Grid Security

DOE CSG Training 2003

Baltimore, MD

Bob Cowles

bob.cowles@stanford.edu

Dane Skow

dane@fnal.gov

Roadmap

- Crypto
  - Intro to asymmetric encryption and digital signatures
  - PKI & SSL
- Grid security based on Globus tools
- Authentication services (MyProxy, VSC, KCA)
- Authorization services (SAZ, CAS Akenti)
- For today's slides:

  http://www.slac.stanford.edu/~rdc/doecsg03-gridsec.ppt

Symmetric Encryption

- Communicating parties share a secret (key)
- If Alice sends encrypted message to Bob, they must agree in advance on a key
- If also Jules, then Jules ...
  - Has a different key and gets different ciphertext
    - Complicates key distribution, bad policy
  - Has same key and gets same ciphertext
    - Can read other messages between Alice and Bob

Symmetric Encryption (2)

- Third party distribution of keys became the weakest link in the system
- Has been a severe problem with all the ciphers discussed so far
  - Consider submarines submerged for extended periods
  - Consider agents behind enemy lines

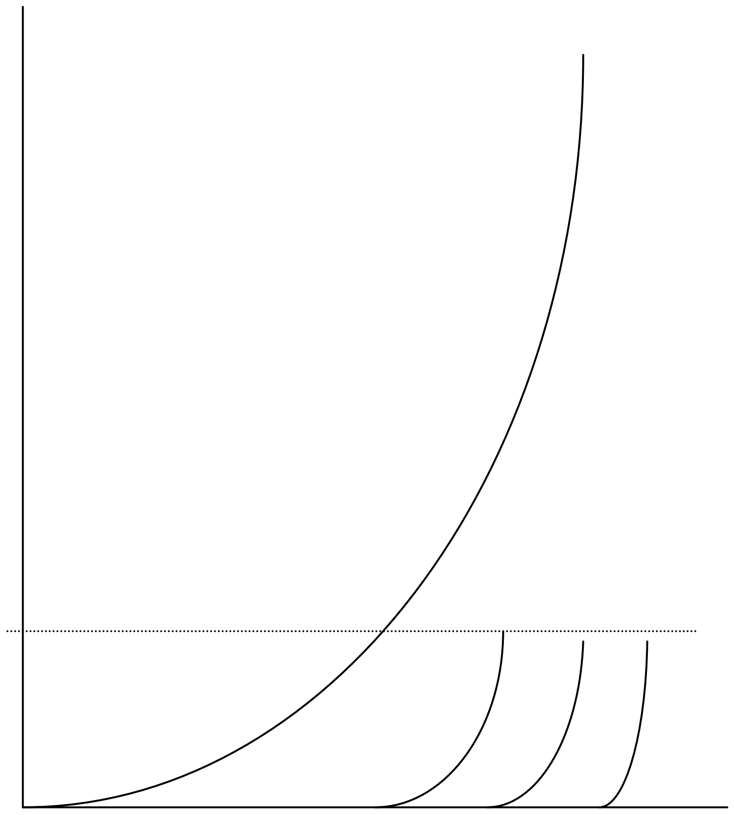
Lockbox Problem

- Alice wants to send a box to Bob without anyone being able to open it
- She padlocks it and sends to Bob
- Bob padlocks it and sends to Alice
- Alice removes her lock and sends to Bob
- Bob removes his lock and opens box
- Success! ?? Really?

One Way Functions

- Also called trapdoor functions
- Like mixing paint, or breaking an egg
- Modular arithmetic is popular form
- A mod B is remainder of A / B
- What time will it be in 42 hours?

One Way Functions (2)



- Consider Y^x mod P as a function
- Computationally very expensive to reverse
- Alice and Bob agree (publicly), Y=5, P=7
- 5^x mod 7

Diffie-Hellman Key Exchange

- Alice picks secret number, say x=2
- Y^x mod P= 5^2 mod 7
- 5*5 mod 7 = 4
- Send a=4 to Bob
- b*b mod 7 = 1
- Shared secret! -- can be used as a key

- Bob picks secret number, say x=3
- Y^x mod P= 5^3 mod 7
- 5*5*5 mod 7 = 6
- Send b=6 to Alice
- a*a*a mod 7 = 1
- Shared secret! -- can be used as a key

Diffie-Hellman Key Exchange

- Eve intercepts values used for Y and P but can't use them to deduce x in a simple way.
- Developed and first publicly demonstrated in 1976
- Alice and Bob no longer have to meet or trust a 3rd party for key exchange
- Still inconvenient -- "real-time" exchanges to establish a key

Lockbox Problem (2)

- Consider a special, universal lock for which you can get pairs of keys
- If locked with key A only unlocks with B and vice versa
- Bob makes copies of key A and distributes them -- send him anything locked with key A and only he can open it with his key B
- Interesting, but how do we build the lock?

Asymmetric Encryption

- A different key is required to encrypt than is required to decrypt
- Key distribution problem is eased -- public key distributed far and wide
- No “real-time” exchange issues
- Diffie at Stanford develops general concept (1975) but could not find a function

RSA

- Rivest, Shamir, Adleman, MIT, 1977
- Discovered a function that worked
  - One way based on the difficulty of factoring numbers into prime numbers (Prime number - evenly divisible by itself and 1)
  - Public key is product of two large primes, N
  - Decryption only possible if prime factors are known, p, q

RSA (2)

- Preliminaries
- Alice picks 2 large primes, p and q (17 and 11 are good for our purposes)
- The product, N is 187.
- Choose e such that e is relatively prime with ((p-1)*(q-1)), say 7
- Alice’s public key is N, e -- 187, 7

RSA (3)

- Bob has counted the ways he loves Alice and wishes to send her the number, 42
- Formula for ciphertext C from message M

$$C = M^e \bmod N$$

$$C = 42^7 \bmod 187 =$$

$$230539333248 \bmod 187 = 15$$

RSA (4)

