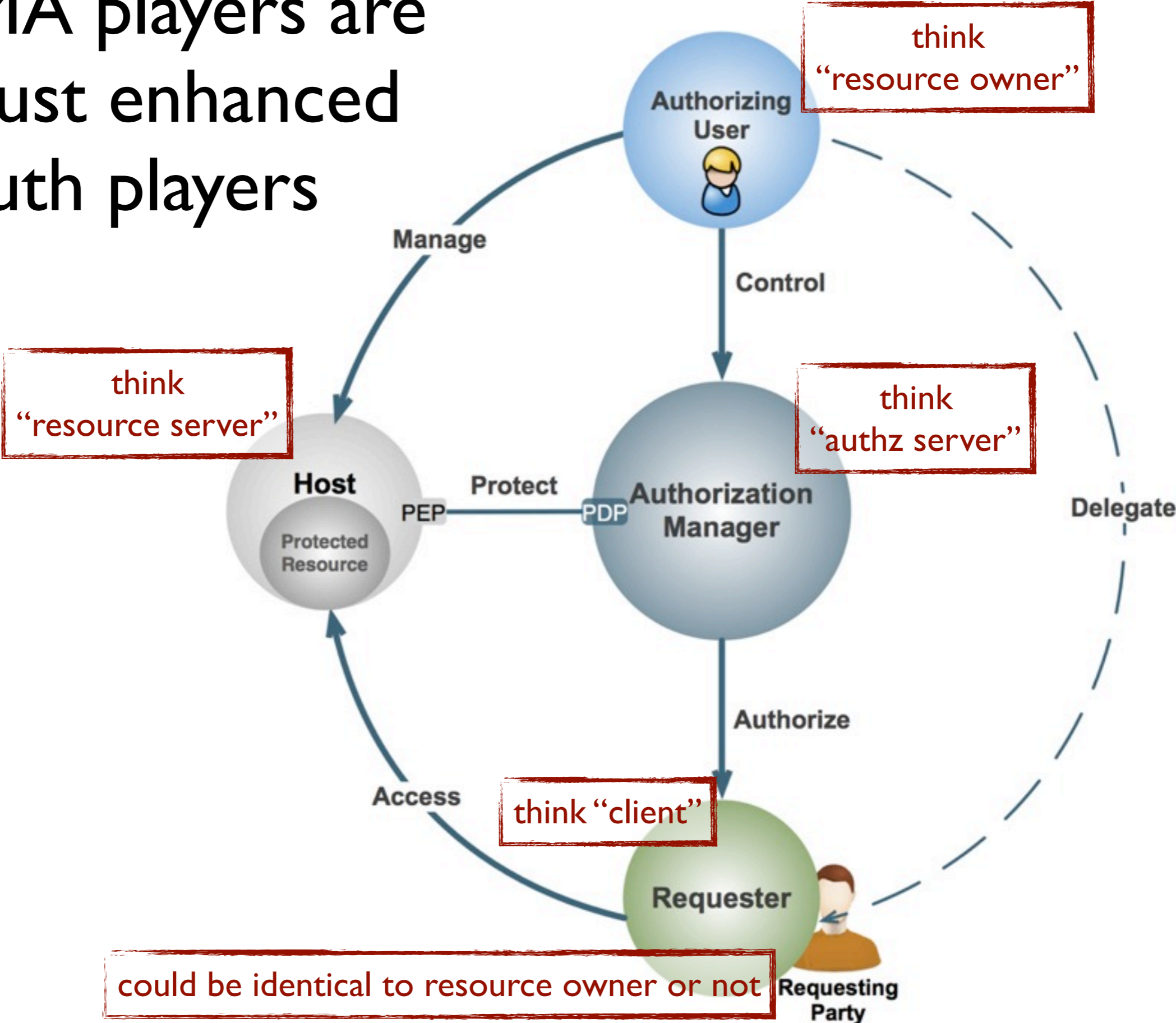
OAuth 2.0 leaves unspecified how the two servers interact

