# Macaroons and dCache

# … or delegating in a cloudy world

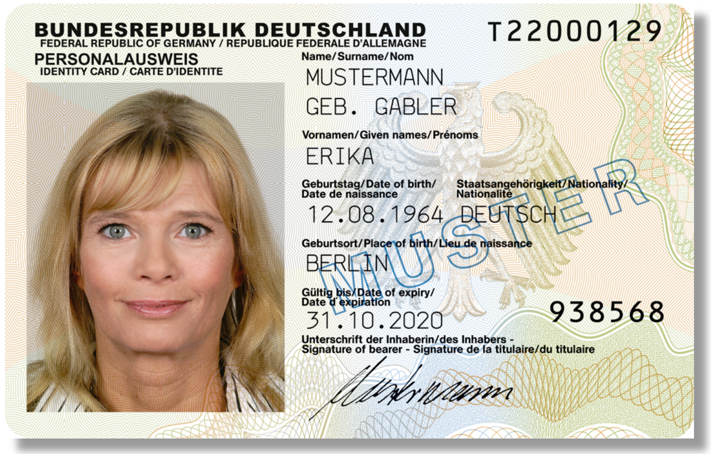
Patrick Fuhrmann
Paul Millar

On behave of the project team

## AAI … but

This talk is about the second 'A': **Authorisation.**

## Quick recap: which is which?


BUNDESREPUBLIK DEUTSCHLAND
FEDERAL REPUBLIC OF GERMANY / REPUBLIQUE FEDERALE D'ALLEMAGNE
PERSONALAUSWEIS
IDENTITY CARD / CARTE D'IDENTITE
T22000129
Name/Surname/Nom
MUSTERMANN
GEB. GABLER
Vornamen/Given names/Prénoms
ERIKA
Geburtstag/Date of birth/ Date de naissance
12.08.1964
Staatsangehörigkeit/Nationality/ Nationalité
DEUTSCH
Geburtsort/Place of birth/Lieu de naissance
BERLIN
Gültig bis/Date of expiry/ Date d'expiration
31.10.2020
938568
Unterschrift der Inhaberin/des Inhabers - Signature of bearer - Signature de la titulaire/du titulaire
MUSTER