- Alice must calculate special number d

  $e * d = 1 \bmod ((p-1) * (q-1))$

  $7 * d = 1 \bmod 160$

  $d = 23$

- To decrypt the message

  $M = C^d \bmod 187 = 15^{23} \bmod 187$

  $M = 15*38*135*102 \bmod 187 = 42$

Digital Signature

- Encrypt with private key
- Can only decrypt with public key
- Anyone can verify that you “signed” the document since only you know the corresponding private key.
- Signed and encrypted messages
  - Encrypt with your private key
  - Encrypt with recipient's public key
  - Alternative with less overhead

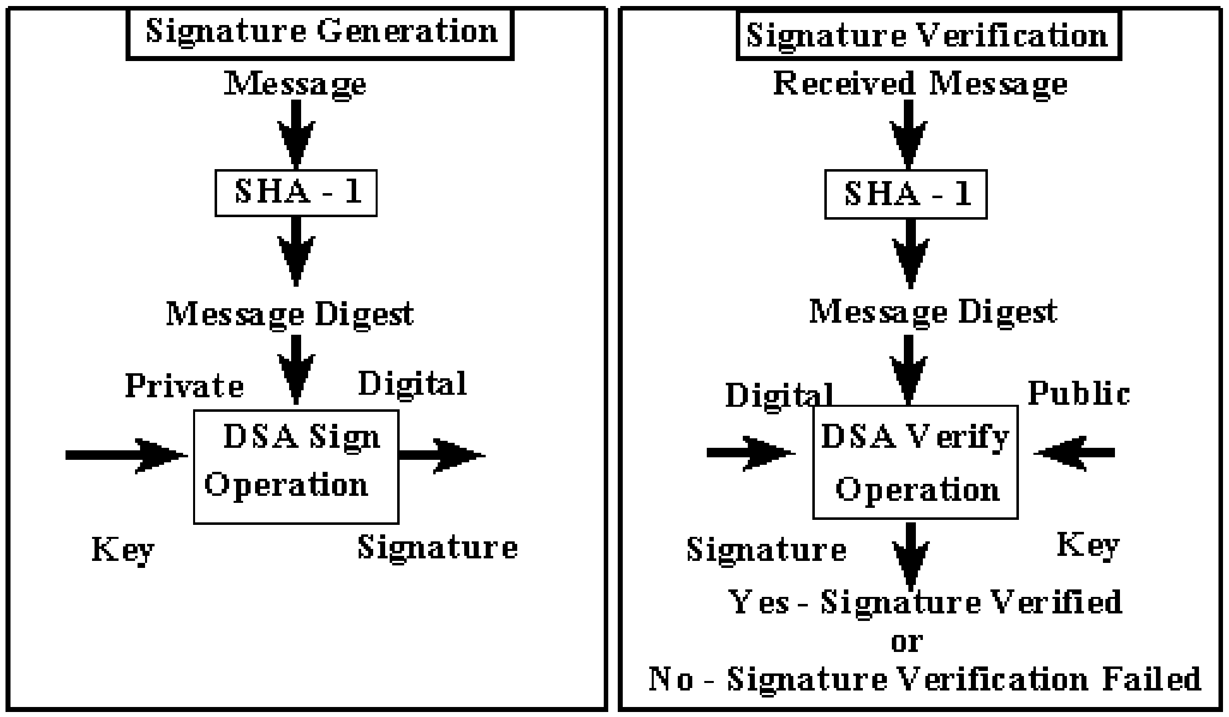
One Way Functions

- Use function to “compress” a message M into a fixed size string F(M) – “hash”
- Goal – infeasible given M to find $M \neq$ M where F(M) = F(M)
- Note: “infeasible” not “impossible”
- Useful for passwords, signatures, message integrity, …

Digital Signature with Hash

- Compute hash value of a message
- Use private key to encrypt (sign) the hash
- Much faster than signing the whole message
- Common hash functions are iterated
  - MD4, MD5, SHA1
  - Init state + input -> *f* -> Init state …
  - *f* is some non-linear function

SHA-1 with Digital Signature



Certificates

- Back to key distribution problem
- Who certifies Bob’s public key is Bob’s?
- Certificate Authority (CA) generates key pair, and digitally signs the public key (plus additional info) making it a Certificate
- CA responsible for establishing identity

X.509 Certificate Contents

- Certificate version, serial number
- Certificate authority (X.500 name)
- Validity period (start, stop)
- Subject (X.500 name)
- Subject public key information
- Extension fields (v3)
- Certificate authority digital signature

Certificate Binding

- Linkage between public key & identity of a:
  - Person, organization, or device (“subject”)
  - Associated with use of private key
  - Used by a “relying party”
- Person linked to identity by either
  - Certificate Authority, or
  - Registration Authority (optional)

PKI

- Public Key Infrastructure
- Chain of CA’s going back to trusted root of Certificate Authorities
- Revocation servers to maintain Certificate Revocation Lists (CRL).
- Many standards, no real infrastructure in US

Hierarchal PKI

- Names guaranteed to be unique globally
- Minimize self-signed keys (entities that you trust because you have to)
- Global namespace – political/technical problems
- High cost for full implementation

PKI Evolution

- **Diffie-Hellman** scheme was to use central phonebook to save/distribute public keys
- **Kohnfelder** felt central phonebook was bottleneck, proposed central authority to sign [name, key] entry and let them loose on the Internet

X.509 CA & RA Roles

- CA
  - Key pair handling
  - Authenticate requests to generate certificate
  - Create, store & revoke certificates
- Nothing inherent in X.509 about key escrow

- RA
  - Identify subscriber
  - Sign and transmit certificate request
  - Return certificate to subscriber
- CA might not have private key except for key recovery

