Introduction to SciTokens

Brian Bockelman,
On Behalf of the SciTokens Team

https://scitokens.org

This material is based upon work supported by the National Science Foundation under Grant No. 1738962. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

SciTokens: Federated Authorization Ecosystem for Distributed Scientific Computing

- The SciTokens project, starting July 2017, aims to:
    - Introduce a ***capabilities-based*** **authorization infrastructure** for distributed scientific computing,
    - provide a **reference platform**, combining CILogon, HTCondor, CVMFS, and Xrootd, AND
    - **Implement an instance** to help our science stakeholders (LIGO and LSST) better achieve their scientific aims.
- In this presentation, I’d like to unpack what this means, give a short demo, and outline possible use cases for the WLCG.

Capabilities-based Auth Infrastructure

- At the core of today’s AAI is the concept of *identity* and *impersonation*.
    - A grid certificate provides you with a globally-recognized identification.
    - The grid proxy allows a third party to impersonate you, (ideally) on your behalf.
    - The remote service maps your identity to some set of locally-defined authorizations.
- We believe this approach is fundamentally wrong because it exposes too much global state: identity and policy should be kept locally!

Capabilities-based Auth Infrastructure

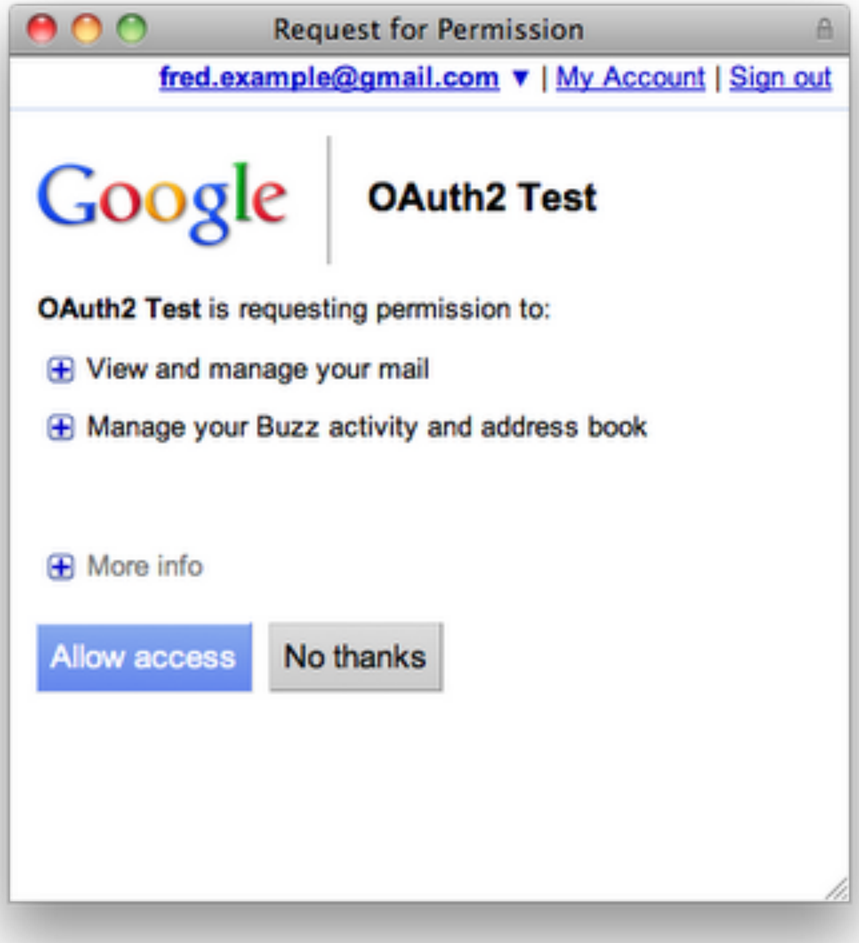
- We want to change the infrastructure to focus on *capabilities*!
    - The tokens passed to the remote service describe what authorizations the bearer has.
    - For traceability purposes, there may be an identifier that allows tracing of the token bearer back to an identity.
    - Identifier != identity. It may be privacy-preserving, requiring the issuer (VO) to provide help in mapping.
- Example: “The bearer of this piece of paper is entitled to write into /castor/cern.ch/cms".