Credential

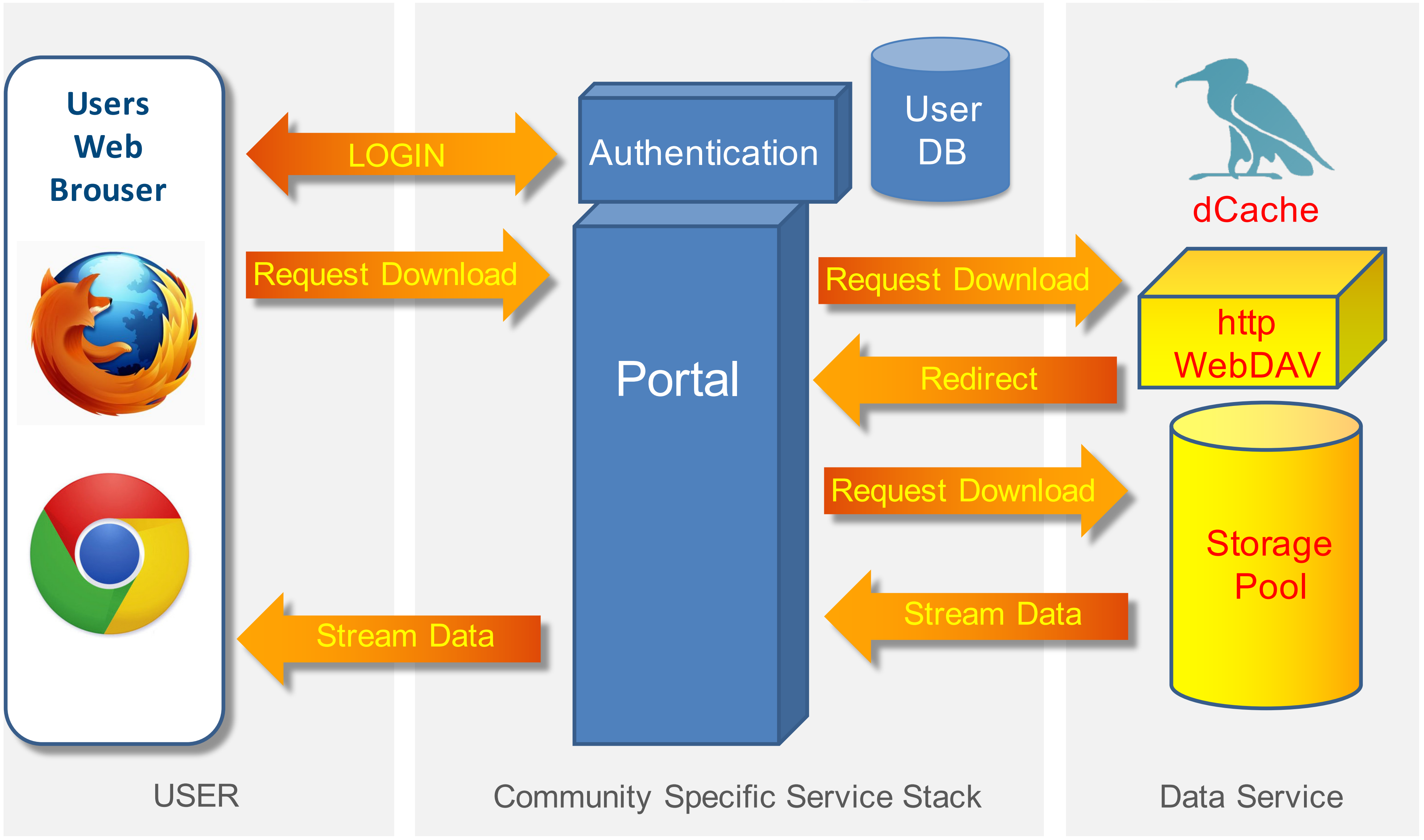
Authentication

Authorization

