



# Access Control

ECE568 – Lecture 17  
Courtney Gibson / David Lie  
University of Toronto ECE

---





Authentication

# Web Authentication

Last lecture we discussed the use of cookies to authenticate users on the web:

- On first visit, user is authenticated through a traditional method (i.e. username/password)
- If successful, a cookie is issued that is valid for a period of time. Subsequent visits within that period use the cookie.

# Problem

As number of websites grows, each website needs its own password and username combination:

- Users should use a different username and password for each website. Why?
- Each website needs to implement:
  - User authentication
  - New user registration
  - Password recovery
- Password credentials stored all over the place

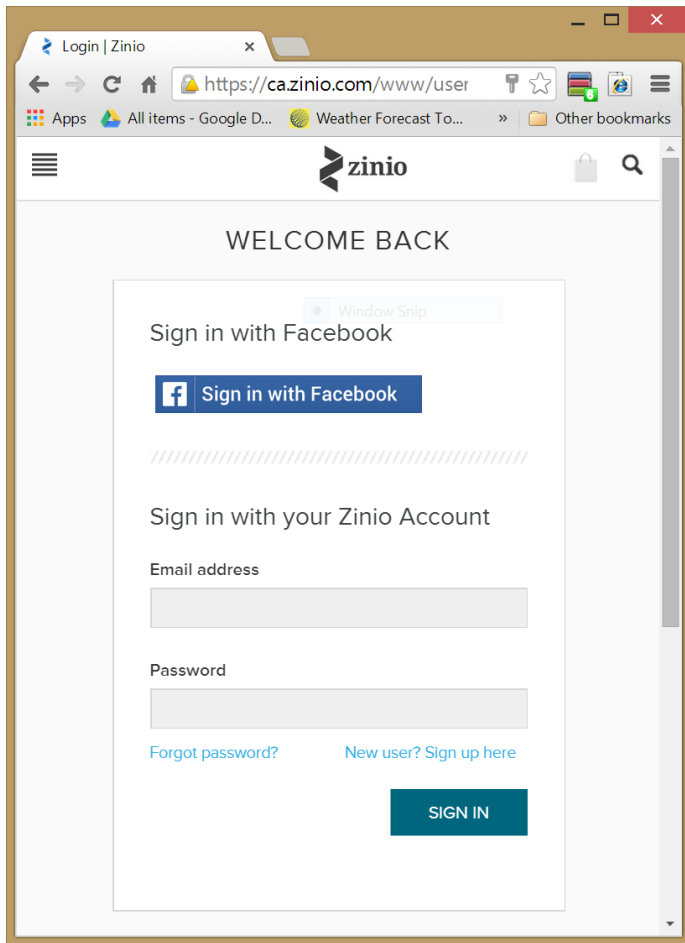


# Solution: Federated Identity

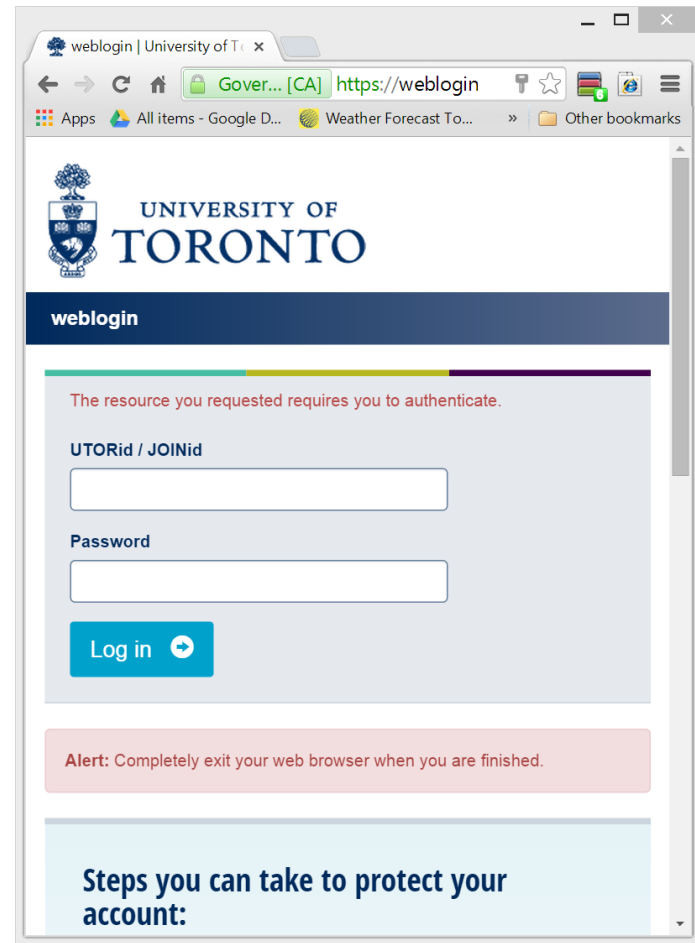
In **Federated Identity** systems, we designate a central **Identity Provider**

