

Cacl

A CA System with Automated User Authentication

William J. Link
San Diego Supercomputer Center
University of California, San Diego

September, 2003

1 Introduction and Summary of Operation:

Cacl is an OpenSSL based CA (Certificate Authority) system which was developed at the San Diego Supercomputer Center to speed up and simplify the issuance and use of digital certificates, both for users and for the CA manager. It accomplishes this, in part, by automatically authenticating the identity of a certificate requester, thus reducing the administrative burden of CA management and essentially eliminating delays for the users. In addition, cacl creates for the user the certificate environment needed by Globus software, which helps simplify the task of getting started in grid computing. The system was conceived of, designed, and developed at SDSC by the author and Wayne Schroeder. It was put into production use in 2000 and has been running at SDSC since that time. The Cacl CA system has replaced a Netscape Certificate Management System for use in issuing all client and server certificates at SDSC.

The Cacl CA system is based on a client/server model. The client program, called cacl (Certificate Authority CLient), is run by users logged into machines within the SDSC domain, to request user certificates. The cacl program is the only part of the CA system which is visible to users and consequently the CA system as a whole has come to be known as the Cacl CA system.

When a user runs the cacl program to request a certificate he is prompted for the password used to login to the account from which cacl is being run. Next the user is prompted for a private key encryption password. The user's name, which will be included in the certificate request, is extracted from the user's /etc/passwd entry. The cacl program constructs a PKCS#10 (Public Key Cryptography Standards) certificate request based on a modified OpenSSL configuration file into which user information has been added.

After cacl constructs a certificate request it then encrypts the request using the certificate authority's certificate. This is done in order to protect the

certificate request file which contains the user's login password in the challenge password field of the request. Cacl then opens a socket connection to the cacl system's CA daemon, which is running on a machine dedicated to CA functions, and sends the request to the CA. The CA daemon decrypts the request using its private key and, after determining from which machine within the SDSC domain the request originated, begins the process of authenticating the user.

At SDSC most of the machines from which cacl can be run have their account information recorded in a single file which is managed by an account management system. The information found in the passwd file maintained by this system is what would normally be found in Unix `/etc/passwd` and `/etc/shadow` files. The CA daemon is able to authenticate the user requesting a certificate by checking this passwd file. After matching the login name and the user name from the GECOS field with the information in the certificate request the password is checked. The salt from the user's encrypted password in the passwd file together with the password provided in the certificate request is run through crypt to encrypt the password supplied in the certificate request. The newly encrypted password is then matched against the user account's encrypted password in the account management system's passwd file for the machine from which the certificate request originated. This process authenticates the certificate requester to the same level of certainty that ordinary account login provides. This is appropriate given that the certificate issued by the CA will provide the user with a method of authentication in addition to ordinary password authentication but not confer any authorization above that available through account login.

After the CA daemon has evaluated the certificate request it will either issue a certificate and send it back to the cacl client program through the socket connection or send a rejection notice back to cacl. In either case it usually takes only ten to twenty seconds to establish the socket connection, send the request, evaluate the request, and get a certificate or rejection notice back to the user.

The third component of the CA system is the caad (CA ADministrator) program which an administrator can use to revoke certificates and create server certificates. This program is available only to the administrators of the CA system and must be run on the machine which provides the CA service.

2 CACL, from the user's perspective:

One of the objectives of this certificate authority project was to make the process of requesting a certificate as quick and easy for the user as possible. Furthermore, it was an objective of the project to create for the user a working environment from which the user could run Globus commands. This was done by putting the user's certificate and encrypted private key in a `.globus` directory within the user's home directory together with the certificate and key in PKCS#12 format to facilitate its use in a web browser.