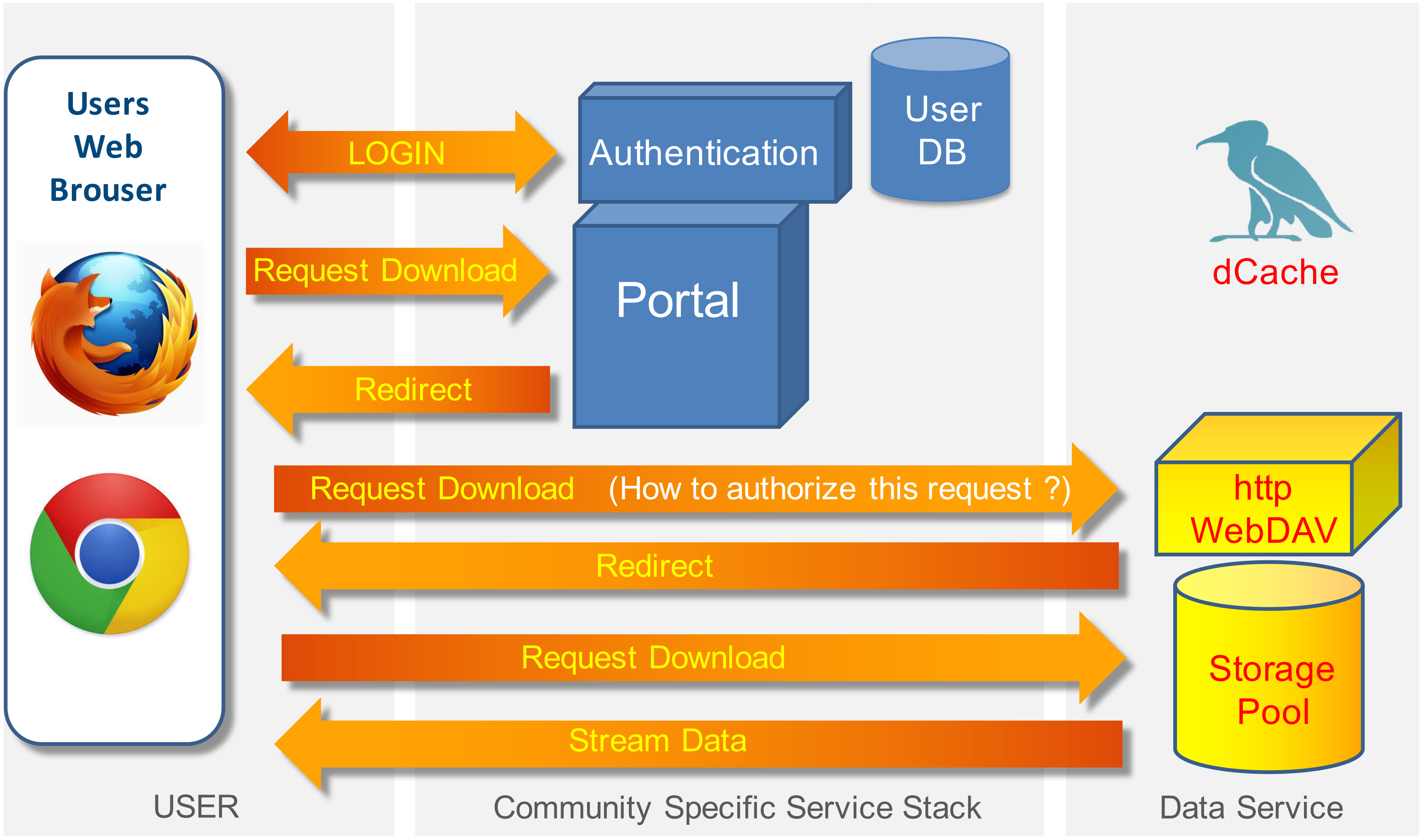
## Authorisation without authentication?