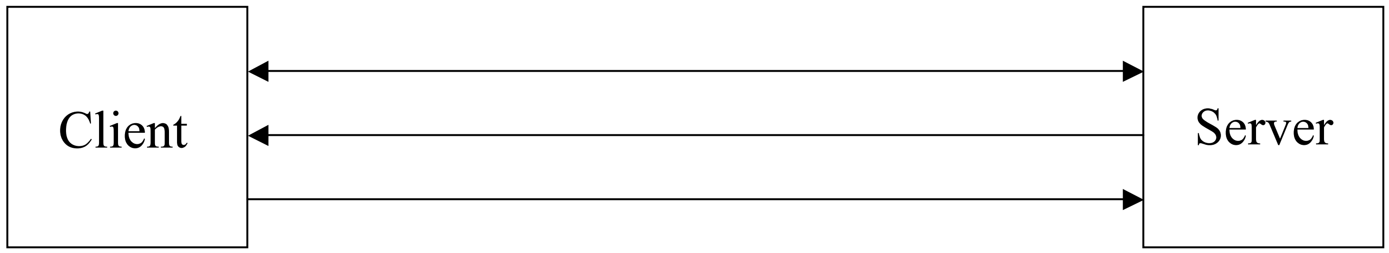
X.509 Certificate Revocation

- CRL lists revoked certificates
- Problems
  - Freshness / frequency / bandwidth
  - Size of CRL over time
  - Replay attacks
- Solution
  - On-line Certificate Status Protocol (OCSP)
  - Revocation can never be offline
- Cost of issuance + cost of revocation = constant

SSL

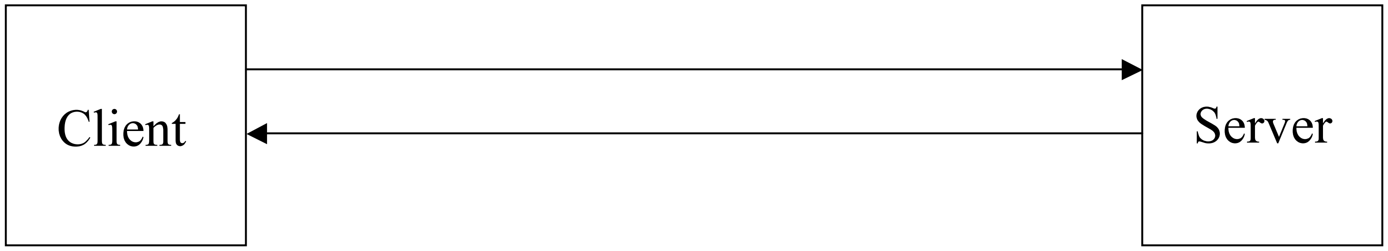
- The problem
  - Client must authenticate server
  - (Server can optionally authenticate client)

SSL – Session Start



1. Exchange random numbers
2. Certificate$_{server}$, session_ID
3. (premaster_secret) PubK$_{server}$

SSL – Connection Start



1. Old session_ID, New random_number$_{server}$
2. New random_number$_{client}$

SSL – Finish Handshake

- Generate keys
  - `master_secret =`
    `MD5/SHA (pre_master_secret, random_numbers)`
  - Encryption key defined from bits in `master_secret`
- Send first message; if not correct, abort

SSL – Naming

- Based on DNS names
- DNS must match name in certificate
  - Prevents man-in-the-middle attacks
  - Protection can be overridden by user
  - Still subject to spoofing attacks
- No CRL is defined in specification

Grid Vision

- Researcher authenticates using PKI-based application
- Request job to be run
- Scheduler determines where job runs
- Data and computational resources brought together
- Results are stored/returned to researcher

HEP Environment

- HEP experiments use multiple physical sites
- Physical sites host multiple experiments
- Researchers caught in the middle
- For Grid to be successful, account/userid issues must be addressed
  - By sites
  - By experiments

The Statement of the Problem

- Provide trusted authentication and authorization checking across security and trust domains
- Risk model is difficult to determine
  - What are threats and vulnerabilities?
- Protect but not interfere (too much)
  - Balanced to reduce over/underprotection
  - On the edge of chaos …

Infrastructure Requirements

- Experiment (Virtual Organization) needs to provide identification infrastructure (PKI)
- Certificate must provide required info for site to set up accounts
- Experiments and sites must agree on AUP
- Automatic enrollment/account creation
- Accept PKI for local account authorization

GSI

- Uses SSL for authentication and message protection
- Adds features needed for Single-Sign on
  - Proxy Credentials
  - Delegation

Credentials

- Each user has a set of credentials
  - used to prove their identity on the grid
  - X509 certificate and private key
- Private key encrypted with a pass phrase
  - Inconvenient for repeated usage

Obtaining a Certificate

- The program grid-cert-request is used to create a public/private key pair and unsigned certificate in ~/.globus/:
  - usercert_request.pem: Unsigned certificate file
  - userkey.pem: Encrypted private key file
    - Must be readable **only** by the owner
- Receive a signed certificate

  Place in ~/.globus/usercert.pem

Your New Certificate

NTP is highly recommended

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 28 (0x1c)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=US, O=Globus, CN=Globus Certification Authority
        Validity
            Not Before: Apr 22 19:21:50 1998 GMT
            Not After : Apr 22 19:21:50 1999 GMT
        Subject: C=US, O=Globus, O=NACI, OU=SDSC, CN=Richard Frost
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:bf:4c:9b:ae:51:e5:ad:ac:54:4f:12:52:3a:69:
                    <snip>
                    b4:e1:54:e7:87:57:b7:d0:61
                Exponent: 65537 (0x10001)
Signature Algorithm: md5WithRSAEncryption
        59:86:6e:df:dd:94:5d:26:f5:23:c1:89:83:8e:3c:97:fc:d8:
        <snip>
        8d:cd:7c:7e:49:68:15:7e:5f:24:23:54:ca:a2:27:f1:35:17:
```

Certificate and Key Data

Sample usercert.pem:

```
-----BEGIN CERTIFICATE-----
MIICAzCCAWygAwIBAgIBCDANBgkqhkiG9w0BAQQFADBHMQswCQY
       <snip>
u5tX5R1m7LrBeI3dFMviJudlihloXfJ2BduIg7XOKk5g3JmgauK4
-----END CERTIFICATE-----
```

Sample userkey.pem:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,1E924694DBA7D9D1
+W4FEPdn/oYntAJPw2tfmrGZ82FH611o1gtvjSKH79wdFxzKhnz474Ijo5Bl
       <snip>
et5QnJ6hAO4Bhya1XkWyKHTPs/2tIflKn0BNIIIYM+s=
-----END RSA PRIVATE KEY-----
```