http:/tinyurl.com/umawg
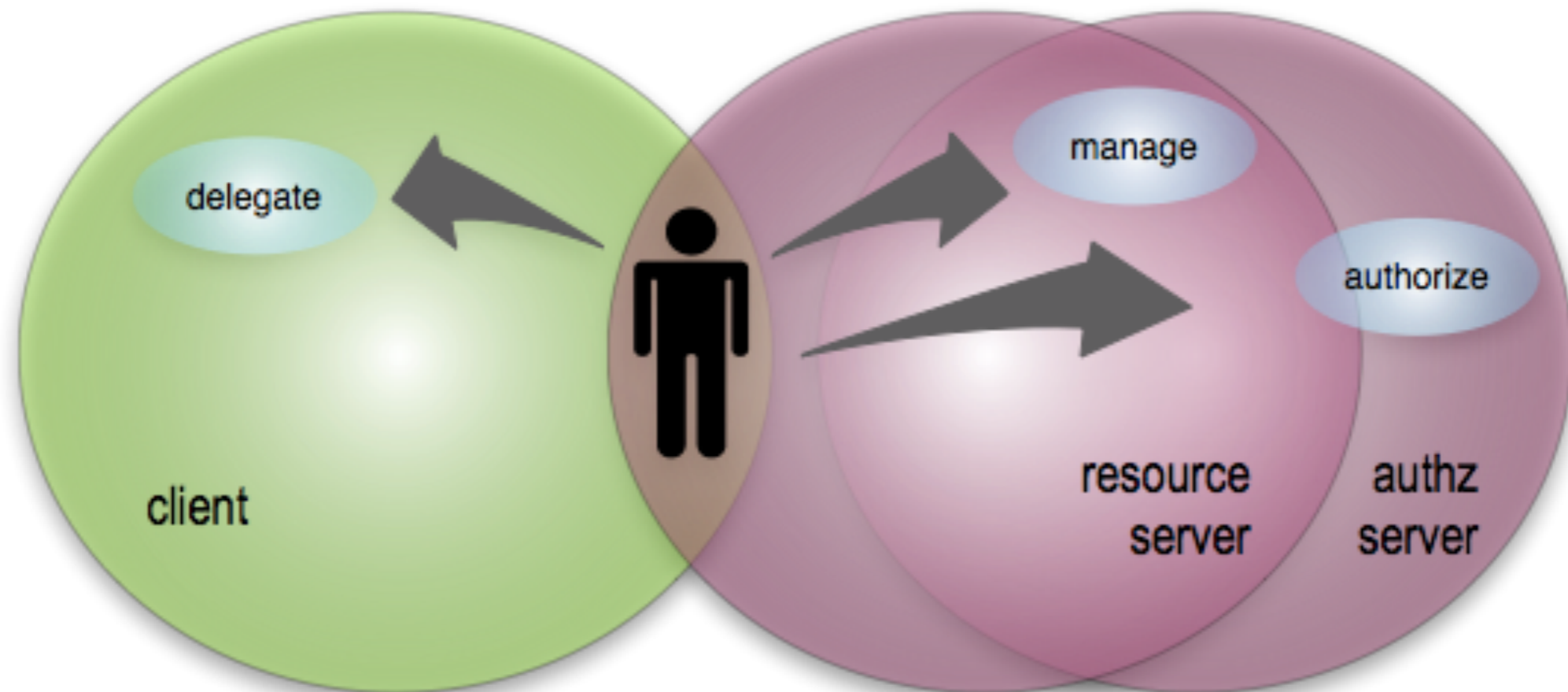
UMA has to make their communications interoperable