- A single service maintains a user's username and password. Handles new user registration and password recovery.
- Multiple services rely on the identity provider to help authenticate users that interact with the services.

# Examples



A screenshot of a web browser displaying the Zinio login page. The browser's address bar shows the URL <https://ca.zinio.com/www/user>. The page features the Zinio logo at the top center. Below the logo, the text "WELCOME BACK" is displayed. The main content area is a white box with a light gray border. It contains a "Sign in with Facebook" section with a Facebook icon and a "Sign in with Facebook" button. Below this is a "Sign in with your Zinio Account" section. It includes an "Email address" label and a text input field, followed by a "Password" label and another text input field. At the bottom of this section are two links: "Forgot password?" and "New user? Sign up here". A large blue "SIGN IN" button is positioned at the bottom right of the white box.



A screenshot of a web browser displaying the University of Toronto weblogin page. The browser's address bar shows the URL <https://weblogin>. The page features the University of Toronto crest and name at the top. Below the header, the text "weblogin" is displayed. The main content area is a light gray box. It contains a red message: "The resource you requested requires you to authenticate." Below this is a "UTORid / JOINid" label and a text input field, followed by a "Password" label and another text input field. A blue "Log in" button with a right arrow icon is positioned below the password field. At the bottom of the gray box is a red alert box with the text: "Alert: Completely exit your web browser when you are finished." Below the alert box is a light blue box with the text: "Steps you can take to protect your account:".

# Case Study: The Kerberos System

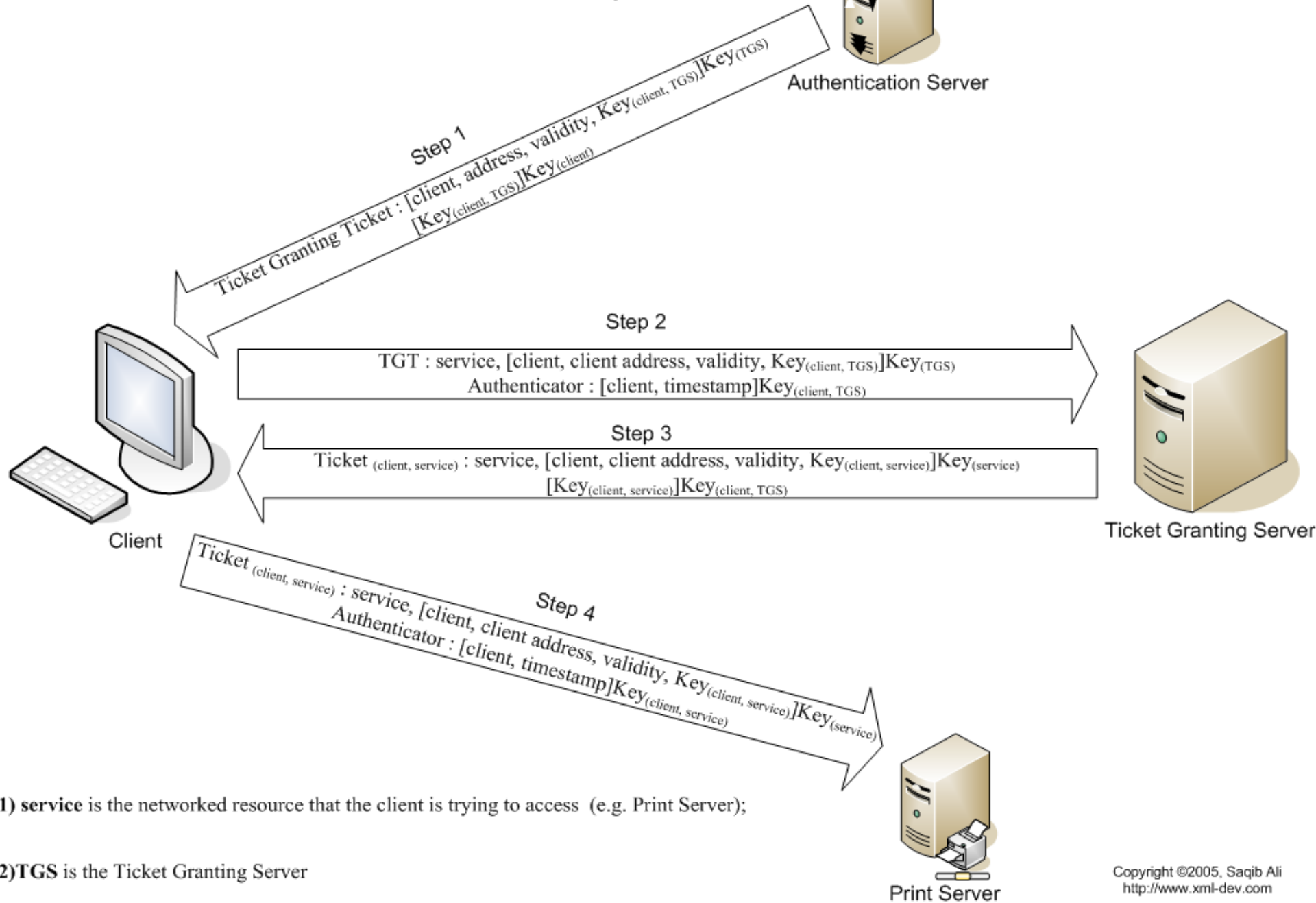
Kerberos is a network wide authentication service that controls access to a set of servers:

- It uses a trusted third party and the symmetric Needham-Schroeder protocol for key exchange

The trusted third party consists of an **authentication server** and a **ticket-granting server**

- The authentication server is used for user authentication
  - Knows all user passwords
  - It provides tickets-granting ticket to a user indicating that the user is authorized to use the ticket-granting service
- The ticket-granting server performs access control
  - It provides **tickets** to a user indicating that the user is authorized to use some service
  - Knows secret keys of all services

# Kerberos Operation





# The Kerberos System

A ticket is a short-term key that indicates the user's name, what services she can access and an expiry time, and it is encrypted with the secret key of service

- It is essentially a capability that the user presents to gain access to each service
- Windows 2000 and later use Kerberos as their default authentication method
- For more information:

<http://web.mit.edu/kerberos/www/papers.html>

# Single Sign-On vs. Federated Identity

Related and often used interchangeably, but not exactly the same:

- **Single Sign-on (SSO)** is a “what”:  
Describes what users have to do. They authenticate once and then can access multiple services.
- **Federated Identity** is a “how”: Describes how authentication is achieved.

# Examples

When is something both SSO and Federated Identity?

- UofT Services
- Google Services

Login via one central service, then use that authentication to later access other services.



# Examples

When is it only SSO?

- When no outsourced identity is being provided to other external providers
  - e.g., Amazon, ECF
- Non-Federated, SSO-only systems are becoming rarer: often SSO systems provide the ability to federate identity
  - (Why? Because providers want data on your online activity!)

# Examples

When is it only Federated Identity?

- Again, less-usual... but a possible situation when service providers don't trust each other
- Uses a central authentication service, but requires re-authentication each time you use a new service

# How it works

Typical web flow today:

1. User authenticates to the identity service provider.
2. Service provider sends user a unforgeable “token” that contains the user’s identity
3. User presents the token to the service they want to access, which accepts it in lieu of authenticating the user
4. The service now has a certified identity of the user



# Security analysis

The token must be **unforgeable**.

- Otherwise, an attacker could just guess it and impersonate the user.

The token must not be **leaked** to an attacker.

- Neither the user nor the authentication service are involved in validating the token.

# Example: OAuth

- Open specification, many existing open-source implementations
- Used by Facebook and Google, two common public identity providers
- Two versions:
  - **OAuth 1.0** published in 2010
  - **OAuth 2.0** published in 2012: adds user-provisioning
  - Note 2.0 is not backwards compatible with 1.0. Combining the two can lead to security problems. Most providers use 2.0

# OAuth

We've been talking about **authentication**. OAuth is commonly used for authentication. However, it's actually an **authorization** protocol. What's the difference?