“Logging on” to the Grid

- Authenticate:

```
% grid-proxy-init
Enter PEM pass phrase: ******
```

- Creates temporary, short-lived proxy credential
- Options for grid-proxy-init:

```
-hours <lifetime of credential>
-bits <length of key>
-help
```

Single Sign-on

- To support single sign-on GSI adds the following functionality to SSL:
  - Proxy credentials
  - Credential delegation

Proxy Credentials

- Proxy credentials are short-lived credentials created by user
  - Short term binding of user's identity to alternate private key
  - Stored unencrypted for easy repeated access
  - Short lifetime in case of theft
  - Enables user to authenticate once then perform multiple actions without reauthenticating

grid-proxy-init Details

- grid-proxy-init creates the local proxy file.
- Passphrase, used to decrypt private key.
- Private key is used to sign a proxy certificate with its own, new public/private key pair.
  - User's private key not exposed after proxy has been signed
- Proxy placed in /tmp, read-only by user
- NOTE: *No* network traffic!
- grid-proxy-info displays proxy details

Important Files

- /etc/grid-security
  - hostcert.pem: used by the server in mutual authentication
  - hostkey.pem: private key corresponding to the server’s certificate (read-only by root)
  - grid-mapfile: maps grid subject names to local user accounts (really part of gatekeeper)
- /etc/grid-security/certificates
  - CA certificates: trusted certs. so needn’t be verified
  - ca-signing-policy.conf: defines the subject names that can be signed by each CA

Important Files (2)

- $HOME/.globus
  - usercert.pem: User’s certificate (subject name, public key, CA signature)
  - userkey.pem: User’s private key (encrypted using the user’s pass phrase)
- /tmp
  - Proxy file(s): Temporary file(s) containing unencrypted proxy private key and certificate (readable only by user’s account)
    - Same approach Kerberos uses for protecting tickets

Secure Services

- On most unix machines, inetd listens for incoming service connections and passes connections to daemons for processing.
- On Grid servers, the gatekeeper securely performs the same function for many services
  - It handles mutual authentication using files in /etc/grid-security
  - It maps to local users via the gridmap file

Sample Gridmap File

- Gridmap file maintained by Globus administrator
- Entry maps Grid-id into local user name(s)

```
# Distinguished name                                    Local
#                                                     username
"/C=US/O=Globus/O=NPACI/OU=SDSC/CN=Rich Gallup”         rpg
"/C=US/O=Globus/O=NPACI/OU=SDSC/CN=Richard Frost”      frost
"/C=US/O=Globus/O=USC/OU=ISI/CN=Carl Kesselman”        u14543
"/C=US/O=Globus/O=ANL/OU=MCS/CN=Ian Foster”            itf
```

Simple job submission

- globus-job-run provides a simple RSH compatible interface

```
% grid-proxy-init
  Enter PEM pass phrase: *****
% globus-job-run host program
[args]
```

- Job submission will be covered in more detail later

Delegation

- GSI enables user to create and delegate proxy credentials to processes running on remote resources
- Allows remote processes and resources to act on user’s behalf
- Important for complex applications that need to use Grid resources
  – E.g. jobs that needs to access data storage

Delegation II

- Delegation = remote creation of a (second level) proxy credential
  - New key pair generated remotely on server
  - Proxy cert and public key sent to client
  - Clients signs proxy cert and returns it
  - Server (usually) puts proxy in /tmp
- Allows remote process to authenticate on behalf of the user
  - Remote process "impersonates" the user

Limited Proxy

- During delegation, the client can elect to delegate only a “limited proxy”, rather than a “full” proxy
  - GRAM (job submission) client does this
- Each service decides whether it will allow authentication with a limited proxy
  - Job manager service requires a full proxy
  - GridFTP server allows either full or limited proxy to be used

Restricted Proxies

- A generalization of the simple limited proxies
  - Desirable to have fine-grained restrictions
  - Reduces exposure from compromised proxies
- Embed restriction policy in proxy cert
  - Policy is evaluated by resource upon proxy use
  - Reduces rights available to the proxy to a subset of those held by the user
    - A proxy no longer grants full impersonation rights
  - Extensible to support any policy language
- Will be in future version > GT 2.0

X.509 Difficult to Secure

- Secure private keys and users don't mix
  - No guarantee of good or any password choice
    - In fact, many users don't *want* password on their keys
  - No guarantee of secure private key location
    - E.g., users store keys in network based file systems
  - No guarantee how private key was handled
    - E.g., users copy/e-mail keys to remote machines & leave them
- User managed keys *should not* be trusted

Today’s Solutions

- Protect Long-Term Certificate
  - Use proxy-certs to limit key exposure damage
    - Grid-proxy-init
- Make x.509 cert handling convenient
  - Limit avenues for user error
    - SACRED, MyProxy
- Protect Identity Cert and Make it Easier
  - KCA, Smart Cards, VSC

