

# PKI— Deployment and Application Issues

A Public Key Infrastructure for  
Advanced Network Technologies Workshop  
April 27–28, 2000, NIST

**James A. Rome**

*Executive Secretary, IST*

*Center for Information Infrastructure Technology*

DOE Y12, Advanced Technology Directorate

Oak Ridge, Tennessee 37830-8027

(865) 574-1306      jar@y12.doe.gov

<http://www.ornl.gov/~jar>





# Authorization is what counts

---

PKI can provide strong authentication, but only the owners of resources can authorize their use

- How do you use certificates in the authorization process?
- Do you know which *John Smith* it is?
- Can you guarantee that stakeholder rights are enforced?
- Is there an audit trail for legal action in case of criminal activity?
- Who determines and maintains the security policies?



# Web applications are “easy”

---

- Server certificates allow
  - ◆ Encrypted data via SSL
  - ◆ “Assurance” of the server ID
- Client certificates allow
  - ◆ Strong client authentication
  - ◆ S/Mime encrypted and/or signed e-mail
- Object signing certificates allow
  - ◆ Assurance of Java applets, plug-ins, trusted code
- COTS certificate issuance and management

<http://mmc.ciit.y12.doe.gov/jar/MMCCerts.html>



# Client certificates stored in browsers

---

- Hard to use on someone else's computer  
(you cannot put the certificate on a floppy disk and use it directly)
- Only the latest browsers can manage certificates
- It is very difficult to create Web applications that can access the client certificate DN directly, so that you can use it to implement policy decisions
  - ◆ The usual APIs expect access via LDAP servers
- It is almost impossible to allow a user to access his private key outside of the browser



# Java servlet code to get DN

```
public void doGet(HttpServletRequest servReq, HttpServletResponse servRes) throws IOException
{
    ServletOutputStream out = servRes.getOutputStream();
    String clientCert = "-----BEGIN CERTIFICATE-----\n" + servReq.getHeader("auth-cert") +
        "\n-----END CERTIFICATE-----\n"; // Newlines are critical!!!
    byte[] bytes = clientCert.getBytes();
    try {
        CertificateFactory certFactory = CertificateFactory.getInstance("X.509");
        InputStream certStream = new ByteArrayInputStream(bytes);
        X509Certificate certificate =
            (X509Certificate)certFactory.generateCertificate(certStream);
        Principal dn = certificate.getSubjectDN();
        /* Do things with the contents of the DN here */
        out.println("<B>Certificate DN=</B>" + dn.getName());
    }
    catch ( CertificateException ex ){
        out.println( "Certificate exception " + ex );
    }
}
```

```
Certificate DN=EmailAddress=jar@y12.doe.gov,
CN=James A. Rome,
OID.0.9.2342.19200300.100.1.1=jar, L="Oak
Ridge, TN", ST=Administrator, OU=Center for
Information Infrastructure Technology,
O=Materials Microcharacterization Collaboratory,
C=US
```

But this is a server- and language



# The Tower of Babel rules

---

- Numerous certificate formats:
  - ◆ PKCS #7, PEM, Base 64, DER
- Different signature types:
  - ◆ DSA, RSA
- Crypto interfaces
  - ◆ PKCS #11
- There are certificate bags:
  - ◆ PKCS #12 (<http://www.rsasecurity.com/rsalabs/pkcs/pkcs-12/>)
- There are many security providers:
  - ◆ Sun (<http://java.sun.com/products/jsse/>)
  - ◆ IAIK (<http://jcewww.iaik.tu-graz.ac.at/>)
  - ◆ Cryptix (<http://www.cryptix.org/>)
  - ◆ Fortezza (<http://www.armadillo.huntsville.al.us./software/>)
  - ◆ Entrust . . .



# Security needed in many applications

---

- Java servlets, applets and applications
- C/C++ number crunching programs
- Perl
- Database-enabled connectivity
- CORBA

These all have hooks for (or can be hooked into) PKI-based security services

- What certificates can they use? (X.509, which CAs)
- How does the user access his certificate and private key?

**Security solutions must work on all consumer platforms**



# Security policies

---

## Security policies can be hard to define and implement

- Do permissions have to be verified in real-time? A revoked VISA card can spend a lot of money in a few seconds.
- Access control lists (ACLs) seem impossible to maintain.
- Role-based access control (RBAC) may not be fine-grained enough.

### Complicated policy type examples:

- Two out of three VP signatures to write a check for over \$10k
- Access only during business hours
- Decisions inside an executable according to who is running it
- Length of access before reauthorization
- Ability to spawn jobs (Kerberos has it, PKI does not)
- The lifetime of a policy or a permission
- Is permission to do something also permission to grant it? The reverse seems to be true.