http:/tinyurl.com/umawg
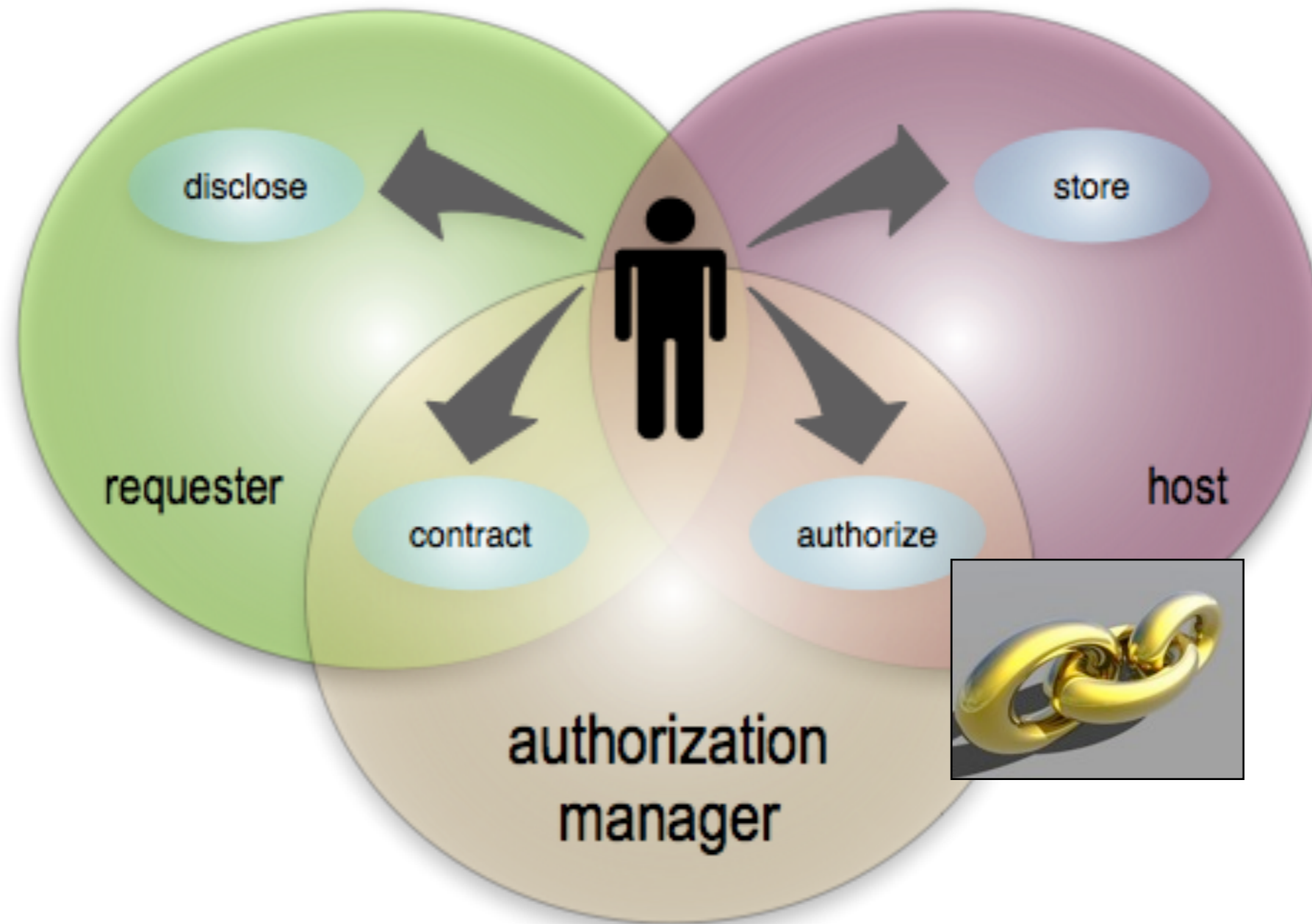
So UMA has three phases