Capabilities versus Identities

- If GSI took over the world, an attacker could use a stolen grid proxy to make withdrawals from your bank account.
- With capabilities, a stolen token only gets you access to a specific authorization (“stageout to /store/user at Nebraska”).

The World Uses Capabilities!

- The rest of the world uses capabilities for distributed services.
    - The authorization service creates a token that describes a certain capability or authorization.
    - Any bearer of that token may present it to a resource service and utilize the authorization.
- The primary way this is implemented is through OAuth2.
- When you click “allow access” on the right, the **client** at “OAuth2 Test” will receive a token. This token will permit it to access the listed subset of Google services for your account.
- OAuth2 is used by Microsoft, Facebook, Google, Dropbox, Box, Twitter, Amazon, GitHub, Salesforce (and more) to allow distributed access to their identity services.



Request for Permission

fred.example@gmail.com ▼ | My Account | Sign out

Google | OAuth2 Test

OAuth2 Test is requesting permission to:

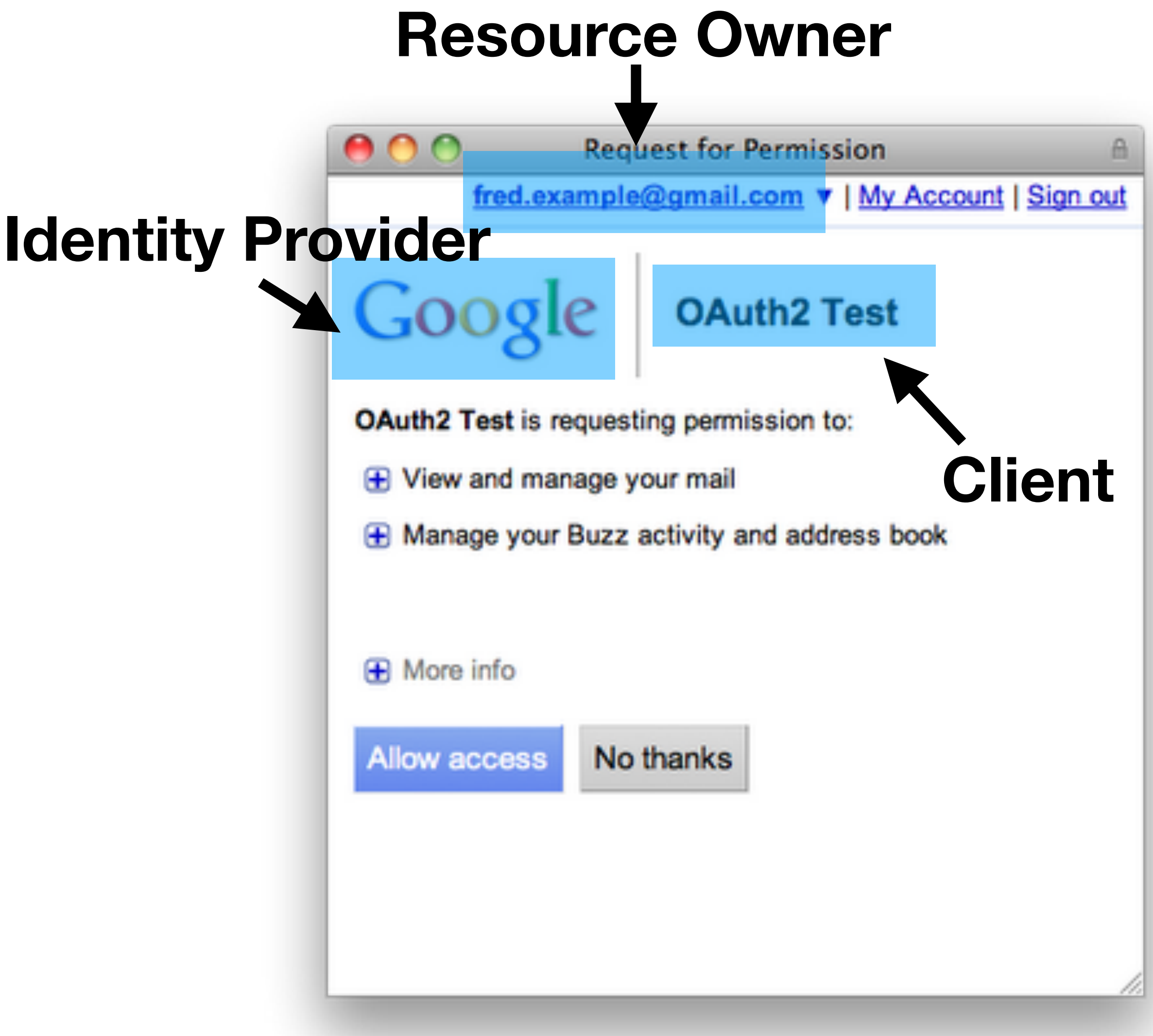
⊞ View and manage your mail

⊞ Manage your Buzz activity and address book

⊞ More info

Three-Legged Authorization

- In OAuth2, there are three abstract entities involved in the authorization workflow:
  - **Authorization server** (identity provider) issues capabilities.
  - The **resource owner** (end-user) approves authorizations.
  - The client receives tokens. Often, this is the third-party website or smartphone app.
- Once the token is issued, it can be used at the **resource server** to access some protected resource.
  - In the Google example, Google runs both the authorization and resource servers.



Resource Owner

Identity Provider

Client

Request for Permission

fred.example@gmail.com ▾ | My Account | Sign out

OAuth2 Test is requesting permission to:

⊞ View and manage your mail

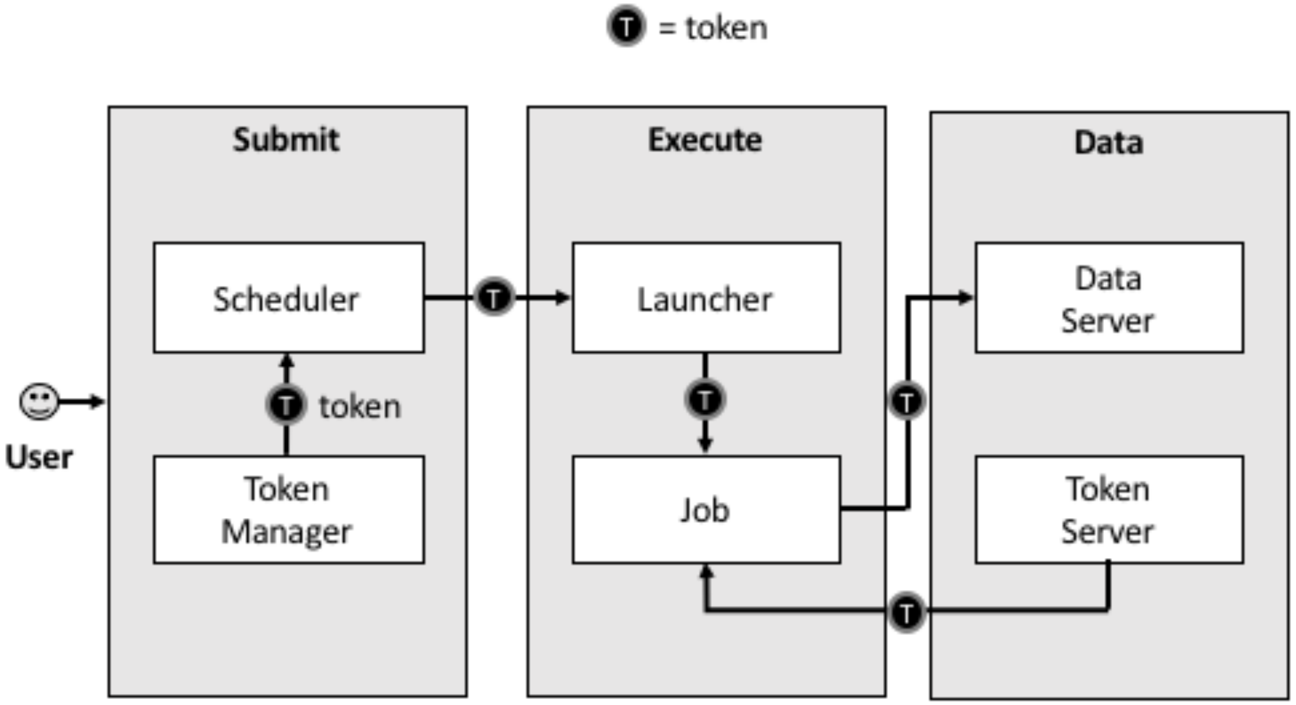
⊞ Manage your Buzz activity and address book