## That is this all about, Starting with a use-case

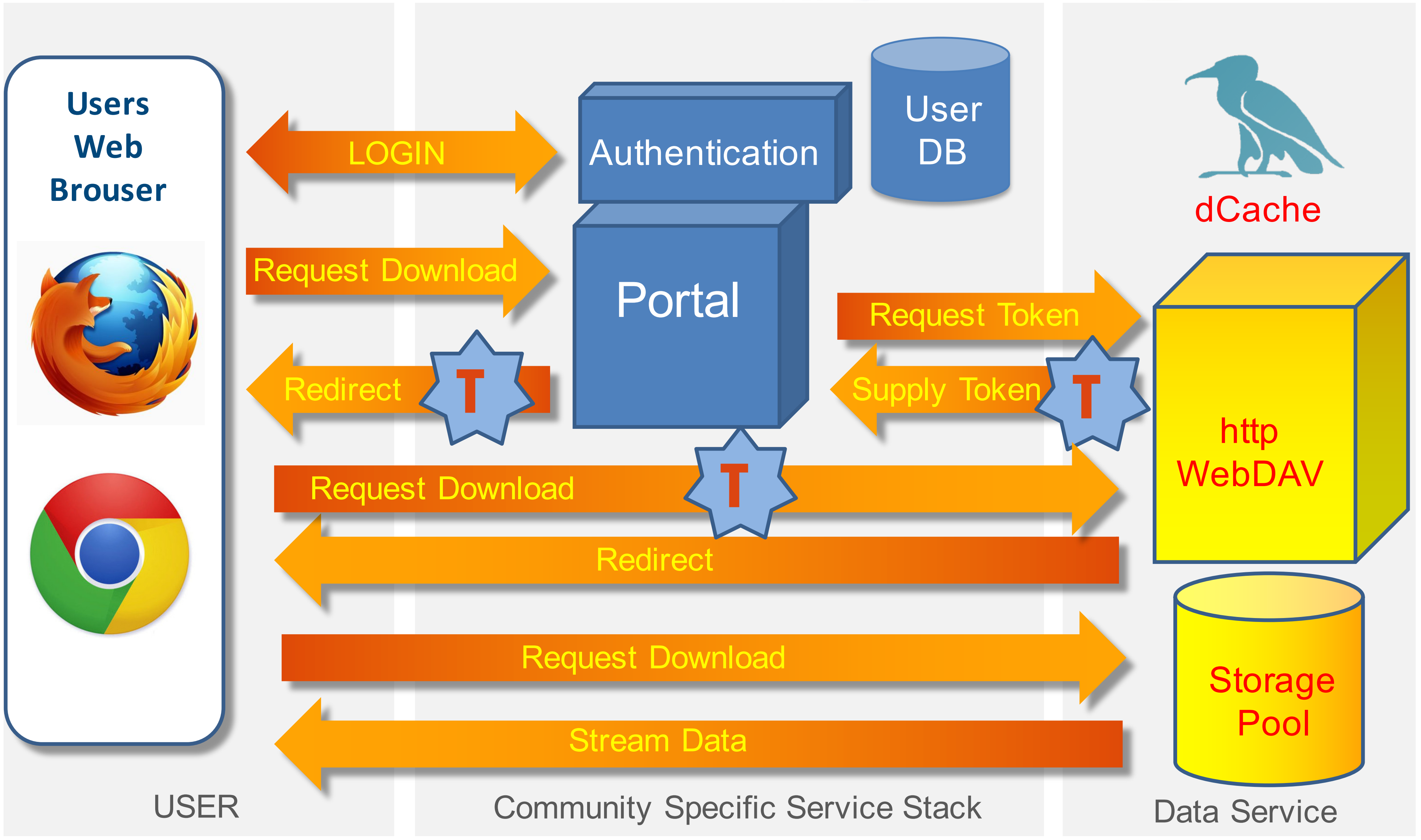
# Photon Science portal use-case



# Desired: client downloads directly



## Desired: client downloads directly



## What are bearer tokens?

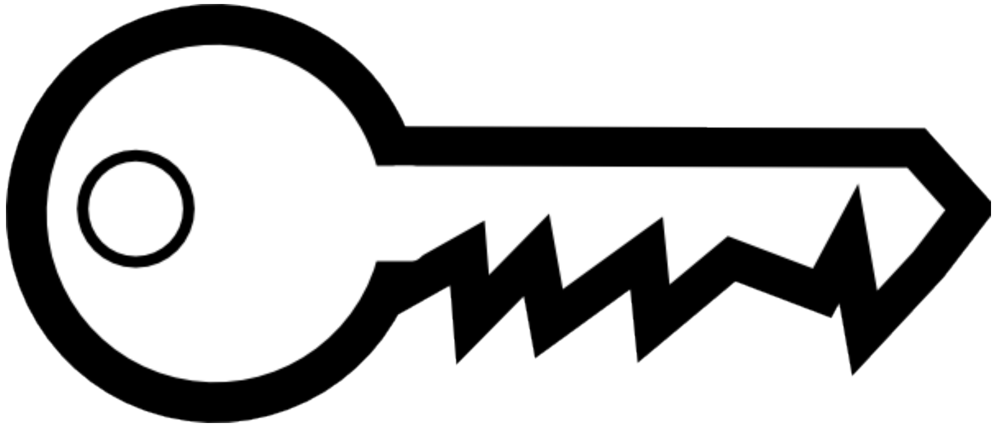
**Bearer token** is something the user presents with a request so the server will authorise it. There's no interaction between client and server.

Examples of bearer tokens:

- HTTP BASIC authn, anything stored as a cookies.

Counter-examples:

- X.509 credential,
- SAML,
- Kerberos.



## Bearer tokens for download authz

- Redirection should work **without JavaScript**,
- Simple: **embed token** in redirection URL.

**http://webdav.example.org/path/to/file?authz=<TOKEN>**

(There are nicer ways of embedding the token, but the URL is the only thing we can control)