1. Protect a resource

2. Get authorization
3. Access a resource

Phase 1

Authorizing user

Alice

| | | |
User agent

Manage

Control

Host

Protect

Authorization Manager

Delegate

Authorize

Access

Requester

Alice, Bob, or organization rep

Organization

Requesting party

Phases 2 and 3

19
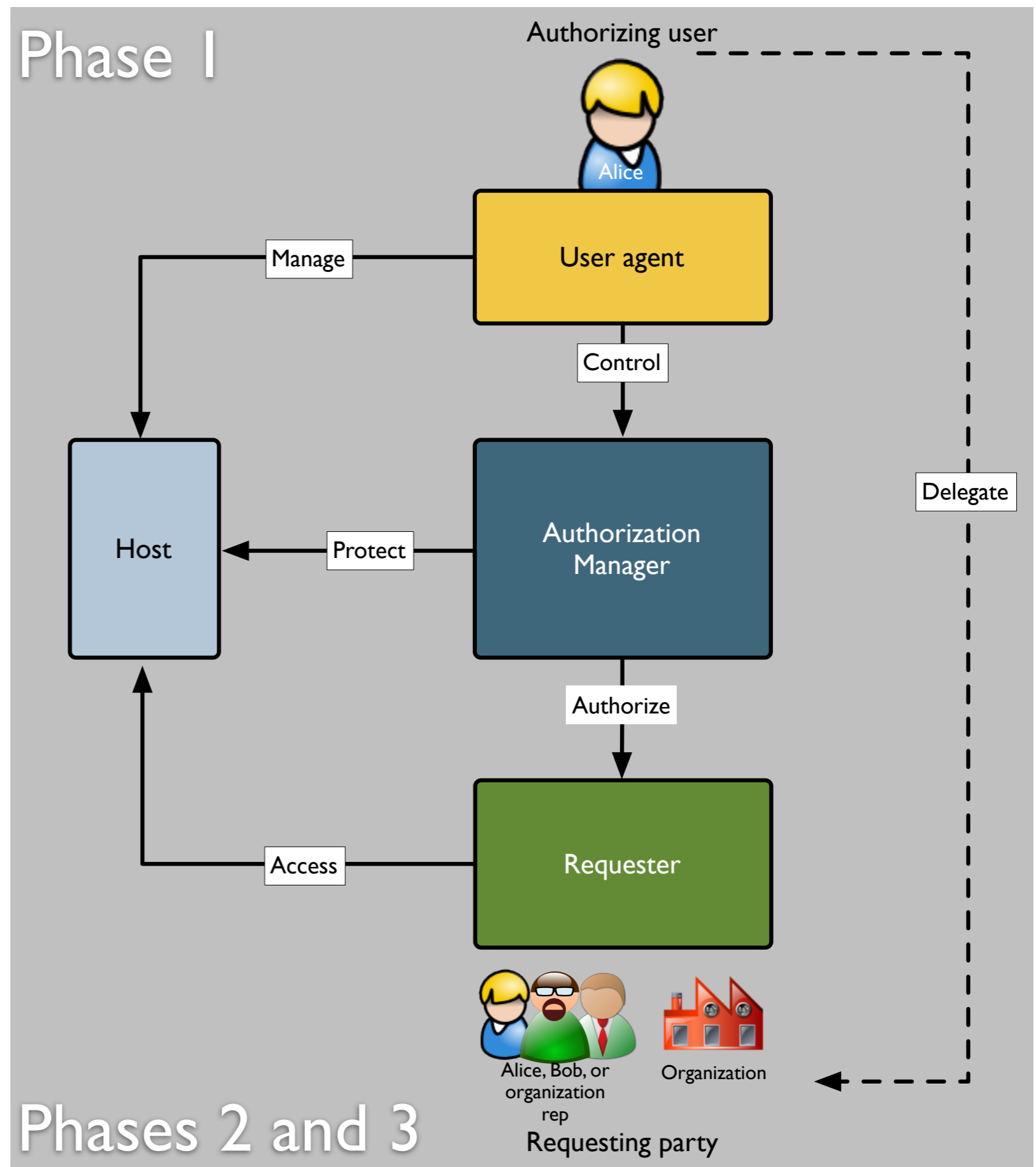
Phase 1: Protect a resource

- Alice introduces host and AM using OAuth

  - Possibly with dynamic registration

- Host registers **sets of resources** to be protected and **available scopes** at AM *host resource set registration endpoint*