⊞ More info

SciTokens

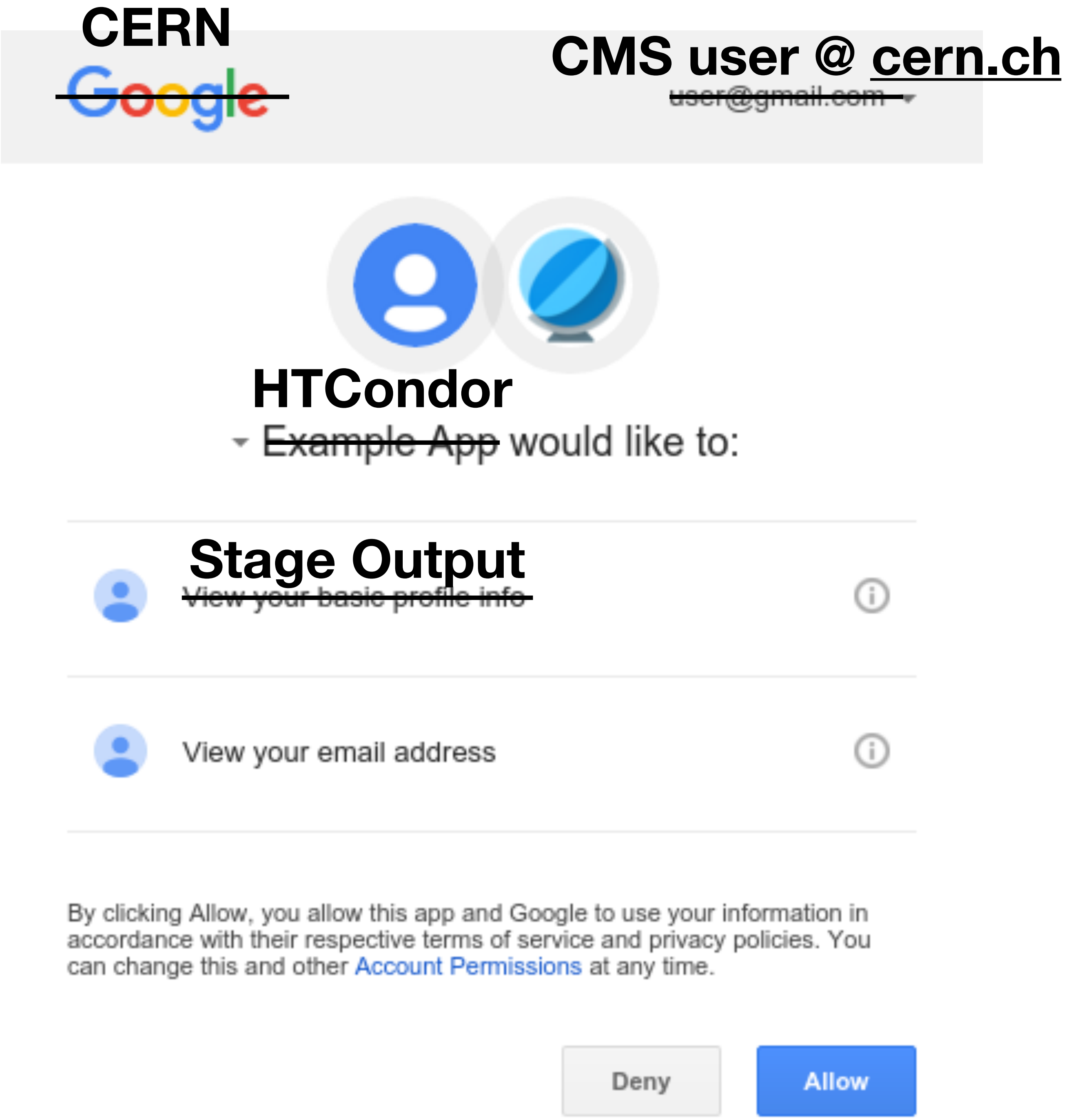
- The SciTokens team is working to integrate an OAuth2 client into the HTCondor submit host.
    - OAuth2 support at CILogon is being enhanced with VO-defined scopes.
- HTCondor is being enhanced to manage the token lifetime (refreshing as needed), possibly attenuating it, and delivering it to the job.
- Data services (CVMFS, Xrootd) are being enhanced to allow read/writes utilizing tokens instead of grid proxies.