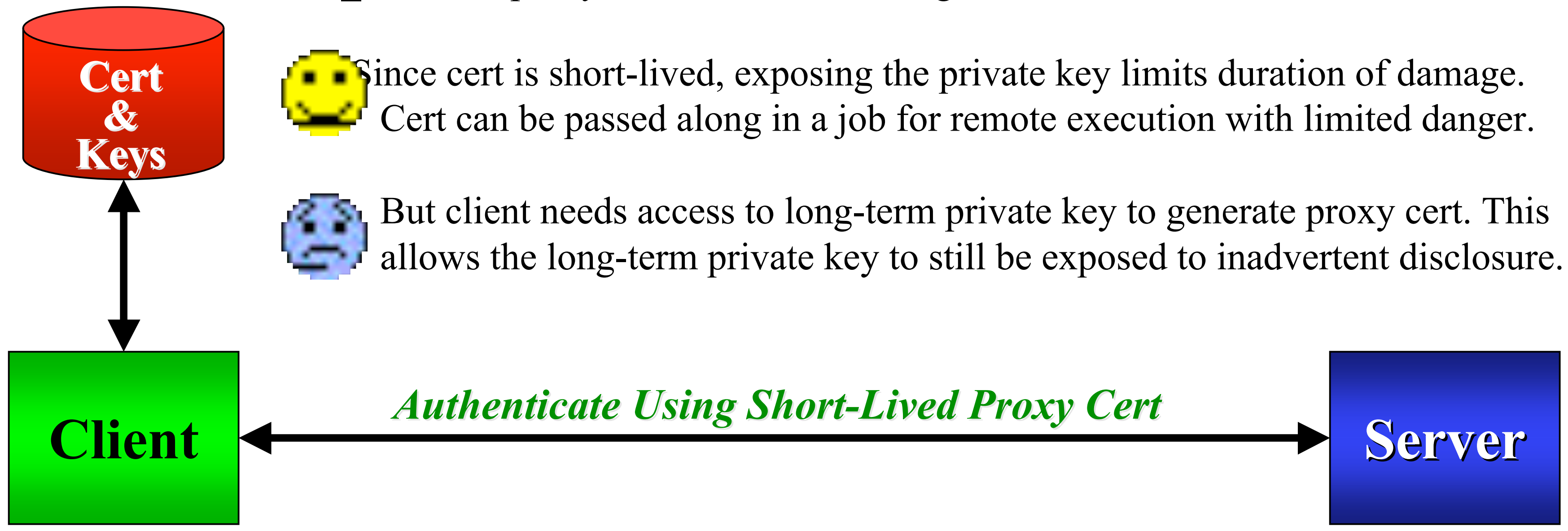
Globus grid-proxy-init

Proxy Cert Steps:

_ Client generates a new public/private key pair.
_ Uses it to construct a new short-lived cert. This is called a proxy cert and is distinguished by the addition of /CN=proxy to the User's name.
_ Signs the new cert with the long-term cert's private key
_ Uses the proxy cert wherever the long-lived cert would be used

Since cert is short-lived, exposing the private key limits duration of damage. Cert can be passed along in a job for remote execution with limited danger.

But client needs access to long-term private key to generate proxy cert. This allows the long-term private key to still be exposed to inadvertent disclosure.

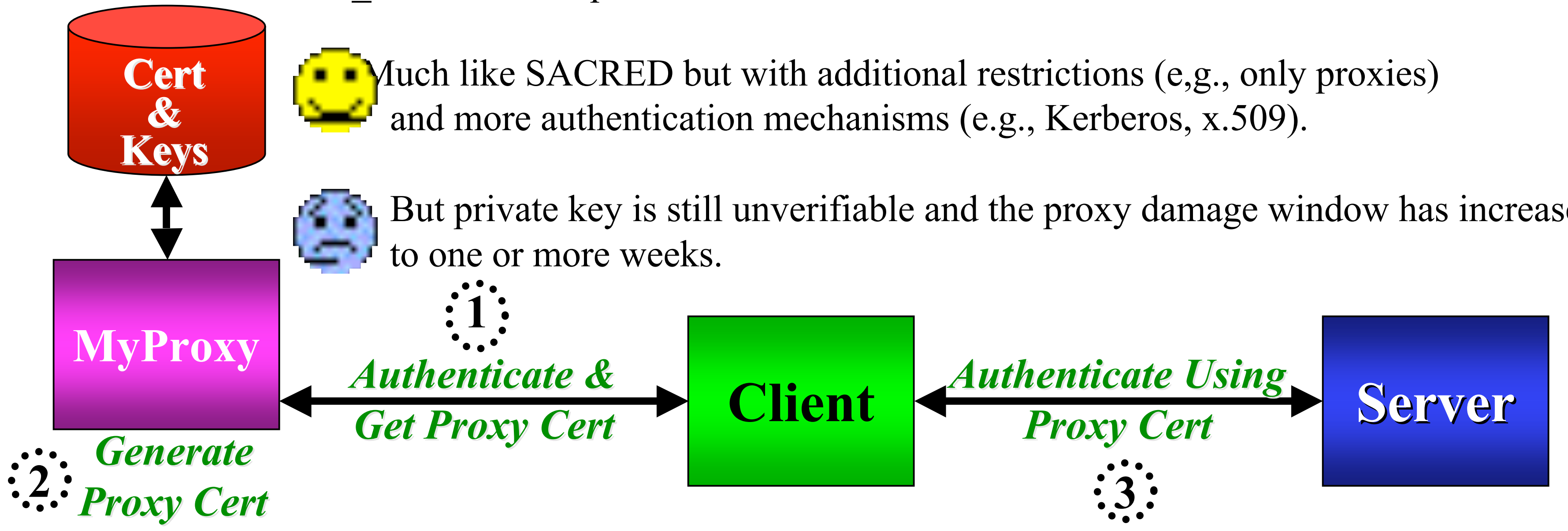


MyProxy

***MyProxy Steps*:**	_ Client contacts an allowable server via special protocol.
myproxy-init	_ Uploads delegated short-lived (e.g., 1 week) proxy credentials associated with an arbitrary userid/password and download restrictions.
myproxy-get-delegation	_ User or service acting in behalf of the user can download a MyProxy generated short-lived proxy cert for use with a server.
	_ Uses account/password to download these elsewhere when needed.

🙂 Much like SACRED but with additional restrictions (e,g., only proxies) and more authentication mechanisms (e.g., Kerberos, x.509).

🙁 But private key is still unverifiable and the proxy damage window has increased to one or more weeks.