- Alice ensures AM knows her policies for sharing them
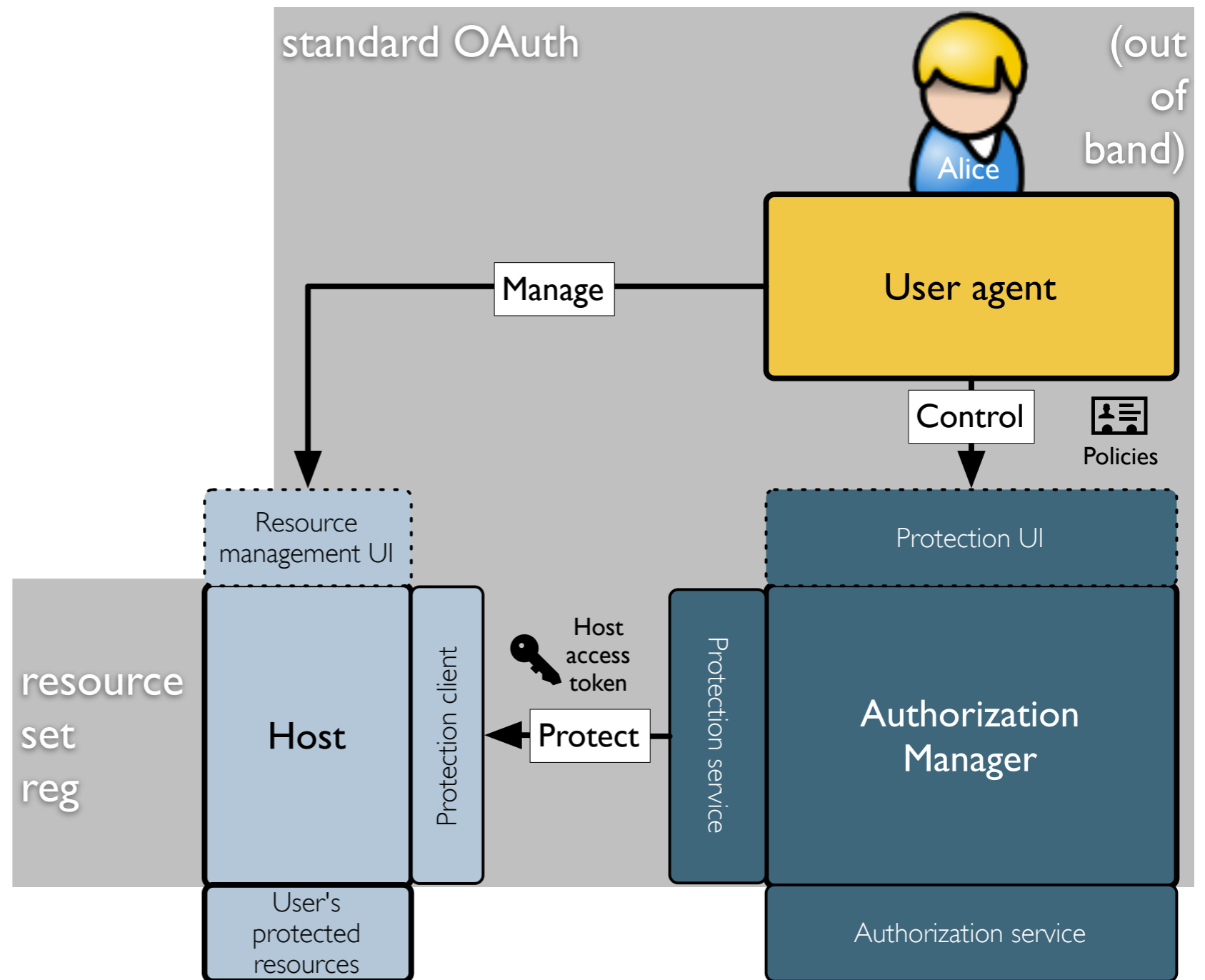
Working with resource set registration and scopes