The SciTokens Model



End-Goal

- The end-goal is this ->
- The first time you use HTCondor, you navigate to a web interface and setup your desired permissions.
  - On every subsequent `condor_submit`, HTCondor will transparently create the access token for you. *User sees nothing*.
- Replace CERN, usernames, and authorization as desired.
  - **Goal**: our first use of OAuth will be to stageout from payload jobs to Box.



CERN ~~Google~~

CMS user @ cern.ch ~~user@gmail.com~~

HTCondor ~~Example App~~ would like to:

Stage Output ~~View your basic profile info~~

View your email address

By clicking Allow, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.

Deny Allow

~~USER MANAGEMENT OF FILES~~

~~PASSWORD IN TERMINAL~~

~~SCITOKENS-PROXY-INIT~~

~~COPY/PASTE~~

Tokens for Distributed Infrastructures

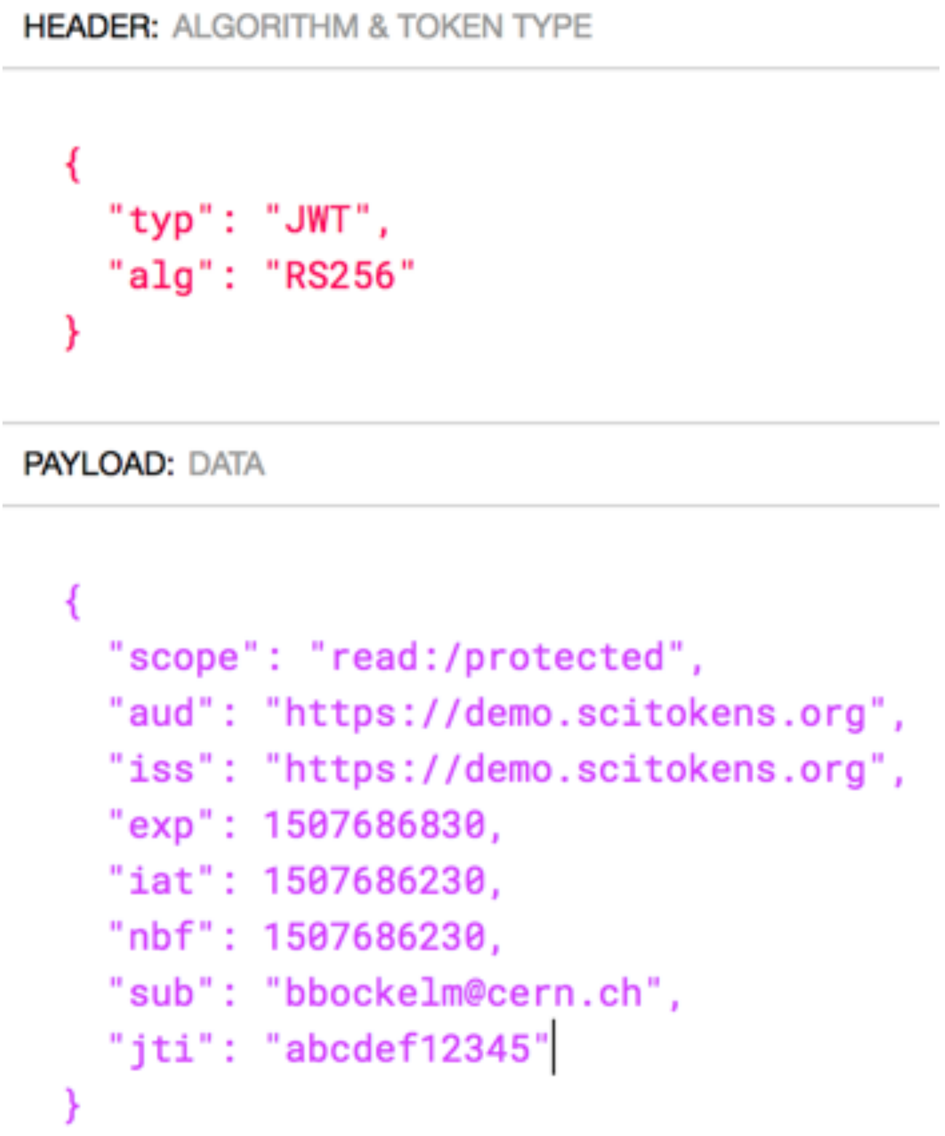
- Distributed science infrastructures are distinct from a “resource server” like Google because they are not run by a single central entity.
- Hence, unlike Google, we can’t use opaque random strings for the token. We need something that allows for **distributed verification**.
  - Given a token, a storage service can determine it is valid.
  - Analogously, given a proxy chain and a set of trust roots, you can determine the GSI proxy is valid.
- Goal: Sites set aside some area for each VO; VOs manage the authorizations within these “VO home” areas.