The user runs the cacl program without command line arguments to request a certificate. Cacl prompts the user for the user's login password for the machine on which he is running cacl then asks the user for an encryption password for

the private key, the user is asked a second time for the encryption password to ensure that it has been entered correctly. Below is an example of what a user would see when running cacl successfully to get a digital certificate.

```
% cacl
```

```
Please enter your login password:
```

```
Next you will be prompted to enter a password which will serve as both the encryption key for your certificate's private key and as the 'export password' for your PKCS#12 converted certificate. Please choose a password that you can remember, you will need to use it in the future.
```

```
Please enter your private key encryption password:
```

```
Verifying private key password, please reenter password:
```

```
You now have a .globus directory containing the following files:
```

```
usercert.p12 - Your digital certificate and private key in a PKCS#12 format certificate
```

```
usercert.pem - Your digital certificate, signed by the CA daemon
```

```
userkey.pem - Your encrypted private key matching the public key contained in your usercert.pem
```

```
*****
```

```
YOUR CERTIFICATE IS VALID FOR FOUR YEARS, DO NOT FORGET YOUR PRIVATE KEY ENCRYPTION PASSWORD AND, DO NOT DELETE YOUR .globus DIRECTORY.
```

```
*****
```

```
%
```

The cacl program is a Perl script which uses OpenSSL to create a PKCS#10 certificate request. When the program is first started it checks its environment to make certain that it can find the openssl command in the location it expects. It also checks for the existence of the configuration file used in the certificate request creation process and the certificate authority's certificate. Lastly, it looks to see if a .globus directory already exists within the user's home directory.

If a `.globus` directory does exist it is moved to `.globus.old` and the user is told of the name change. If both a `.globus` and a `.globus.old` exist in the user's home directory the user is told of this and asked to "clean up" their home Globus environment, the `cacl` program then terminates. If all of the environment checks just mentioned are passed then `cacl` proceeds to to gather the information needed to create a certificate request.

The first piece of information `cacl` gets is the account name under which it is being run. `Cacl` then finds the `/etc/passwd` file entry for the user and extracts the user's real name from the GECOS field of the `passwd` file. The user's name from the GECOS field will be put into the CN (Common Name) field in the certificate request and will become the CN entry within the DN (Distinguished Name) of a certificate issued from the request. The user's account name will also be put into a USERID field in the request and will become part of the certificate DN.

`Cacl` now begins to interact with the user by prompting for his login password for the system from which `cacl` is being run. The password will go into the Challenge Password field of the certificate request. This password will allow the CA daemon to authenticate the identity of the user running `cacl` to the same level of certainty as ordinary login password authentication. `Cacl` handles the entered password in a way that will ensure that any special characters within the password will be preserved and not modified or eliminated by Perl or the `openssl` program.

The user is next asked for a private key encryption password. The user is prompted twice for this password and the response to both prompts are compared to make sure that the password was correctly entered. The password is also checked for length, passwords less than eight characters in length are not accepted. If the private key encryption password is not an appropriate length or if the two entries of the password are not the same the `cacl` program will tell the user of the problem and continue to prompt for a password until an acceptable length password has been correctly entered twice.

After the `cacl` program has gathered all of the information needed for the certificate request it creates a `.globus` directory in the user's home directory and does a Perl `chdir` into the `.globus` directory. At this point `cacl` copies the user certificate configuration file to its working directory and edits the file to insert the user's CN, login name, and login password into the configuration file. `Cacl` next creates a pseudo random number seed file and uses it to create the 2048 bit key pair that will be used in the certificate's public and private keys. The key pair generation process, which is part of the process of generating a PKCS#10 certificate request, is the most time consuming operation done by `cacl` and often takes longer than the wait for the certificate authority to evaluate the certificate request. The certificate request is then encrypted using the CA system's certificate so that the request can be securely transmitted to the CA daemon.

`Cacl` next opens a socket connection to the CA daemon running on the certificate authority system and transmits the encrypted certificate request to the daemon. The CA daemon decrypts the request and evaluates it while the

cacl program, and the user running it, wait for a reply. Ordinarily the wait is less than a minute. The CA daemon returns a reply to cacl through the same socket connection through which the certificate request was sent. The first line of the reply from the CA daemon will be either "Success " or "Error ". If the message indicates a success then the certificate request has been accepted and a signed certificate makes up the rest of the reply message. If the message indicates an error has occurred the nature of the problem is communicated to the user. The most common causes for a certificate request to be rejected are that an incorrect login password has been submitted or that a valid certificate, with an identical distinguished name, already exists for the user. In the latter case the user's currently valid certificate must be revoked by the a CA administrator before a new one can be issued. The cacl program terminates when an error message is received from the CA daemon.