KCA (Kerberos Certificate Authority)

KCA Steps:

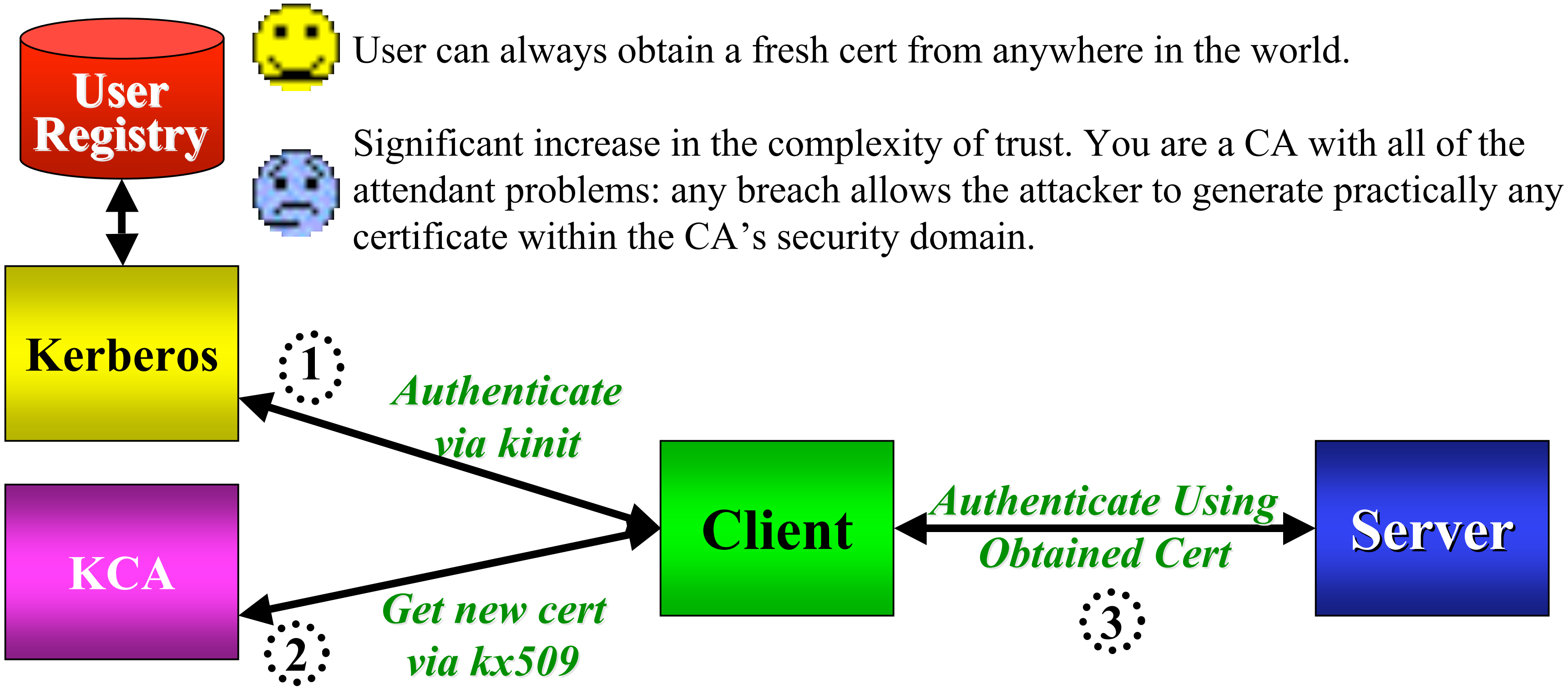
_ User registers with a known organization & gets a Kerberos account.

kinit; kx509 _ Login via Kerberos and get fresh short-lived credentials from a special server,

_ Use obtained certificate anyway you choose.

User can always obtain a fresh cert from anywhere in the world.

Significant increase in the complexity of trust. You are a CA with all of the attendant problems: any breach allows the attacker to generate practically any certificate within the CA's security domain.



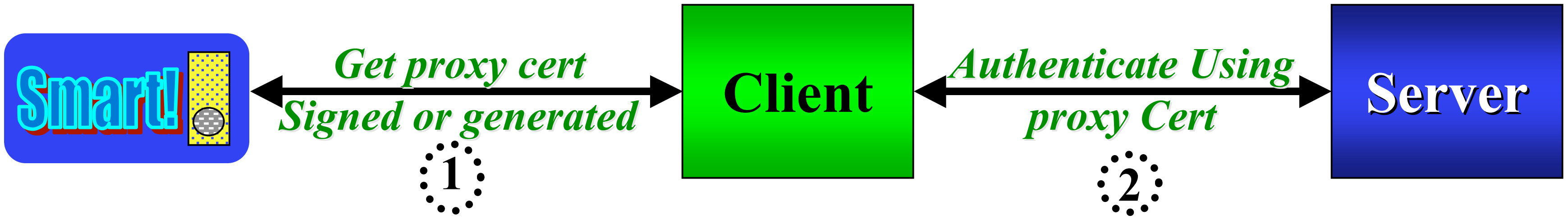
Smart Card

Smart Card Steps:

_ User gets a physical card with a password protected identity cert.
_ User inserts card into a reader, enables it via password, and asks card to either sign a generated proxy cert or generate a signed new one for later use.
_ Use smart card proxy certificate as you would a normal proxy certificate.

🙂 Card is portable so user can obtain a fresh proxy certificate and never see the private key (private key never leaves the card).

🙁 Smart card readers not widely deployed.