demo.scitokens.org

- **Free tokens!** Navigate to https://demo.scitokens.org to get your **free tokens**!
- This demo illustrates the access token format we're working on.
  - Utilizes JSON Web Tokens (JWT) as the access token format.
  - Various RFCs provide clear guidance on how to verify token integrity.
  - Adds a few domain-specific claims for receiving access to storage.
- The tokens are base64-encoded and can be used as part of a curl command to use protected resources.

Example Token, Decoded

- The decoded token contains multiple scopes - basically filesystem authorizations.
- The `aud`ience narrows who the token is intended for.
- The `iss`uer identifies who created the token; value used to locate the public keys needed to validate signature.
- The `sub`ject is an opaque identifier for the resource owner. In this case, it also happens to be the identity.
- The `exp`iration is a Unix timestamp when the token expires. A typical lifetime is 10 minutes.



HEADER: ALGORITHM & TOKEN TYPE

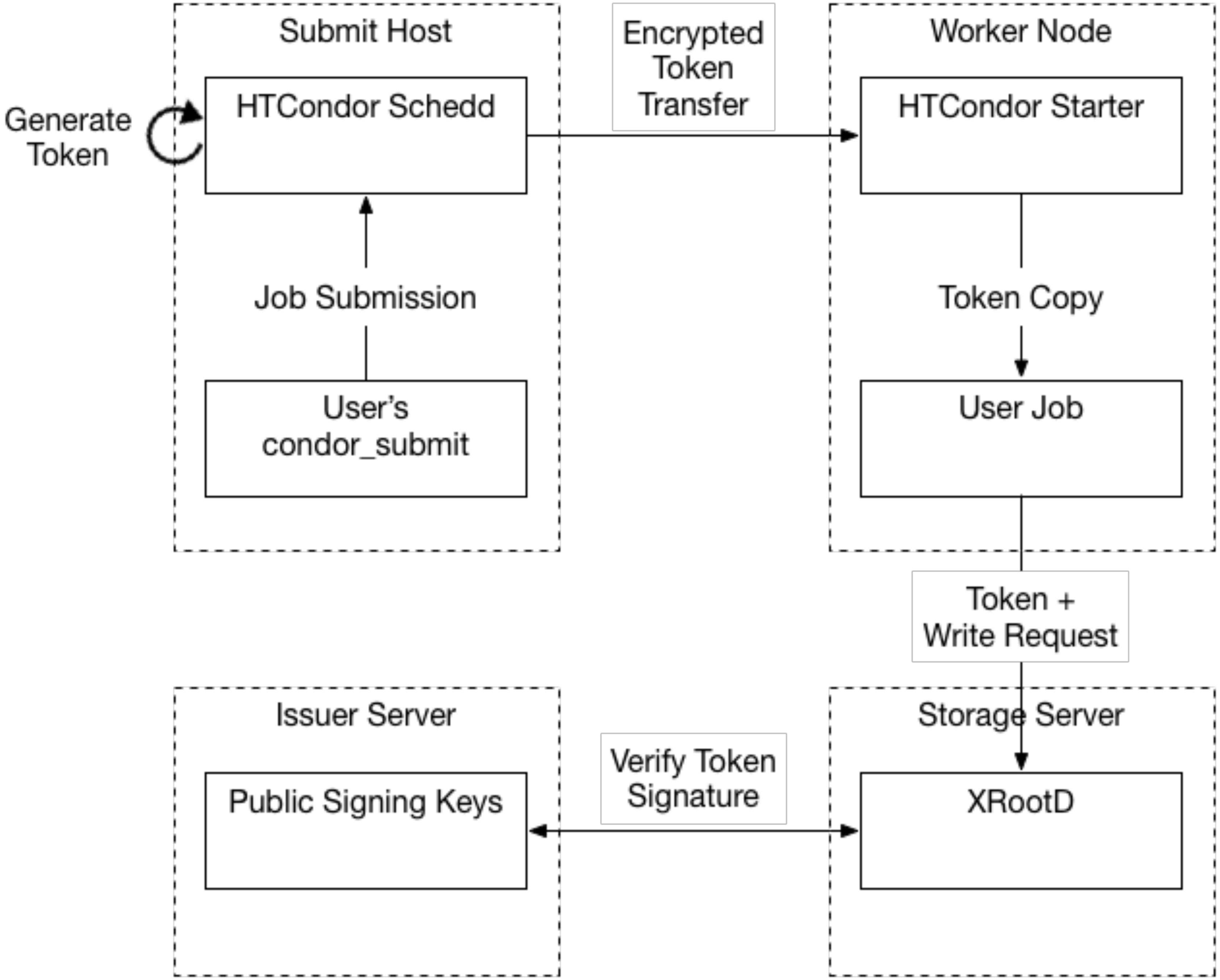
```
{
  "typ": "JWT",
  "alg": "RS256"
}
```