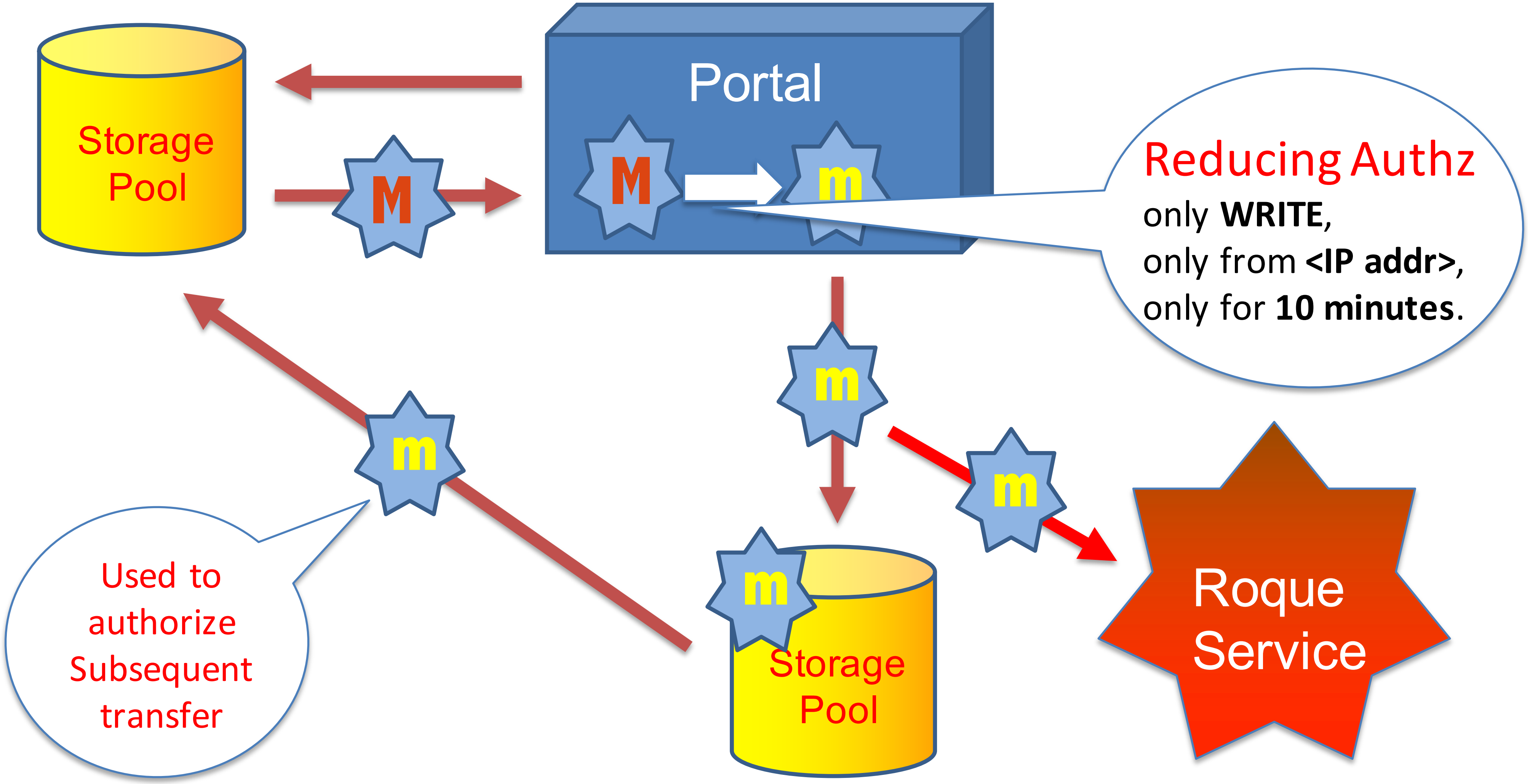
- **Complete token** always sent with the request.
- What can we do to stop someone **stealing** this token?
- … or make the token useless if they steal it.