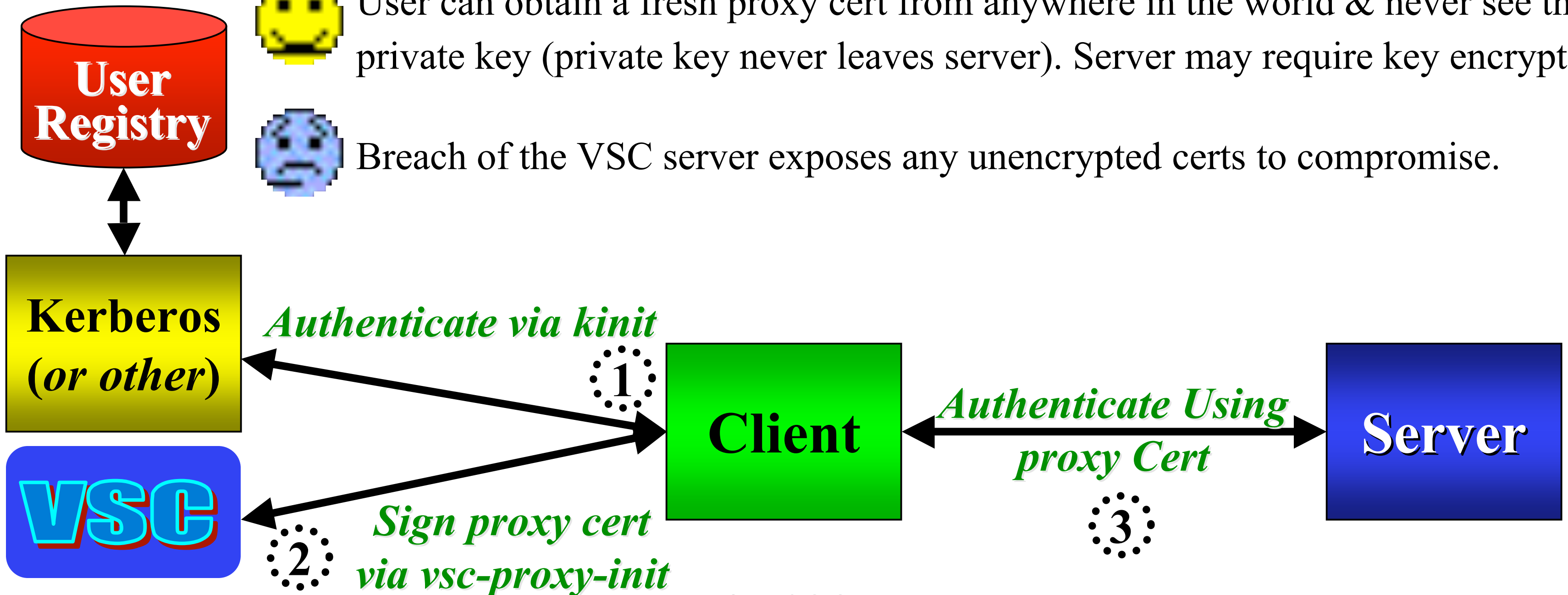
VSC (Virtual Smart Card)

VSC Steps:

- User registers with a known organization & typically gets a Kerberos account.
- User requests the VSC server, only once, to obtain a long-lived cert for them.
- kinit; vsc-proxy-init — Login via Kerberos (or other) and get proxy cert signed by long-term cert.
- Use VSC proxy certificate as you would a normal proxy certificate.

☺ User can obtain a fresh proxy cert from anywhere in the world & never see the private key (private key never leaves server). Server may require key encryption.

☹ Breach of the VSC server exposes any unencrypted certs to compromise.



Software Solution Summary

- Each solution presents its own problems
  - grid-proxy-init
    - Private long term must be available and may be potentially mishandled
  - MyProxy
    - Private long term is available and may be potentially mishandled
  - KCA
    - Private keys never see the wire (no long-term private key) but issuer relies on very strong trust assumptions
  - VSC
    - Private keys are never exposed but long-term keys are concentrated on a secure server

Authentication References

- KCA/x.509
  - http://www.nsf-middleware.org/documentation/NMI-R2/0/KX509KCA/
- MyProxy
  - http://www.ncsa.uiuc.edu/Divisions/ACES/MyProxy/
- Virtual Smart Card
  - http://slac.stanford.edu/~abh/vsc
  - http://www.cs.dartmouth.edu/~pki02/Sandhu/paper.pdf

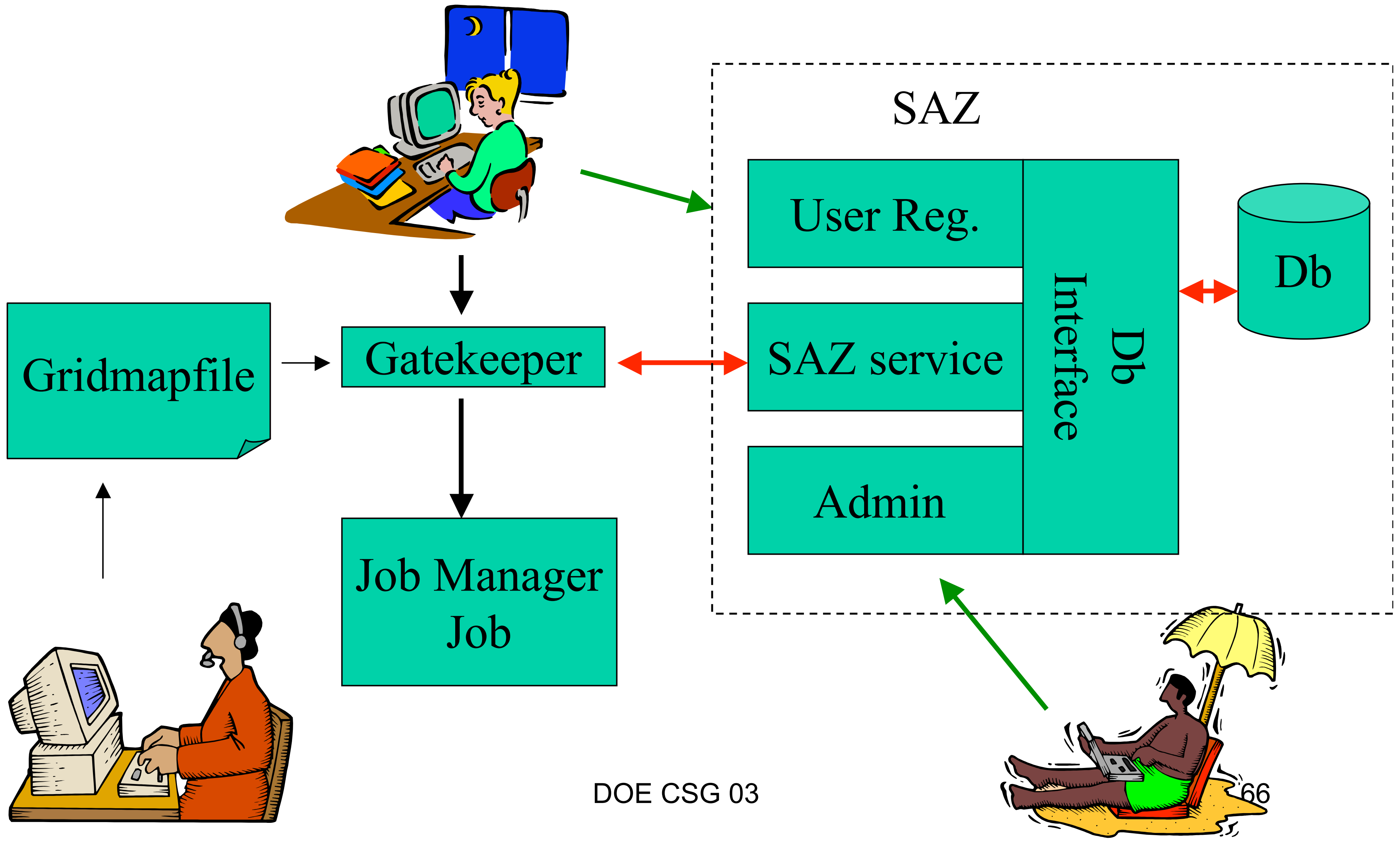
Authorization

- SAZ – Provides a site-level callout for centralizing certain decisions
- CAS – Provides VO-level control over authorizing resources for groups
- Akenti – Provides fine-grained authorization for access to more complex resources

Why is SAZ needed ?