PAYLOAD: DATA

```
{
  "scope": "read:/protected",
  "aud": "https://demo.scitokens.org",
  "iss": "https://demo.scitokens.org",
  "exp": 1507686830,
  "iat": 1507686230,
  "nbf": 1507686230,
  "sub": "bbockelm@cern.ch",
  "jti": "abcdef12345"
}
```

OSG Demo

- We have been able to get a basic end-to-end token-based auth{z,n} workflow working for the OSG VO submit service.
- *This includes* patches to Xrootd to validate tokens presented via HTTP and to write files out with the correct Unix user permissions.
- **Cheats**:
  - instead of using OAuth2 to generate the token, we keep a signing key on the submit host.
  - only one token needed.
  - submit host and storage server owned by OSG.



Wait, I’ve seen this before!

- If you’re from ALICE and getting a sense of déjà vu — you’re right!
    - The capability-based infrastructure is precisely the authorization infrastructure used by ALICE for the **past decade**.
    - SciTokens takes this **successful model**, recasts it using modern web protocols, and utilizes OAuth2 workflows to issue the tokens.
- The use of common protocols and workflows means that we have a large number of battle-tested libraries we can leverage (spend our time doing other stuff besides writing the basics!).
- Using JWT-formatted access tokens is somewhat-commonplace among web companies.
    - I *think* SciTokens is unique in using JWT access tokens for distributed verification in a federated infrastructure.

Implications for WLCG

- As CMS uses a very similar technology stack as the SciTokens project, this would provide a mechanism to begin removing CMS user proxies from the worker node.
    - Proxies were required for glexec, but this is already phased out at some sites.
- Working on a “token exchange service” - given a valid VOMS-based authentication, will issue a corresponding SciToken.
    - An entity - think, FTS3 - with a delegated user proxy could then do a HTTPS transfer without the client cert.
- Combined with the WebDAV COPY command (already supported by FTS3), FTS3 could do a HTTPS 3rd-party-copy without needing GSI credentials at either end.
    - At the site level, this would be a “completely Globus free” transfer both in terms of concepts (GSI) and implementation (Globus Toolkit). **Significant impact**!
    - Toward this end, have a prototype implementation of WebDAV COPY working with Xrootd. With some small FTS3 / GFAL / DAVIX plumbing work, could demonstrate this between a Xrootd host and a DPM or dCache host.

Near-Term Goals

- By the end of the calendar year, we aim to:
    - Have version 1.0 of python and Java libraries.
    - Simple HTCondor OAuth client implementation.
    - Release XRootD token validation plugins.
    - Demonstrate token-based CVMFS access.
    - Demonstrate X509-to-SciToken translation service.
- Within the next 12 months:
    - Use Java library for a dCache authorization plugin.
    - Release plugin for CVMFS support.
    - More fine-grained token management in HTCondor.
    - Integration with LIGO LDAP.
    - Demonstrate 3rd-party HTTPS FTS transfers authorized with SciTokens.

Questions?

(I left out many technical details!)