## Introducing Macaroons

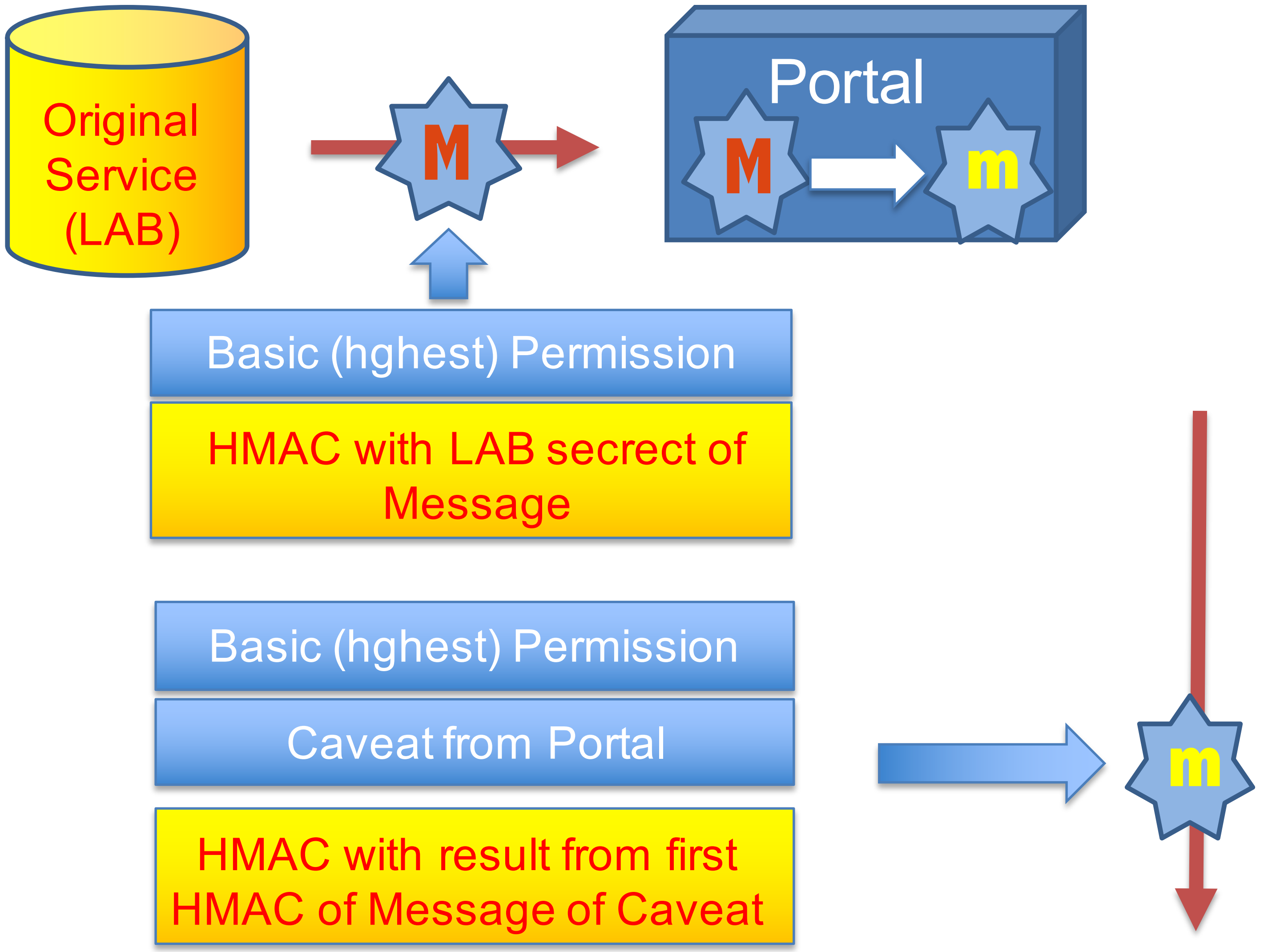
## Macaroons 101

- Macaroon is a **bearer token**.
- Macaroon contains zero or more **caveats**.
- Each caveat **limits** something:
  - **who** can use it, or
  - **what** they do with it.
- Anyone can **add** a caveat to a macaroon:
  - Create a new macaroon that is more limited.
- Nobody can **remove** a caveat from a macaroon.