# Stakeholder Requirements

---

- We must obey regulations imposed by government, universities, companies
  - ◆ Sometimes we have to prove it!
- Who owns the information and resources?
- Who is responsible for maintaining these requirements and permissions?

The standard Unix *r,w,x* permissions for *owner*, *group*, and *world* are too simplistic to express these constraints. ACLs are unmaintainable.



# Secure authorization

---

- For simple applications, strong authentication of the user might suffice.
- But in real life, various stakeholders have control over access to resources and data.
  - ◆ Access can only be allowed after approval by each stakeholder
- The Akenti access control system (William Johnston — LBNL, NASA Ames) can solve this need.

<http://www-itg.lbl.gov/security/Akenti/>



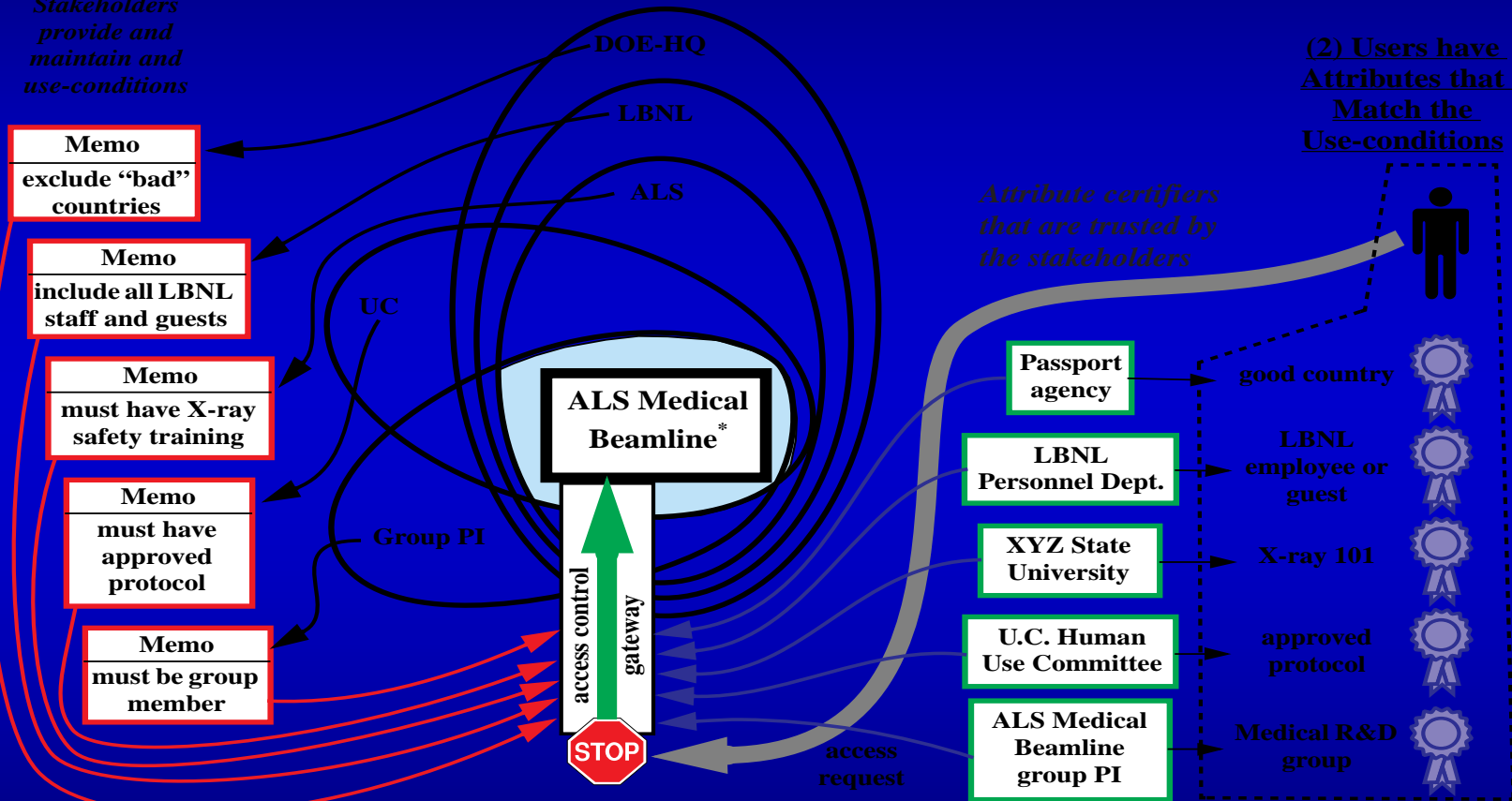
# An example of authorization\*

## (1) Use-conditions are Imposed by Independent Stakeholders

Stakeholders provide and maintain and use-conditions