- In the GTK2, authorization is dealt with via the gridmapfile and there is one per service.
- Sites wish to impose sitewide policy centrally
- Can streamline some maintenance (eg. CRL checking)
- Allows for multiple layers of authorization

What does SAZ look like ?



What does SAZ do ?

- The primary purpose for SAZ is to act as an authorization service for all Grid Resources on a site so that common policy can be enforced.
- A secondary purpose is to centralize maintenance of Certificate Revocation Lists (CRLs)
- The current version implements this in the gatekeeper via an LCAS (from the EDG suite of products) shared memory module.

What more does SAZ do ?

- The service needs to get data about which DNs map to the same person.
- The internal SAZ database is populated through two interfaces: user and admin.
  - Users can add or delete their own DN.
    - We use Kerberos authentication for that transaction.
    - They may add several different DNs to their entry
  - Administrators can add or delete any entry in the database.

CAS Policy Management

- Site policies maintained normally (e.g., gridmap files and unix accounts)
- Community policies maintained with CAS
- Community users and groups manage their individual policies

CAS Resource Provider’s View

- Grants access to a block of resources, using existing access-control mechanism (e.g., grid-mapfile entries, file permissions, etc.)
- Use native mechanisms (e.g. quotas) to set additional policy for the community as a whole
- Install servers modified to enforce the policy in the CAS credentials

CAS Community’s View

- Administrative requests used to maintain the CAS community policy database:
  - controls rights the CAS server will grant to users.
  - controls the CAS server’s own access control policies – can delegate granting rights, maintaining groups, etc.
  - maintains the list of community members

CAS Policy Enforcement

- A resource provider may recognize several CAS servers and may accept CAS authorization for some resources but not others
- Policy assertions signed by CAS are examined; providers can implement an additional level of policy enforcement based on user’s identity
- CAS server does not need to be contacted the for each request

Motivation for Fine-grained Authorization using Akenti

- A Virtual Organization (VO) wants to provide limited services to its members
  - Allow most users to only run a small number of services but possibly with high resource limits
  - Allow developers to run a wider range of programs such as compilers or debuggers but with stricter resource limits
  - Administrators may want to monitor jobs and kill misbehaving user jobs

Motivation II

- Users in the grid-mapfile have the equivalent of a login account on the host.
  - No limit on binaries that can be executed.
  - No limit on compute time or disk resources.
  - All fine-grain authorization is done by OS on the basis of the local user id assigned to the job.
- Users can kill or manage their own jobs, but no other party can.

GRAM (Grid Resource Acquisition and Management) modules

- Gatekeeper
  - Does the admission control based on a static grid-mapfile entry
  - Starts the requested service, e.g. job manager
- Job Manager
  - Parses the Resource Specification Language (RSL) that specifies the binary to be executed; Handles requests pertaining to executing jobs
    - Suspend, stop, query

Job Manager authorization

- Does no authorization on job startup
  - Gatekeeper verified that the user has privileges to run on the machine before starting the job manager
- Only allows the initiator of the job to issue job control directives
- Runs with uid of the user so it can only control jobs started by the initial user.

Add authorization callouts from the Job Manager

- Add a generic authorization callout at the points where a job is started
- And when one of the following job managements requests is made
  - Cancel, suspend, resume, ask for status, change priority of job
  - register or deregister a call-back contact
  - stop/restart the job manager process that is watching the job

Akenti Authorization Server

- Authorization policy created by independent stakeholder as digitally signed certificates.
- Requestors are identified by X.509 certificates or DN and CA's DN.
- Resource gateway asks for a authorization decision based on a resource name and the requestor's identity.
- Akenti finds (pulls) all the relevant authorization policy and returns allowed actions and conditional actions.
- Conditional actions may specify runtime conditions that the resource gateway must evaluate.

Akenti Authorization plug-in

- Handles the interface between the Job Manager and the Akenti authorization service.
- Maps Globus resource name e.g. pathname of binary or job tag to an Akenti policy resource name.
- Interprets Akenti response.
  - Evaluates runtime conditions
    - Policy might limit number of CPUs used
  - Maps Akenti actions to Globus actions
    - e.g. “control job” to cancel job, get job status, suspend, resume, etc.
- Returns allowed or disallowed answer to Job Manager and a Globus Error object.

Summary

- GSI is:
  - X.509 Certificates for authentication
  - PKI for verifying identities in Certificates
  - SSL as the protocol for authentication, confidentiality and integrity
  - Proxy certificates and delegation to support single sign-on

GGF Security Working Groups

- http://www.gridforum.org/security/

- Grid Security Infrastructure (GSI) http://www.gridforum.org/security/ggf1_2001-03/drafts/draft-ggf-gsi-roadmap-02.pdf
- Grid Certificate Policy Design

  http://www.gridcp.es.net/

Working Group Documents

- Grid Security Infrastructure (GSI)
- Grid Certificate Policy Design
- Security Implications of Typical Grid Computing Usage Scenarios http://www.gridforum.org/security/gf5_2000-10/drafts/draft-gridforum-security-implications-01.pdf

Grid Deployment

- As with development of the “information grid” (WWW), we have to think about the kinds of resources we make available
- Requirements for security and account administration will play a large role in how grid services are defined
- Site security and systems administration personnel must be involved