- Scope URIs resolve to scope descriptions

  - They can live anywhere

- Host registers resource sets and maps to available scopes

  - Using RESTful API

```
{
        "scope":
            {
            "_id": "view"
            "name": "View Photo and Related Info",
            "icon_uri": "http://www.example.com/icons/reading-glasses.png"
        }
}
```

```
{
        "scope":
            {
            "_id": "all"
            "name": "All Actions",
            "icon_uri": "http://www.example.com/icons/galaxy.png"
        }
}
```

```
{
        "resource_set":
            {
            "_id": "112210f47de98100"
            "name": "Steve the puppy!",
            "icon_uri": "http://www.example.com/icons/flower",
            "scopes":
                        ["http://photoz.example.com/dev/scopes/view",
                        "http://photoz.example.com/dev/scopes/all"]
        }
}
```

```
PUT /host/photoz.example.com/resource_set/112210f47de98100 HTTP/1.1
Content-Type: application/json
...
```

http:/tinyurl.com/umawg

(intermission)

- The requesting party learns about the resource... *somehow*

  - Emailed link?

  - Discovery service?

  - Microformat data on Alice's blog?

- And it knows how to use the API and scopes at the host...*somehow*

  - Developer documentation?

  - Standardized scopes?

Authorizing user

Alice

User agent | Resource provisioning

Delegate

Requester | Resource discovery

User agent | Web service

Alice, Bob, or organization rep | Organization

Requesting party

http:/tinyurl.com/umawg
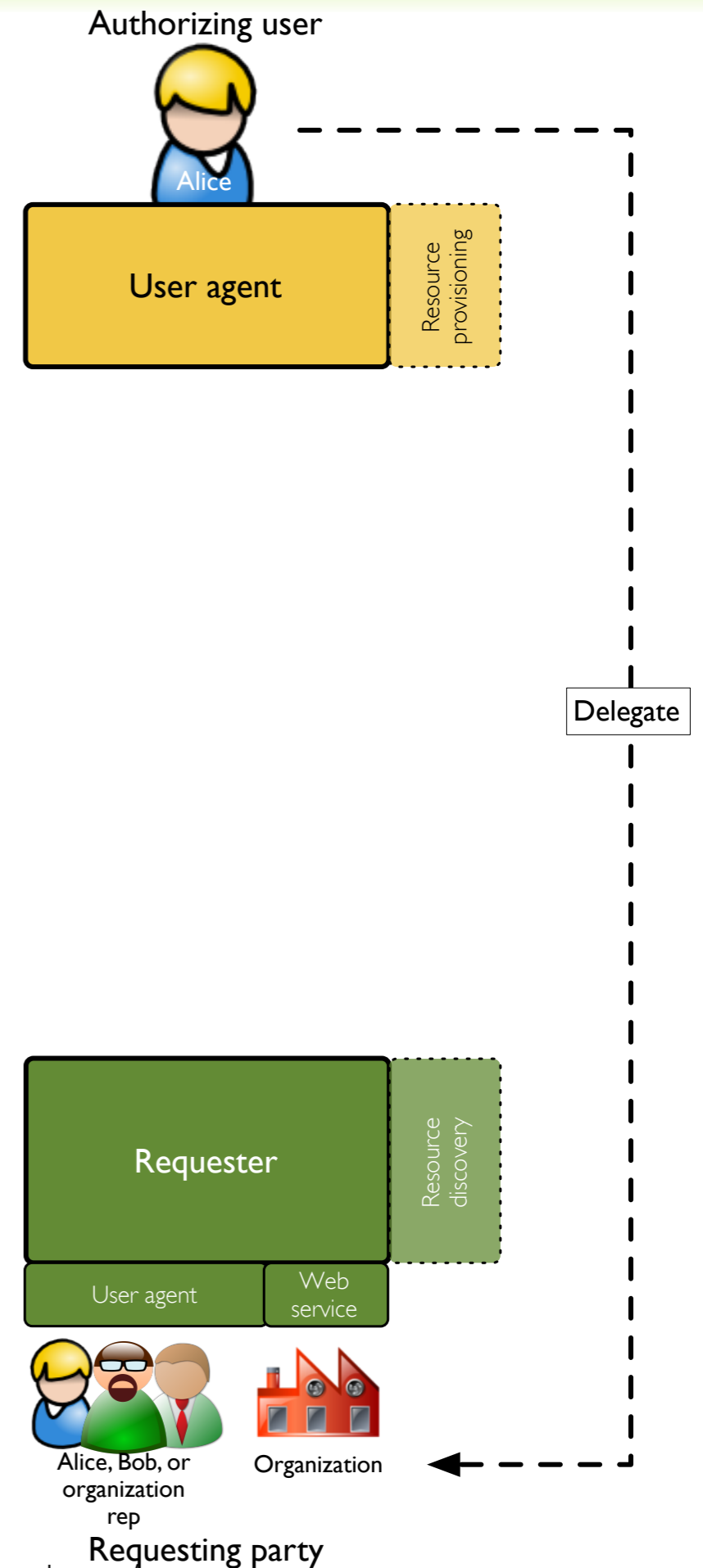
Phase 2: Get authorization

