RESTauth

- “REST” roughly means “at the app layer”
  - But it can be **orthogonal** to the app – one part of an app can be written in FCGI, another in WSGI, and so on
    - Deploy as a plugin to your HTTP server
  - Sessions get URIs, so... the app can GET them!
    - The app can observe things like *session keys*,
      - ...authenticated ID and any authorization data
      - ...channel binding status, and CB cookies' user certs
    - Distributed apps *automatically* supported
    - MAC-based session continuation requires only a key exchange
    - Logout → DELETE session resource
- Support in HTTP/TLS stacks **NOT NEEDED**

RESTauth

- Got a web server with FCGI? WSGI? Servlets?
  - **Universal server-side deployment** via FCGI
  - You do need an authentication mechanism:
    - Mozilla BrowserID fits (and federates, and scales), ZKPPs fit, GSS, SASL, Kerberos, RADIUS – all fit (enterprises need this)
- The app doesn't need to know the details
  - Server-side: Authen. deployed as servlet/fcgi/whatever
    - The app just GETs the session URI referenced by client
  - Client-side: All the client does is POST authentication messages until authentication completes – the interface to the mechanism can be trivial

RESTauth – when it doesn't work

- If you have an application that only does discrete transactions, each requiring authen. from scratch
  - e.g., payments
- When your app framework forces you to use HTTP authentication (or in TLS)
- When your APIs assume half round trip auth (bearer tokens)
  - But you can fudge these: have the service issue bearer tokens for itself

Universal deployment: FCGI and friends

- FCGI is just a standard interface for writing HTTP applications. A clunky interface, but universally available
- WSGI, Servlets, … are similar, but less widely available
- Author server-side RESTful authentication in FCGI → works for nearly all servers.