# Example: 3rd party copy



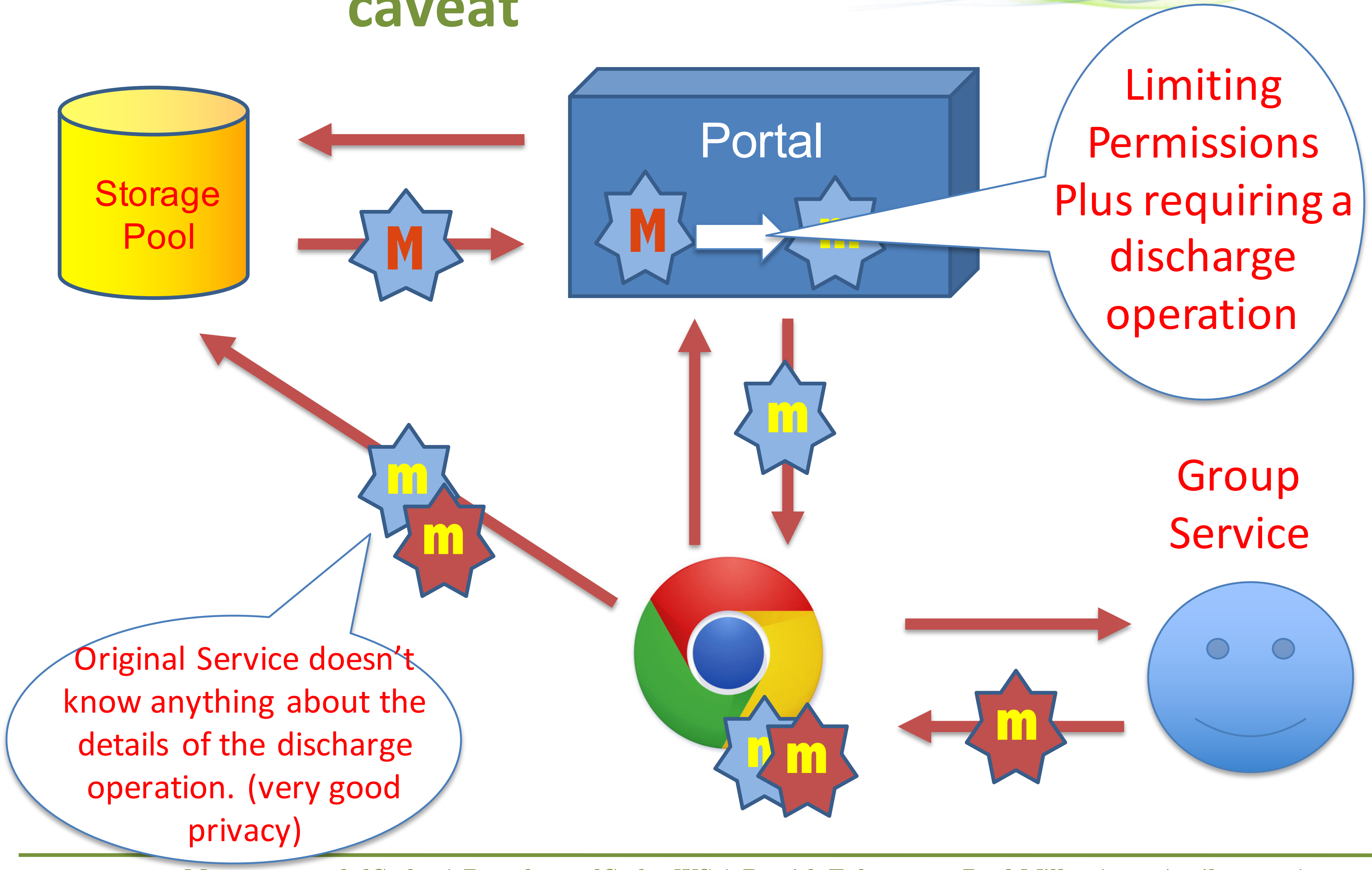
# A bit on security


Original Service (LAB)
M
Portal
M
m
Basic (hghest) Permission
HMAC with LAB secrect of Message
Basic (hghest) Permission
Caveat from Portal
HMAC with result from first HMAC of Message of Caveat
m

## 3rd party caveats – extra cool!

- 1st party caveat can be satisfied by the client.
- 3rd party caveat requires proof from some other service; e.g.
  - only **fred@facebook**,
  - only members of **VO ATLAS**,
  - only if not part of a **denial-of-service attack**.
- The proof is another macaroon: a **discharge macaroon**.