- Requester attempts access but has to get, in turn...
  - A token from AM *requester token endpoint*
  - Permission for sought-after scope from AM *authorization endpoint*
  - Likely providing *claims* to win permission

- Host uses AM *token status endpoint* to check each attempt by requester

- Host uses AM *permission registration endpoint* to register the sought-after scope

check token status;
register permission needed by requester

Host

Protection client

Host access token

Protect

Protection service

Authorization Manager

Authorization service

User's protected resources

Authorize

Permissions

Requester access token

Access

Authorization/access client

Requester

User agent

Web service

Alice, Bob, or organization rep

Organization

Requesting party

attempt at host;
resolve missing bits
at the AM in stages

If Alice is *also* the requesting party...

- She has a "synchronous" authorization experience because the claim she must provide is that she's Alice

  - Otherwise she doesn't have to be around when access is tried

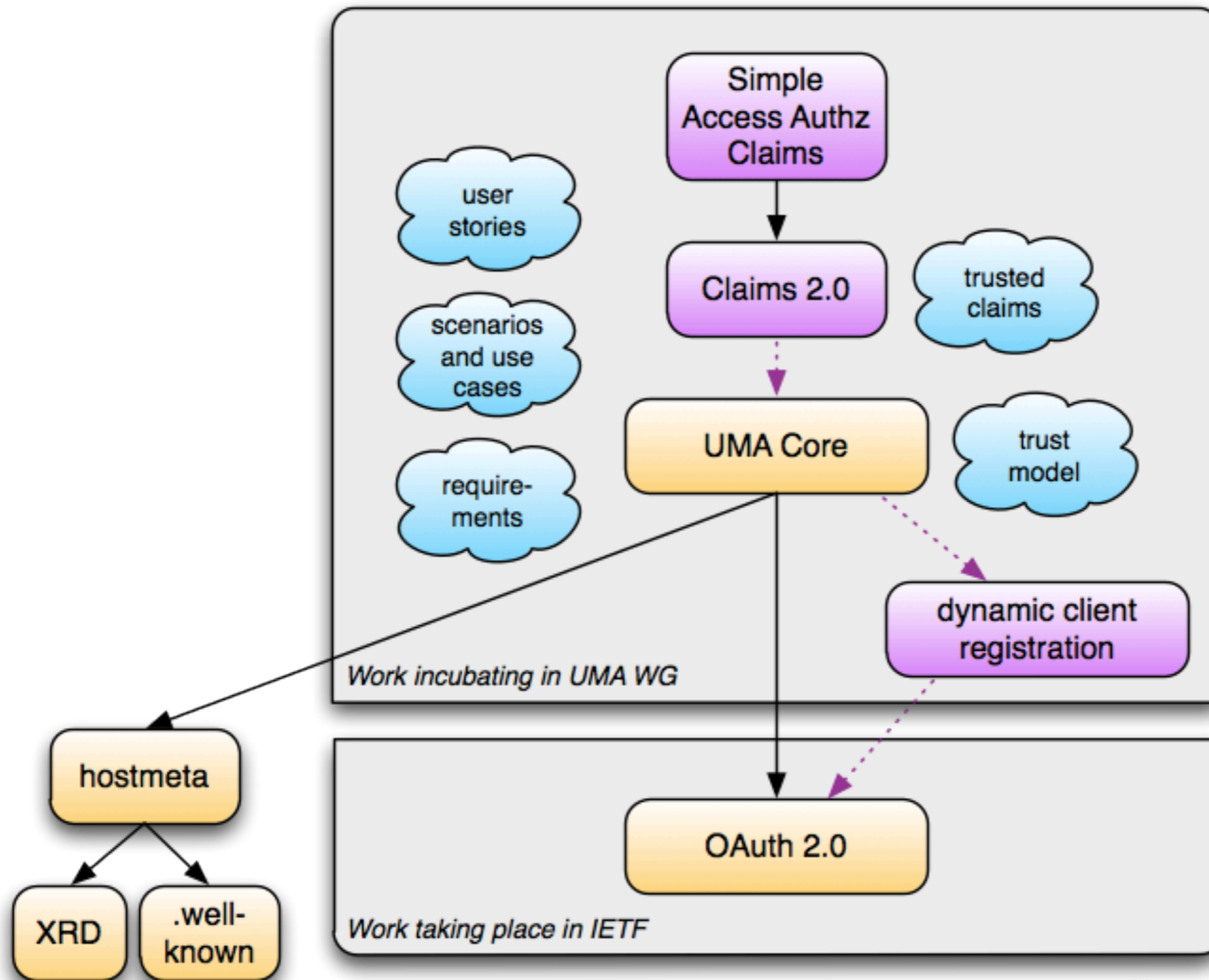- The flow would be the same for Alice, Bob, or anyone else who needs to prove they satisfy the policy