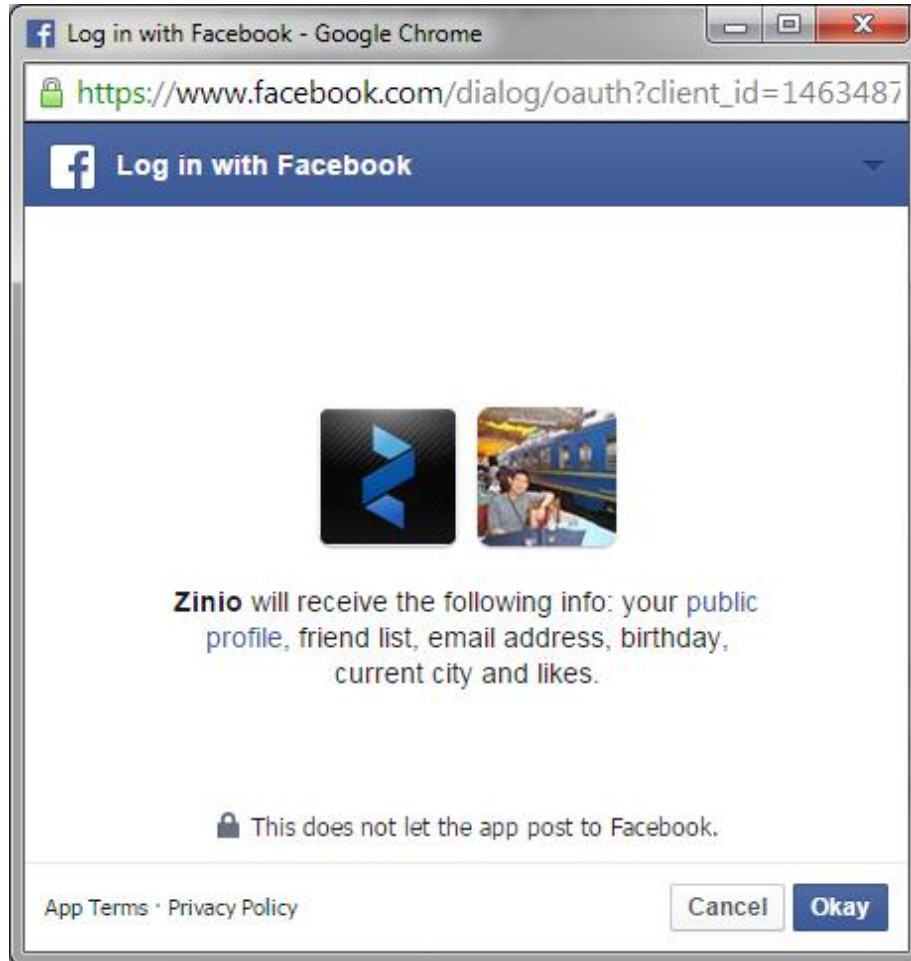


# OAuth

While the goal is different, Authentication and Authorization are related in this flow:

- ◉ **Authentication:** A service wants to determine the identity of an *untrusted user*.
- ◉ **Authorization:** A user wants to permit an *untrusted service* access to *sensitive resources* (e.g., the user's identity and data).

# OAuth



# Authorization Code Grant Flow

OAuth supports several authorization protocols. One of the commonly used ones is called **Authorization Code Grant**.

There are 3 parties in any authorization/authentication event

1. The **Resource Owner** (*i.e.*, the User)
2. The **Authorization Server** (*i.e.*, Facebook)
3. The **Client** (*i.e.*, Zinio)