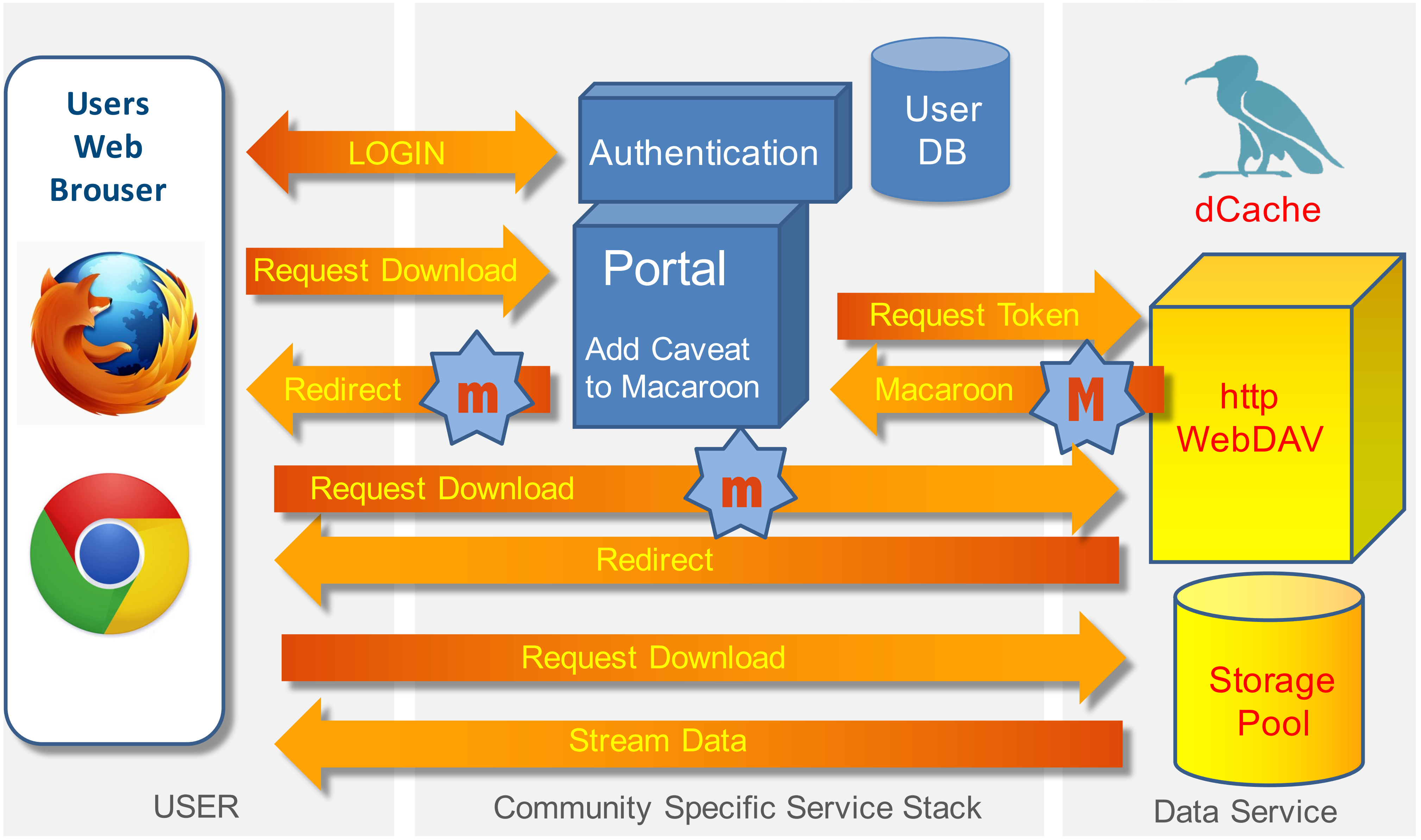
# Example: download w/3rd party caveat


Storage Pool
Portal
M
m
Limiting Permissions Plus requiring a discharge operation
Group Service
Original Service doesn't know anything about the details of the discharge operation. (very good privacy)

## Discharge macaroons

- The client proves it satisfies a 3rd party caveat by having a **discharge macaroon**.
- The original macaroon is only useful with a **valid** discharge macaroon.
- The discharge-macaroon can have **caveats**:
  - Short-lived discharge macaroon can be used to simulate X.509's certificate revocation list.
  - The discharge macaroon can have 3rd-party caveats.

## Solution revisited: macaroons



## For what else are macaroons good?

**Private Sharing!**

## Enabling sharing: a new interface

- **Create** a macaroon:
  - Need to know the macaroon to access the file.
- **List** macaroons:
  - Facilitate sharing files.
- **Facilitate** adding caveats:
  - Purely in-browser or server-side?
  - Third-party caveats? (e.g., member-of-ATLAS caveat)
- **Destroy** macaroons:
  - Unclear if this really makes sense.

# The END

Further reading :

On dCache

**www.dCache.org**

On macaroons by Google:

Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud.

Presentation

Paper

http://research.google.com/pubs/pub41892.html