- We are working on OpenID Connect integration for basic interoperable "trusted claims"

oh yeah, that's me and I can prove it

Alice

Host

Protection client

Host access token

Protect

Protection service

Authorization Manager

Authorization service

User's protected resources

Permissions

Requester access token

Access

Authorize

Authorization/access client

Requester

User agent

Web service

policy: alice@gmail.com can get to this calendar

Alice, Bob, or organization rep

Organization

Requesting party

http:/tinyurl.com/umawg

Phase 3: Access a resource

- The happy path

Authorizing user

Alice

| User agent | Resource provisioning |

Manage

Control · Policies

Resource management UI

check

Host access token

Protection UI

Host · Protection client · Protect · Protection service · Authorization Manager

User's protected resources · Authorization service

attempt succeed

Permissions · Requester access token

Access · Authorize

Authorization/access client

Requester · Resource discovery

User agent · Web service

Delegate

Alice, Bob, or organization rep · Organization

Requesting party

The UMA spec "call tree"

29 May 2011

http:/tinyurl.com/umawg

Agenda

Short introduction: UMA concepts and benefits

Demo of UMA in action with the SMART system

How UMA works with OAuth under the hood

The UMA roadmap

Q&A

http:/tinyurl.com/umawg

Next steps organizationally

- The OAuth WG meeting at IETF81 in July will likely concentrate mostly on OAuth 2.0 completion

- Several new scoping items, including standardizing the authz server/resource server interface, have been brought up

  - We are working to socialize our proposed solutions

- We will keep fleshing out the spec (updating the oauth-umacore I-D)

  - We may proceed to a Kantara All-Member Ballot to try for Recommendation status as well

Next steps technically

- Open-sourcing of Java implementation behind SMARTAM (UMA/j)

    - Also likely Python implementations of host and requester code

- New mobile implementations (Fraunhofer AISEC)

- Experimental deployments by a variety of "Personal Data Ecosystem" companies

- Solving the remaining hard problems for V1.0 and beyond, for example:

    - Profiles for trusted claims handling and OpenID Connect integration

    - Responding to specialized use cases – e.g., secure dynamic discovery and highest security for Project hData healthcare needs

Agenda

Short introduction: UMA concepts and benefits

Demo of UMA in action with the SMART system

How UMA works with OAuth under the hood

The UMA roadmap

Q&A

http:/tinyurl.com/umawg

Thanks for joining us today

Become an UMAnitarian!
Webinar recording will appear soon!

Eve Maler and Maciej Machulak
on behalf of and with thanks to the UMA Work Group
13 July 2011

http:/tinyurl.com/umawg