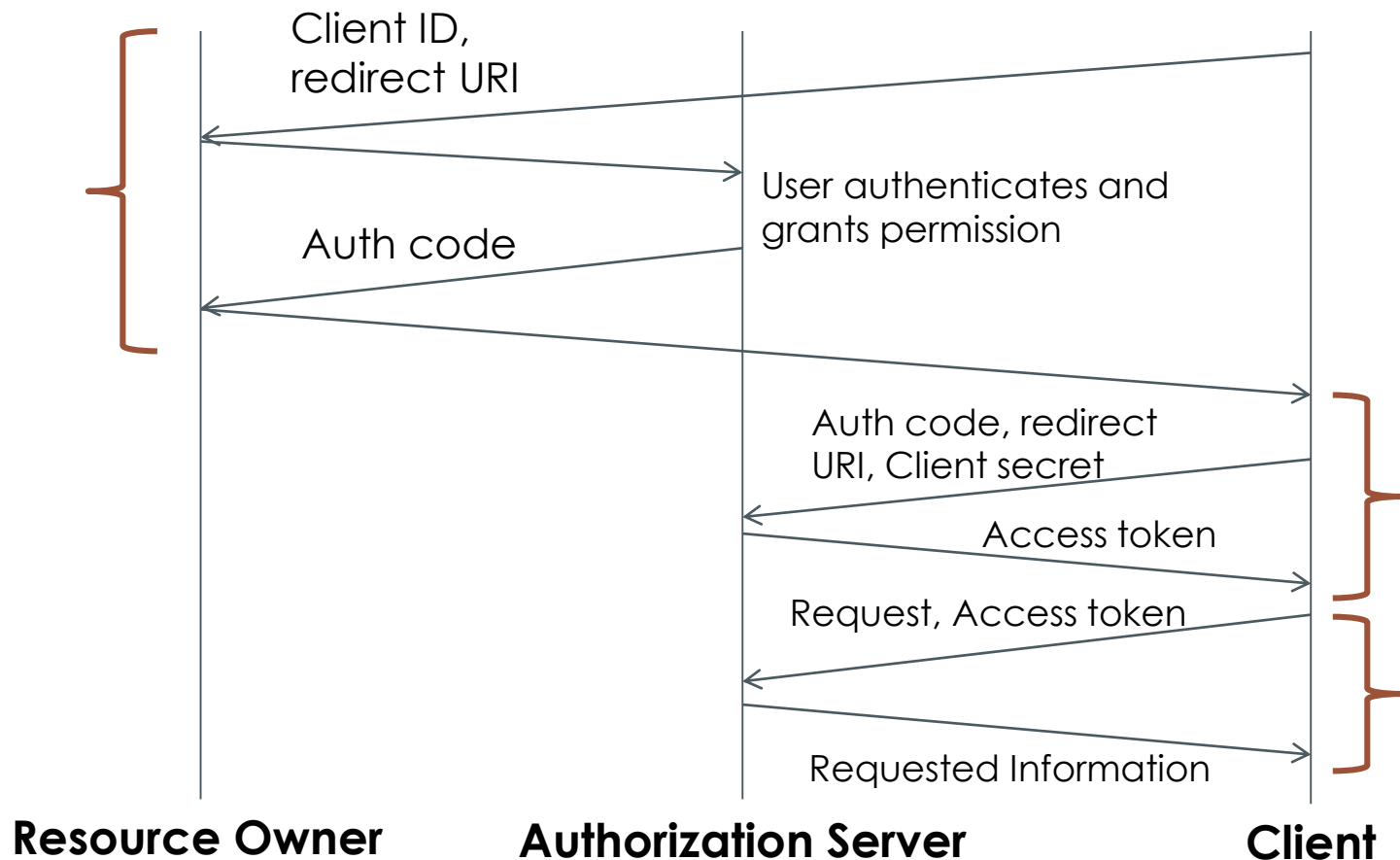
# Authorization Code Grant Flow

Before authorization can occur, the client must register with the authorization server.

Client will:

1. Provide one or more **URIs** where tokens should be sent
2. Receive a unique **Client ID** that identifies the client to the authorization server

# Authorization Code Grant Flow



# Redirects

Note that the Resource Owner is responsible for:

- Transferring the Authorization request (i.e. Client ID, redirect URI) from Client to Auth Server
- Transferring Auth code form Auth Server to Client

If Resource Owner is a web browser used by an end-user (usually the case), this is usually accomplished by using an HTTP 302 redirect.

- User may notice the web browser flicker as it automatically takes the response and sends it to the next party.

# Summary

- Lots of advantages of Federated Identity:
  - No need to roll your own authentication
  - User credentials stored and managed in one place
- Understand the relationship between authorization and authentication.
- Revocation is fine for authorization, but tricky for authentication.





Questions?