## (2) Users have Attributes that Match the Use-conditions

Attribute certifiers that are trusted by the stakeholders



## (3) Access is Granted after Verifying that User Attributes Match the Required Use-Conditions

\* hypothetical

## 1 - Societal Access Control Model



## Authorization in “real life”

---

- Probably, the user is given one document attesting to his satisfaction of requirements. E.g., DOE badge allows entrance to facility.
- The access control enforcer — a door guard, the experiment PI, etc. — validates the capability (e.g., checks the badge) when access is requested.

Akenti implements this model in cyberspace.



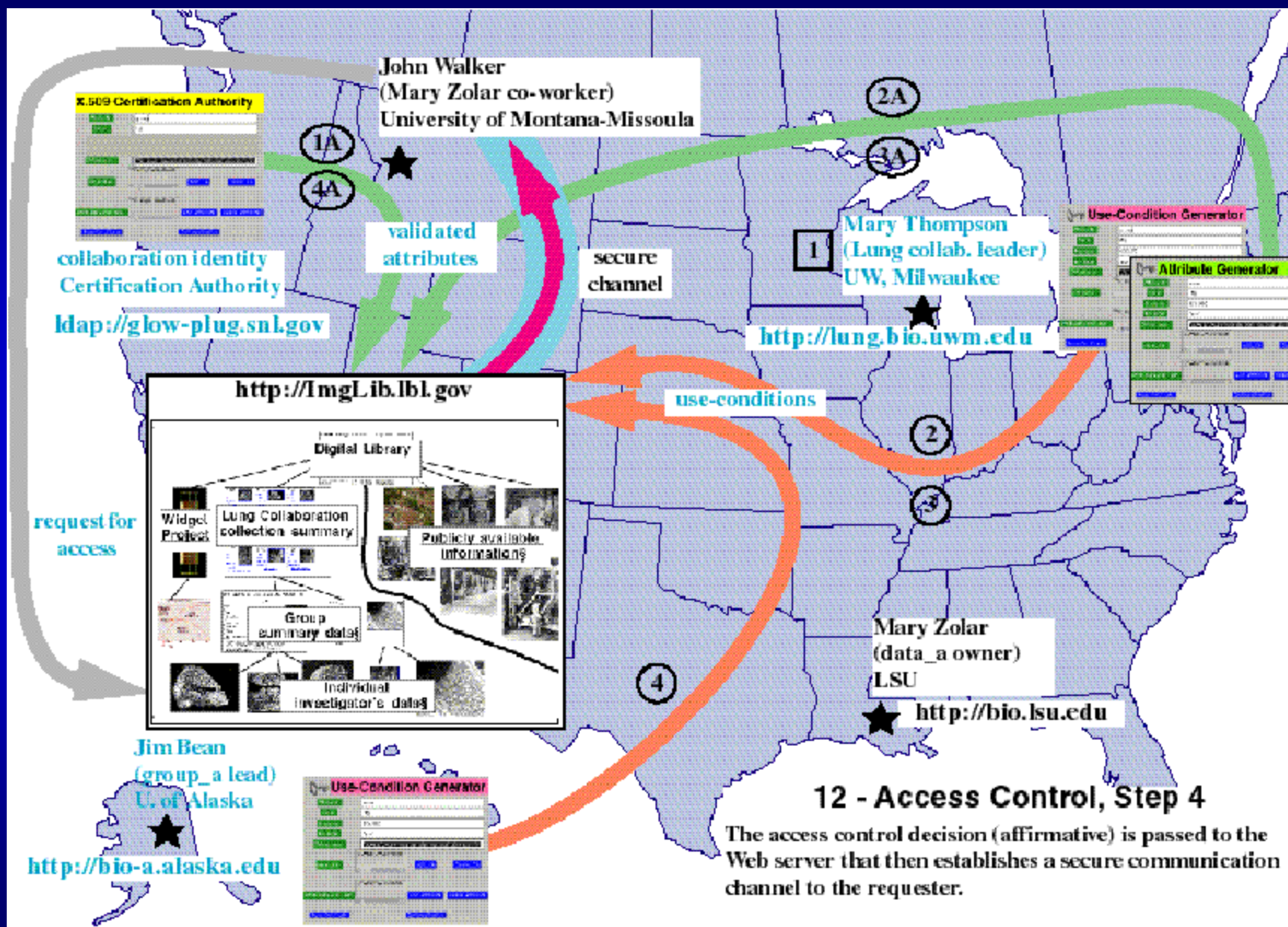
# Akenti reflects current practice

---

- Stakeholders independently make assertions about resource use
- Trusted third-parties certify user attributes required for the use conditions
- Authenticated users that possess the required attributes easily gain access

More details available at:

<http://www-itg.lbl.gov/security/Akenti>





# Akenti distributed policy checks

**Access Viewer**

Progress Panel

Main Resource

Resource: <http://imglib.lbl.gov/imgLi...>

Collection policy elements:

Verifying policy elements:

Verifying access:

Capabilities:

previous next

Signed by:

**Log**

VERIFY\_NAME

Verifying the name of the user and the issuer from a trusted directory sen  
Subject=/C=US/O=Lawrence Berkeley National Laboratory/OU=ICSD/CN=

**Log**

VERIFY\_POLICY

Verifying policy in the following directory  
Dirname=/home/imglib3/http.imglib/akenti-docs/imgLib/COLLECTIONS/

**Log**

VERIFY\_UC\_SIGNATURE

Verifying the signature of the UseConditionIssuer for the following use co  
UC\_uid=rocky.lbl.gov#157a79e2#Wed Jun 10 22:14:12 PDT 1998

Signed by:

**Log**

CACHE\_LOOKUP

looking up a certificate in the cache  
type=2, resource=/C=US/O=Lawrence Berkeley National Laboratory/OU=

Signed by:

**Log**

CACHE\_REQUEST

cache hit  
resource=/C=US/O=Lawrence Berkeley National Laboratory/OU=ICSD/CN=  
version v1.1  
oid lbl.gov  
notValidBefore 000425115343Z  
notValidAfter 000425122343Z  
-----BEGIN CERTIFICATE-----  
MIICdDCCAd2gAwIBAgIBQDANBgkqhkiG9w0BAQQFADBeMQswCQYDVQ  
MCwGA1UEChMITGF3cmVuY2UgQmVya2VsZSxkgTmF0aW9uYWwgTGFI  
MAAsGA1UECzMESUNTRDEQMA4GA1UEAxMH8URDRy1DQTAEfW05OC  
Fw0wMDA1MTUwMDIzNTNaMGoxCzAJBgNVBAYTAiVMS4wLAYDVQQLK  
ZSBBCZXRjZWxleSB0YXRpb25hbCBMYWJvcnF0b3J5M0Q0wCwYDVQQL  
GgYDVQQDEXNNYXJ5J5IFlulFRob21wc29uLXNhMIGfMA0GCsGqSb3DQE  
ADCBiQKBgQCuh4HZei8VKNptQ54RA7PDQuDxE+aXOmNFlu1TH21qT  
AK6aq9gZ27XoKDLWmfWi01ekQCFyr9T5p8zm3ngqA7/6vsWJtMDhb/BX

**Log**

VERIFY\_POLICY

This policy has been successfully verified  
Dirname=/home/imglib3/http.imglib/akenti-docs/imgLib/COLLECTIONS/

Signed by:

**Log**

VERIFY\_ALL\_POLICIES

All policies have been successfully verified

Signed by:



# Certificate requirements

---

- Fast access to certificate servers
  - ◆ Certificates must be checked
- Policy engines must check authorization
- Reliability. If the servers are not up, the user is denied access.
- What happens on a laptop with no networking?

There can be a significant amount of overhead to set up a circuit for a short transaction.

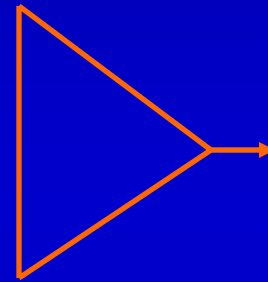


# Akenti summary

---

Certificates can be used to express and enforce complicated and flexible security policies.

- X.509 identity certificates
- User attribute certificate
- Use-condition certificates



authorization  
certificate

Akenti is just now in pilot phase. More details are available from

**William (Bill) Johnston** [johnston@george.lbl.gov](mailto:johnston@george.lbl.gov)



# Good News Things are changing

---

Tools are just being made available to use certificates in end-user applications

- Java JSSE (security extensions) toolkit (9/99)
- Entrust toolkits are now free.
- Enhydra ([www.enhydra.org](http://www.enhydra.org)) open source servlet toolkit (even includes a Web server!)
- Borland's JBuilder 3.5 is available for free
- Akenti (DOE-developed distributed authorization)
- Mozilla security tools (with source code) (<http://www.mozilla.org/projects/security/pki/nss/tools/>)
- JRun servlet environment (<http://www.allaire.com>)