If the CA daemon reports success in the certificate request process then the user's new certificate, in PEM (Privacy Enhanced Mail) format, is written into the .globus directory. The certificate and the corresponding private key, which is already in the .globus directory, are then converted into a PKCS#12 format certificate which can be imported into a Netscape browser. The user will then see the message shown above indicating that they now have a .globus directory, telling them what it contains, telling them that the certificate will be valid for four years, and warning them not to forget their private key decryption password or delete the .globus directory. The cacl program terminates when a certificate has been successfully issued and stored.

3 Caad, the Certificate Authority Administration program:

The caad program is a simple Perl script used for creating server or service certificates and revoking certificates. The program can generate a certificate request, using a configuration file which identifies the certificate type as a server certificate, process the request and sign the resulting certificate using the CA's private key. It can take a server certificate request generated externally by a server program, such as a web server, and sign it with the CA's private key. It can also create multiple server or service certificates by reading a file containing the CNs for the certificates to be created. Lastly, caad can be used to revoke a certificate signed by the cacl CA system.

This is what the caad script looks like when it is run to create a server certificate:

```
%% caad
```

1. Do you want to create a server cert?
2. Do you have a server cert request that you want to sign?
3. Do you want batch create multiple server certs?
4. Do you want to revoke a cert?

Please select the number of the option you would like: 1

Please enter the server name: host/aserver.sdsc.edu

1. SDSC
2. UMICH
3. TACC
4. CALTECH

Please pick a number for the NPACI organization: 1

Please enter your private key encryption password:

Verifying private key password, please reenter password:

Using configuration from sconfig

Generating a 2048 bit RSA private key

.....+++

.....+++

writing new private key to 'host-aserver.sdsc.edu.privkey.pem'

Using configuration from sconfig

Check that the request matches the signature

Signature ok

The Subjects Distinguished Name is as follows

countryName :PRINTABLE:'US'

organizationName :PRINTABLE:'NPACI'

ORGANIZATIONALUNITNAME:PRINTABLE:'SDSC'

commonName :PRINTABLE:'host/aserver.sdsc.edu'

Certificate is to be certified until Jun 16 06:06:40 2007 GMT (1461 days)

Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y

Would you like a decrypted copy of the private key? [y/n] n

%%

When option 1 is selected from the above menu the administrator is prompted for the server name for which the certificate is to be issued. The server name should include the full domain name for the machine on which the certificate will be used. The current convention for Globus server and service certificates is to identify the certificate type in the certificate's CN field by the prefix of host/ or ldap/, for example the CN field in certificates for a machine named b80n10.sdsc.edu might look like this:

- host/b80n10.sdsc.edu

- ldap/b80n10.sdsc.edu

It is important that the machine name used in a computer's certificate CN be its primary name and not an alias. The reason for this is that processes using that certificate to authenticate the server will depend upon a match of the server's name returned by DNS lookup with the CN in the certificate. If the CN and the name returned in a DNS lookup do not match the authentication process will fail.

The administrator is next prompted for the NPACI (National Partnership for Advanced Computational Infrastructure) organization for which the certificate will be issued. The acronym for the unit will be put into the OU (Organizational Unit) field of the certificate's DN.

The caad program will then prompt for a private key encryption password in the same way as the cacl program. The encryption password will be prompted for twice, as cacl does. The key pair is then generated, a certificate request is created, and processed. The administrator is then prompted to sign and commit the server certificate. Lastly, the administrator is offered the option create an unencrypted copy of the private key.

The server certificate and its private key are written to a special subdirectory in the CA's working directory. The administrator can then provide the certificate and key for the server to the system administrator who has requested it.

When option 2 is selected from caad's main menu caad will use the CA's private key to sign a certificate derived from an externally generated certificate request. This feature was put into the caad script so that services which generate their own certificate request and, will not accept an imported certificate and key, could be supported. Many Netscape web services and IIS fall into this category.

Here is an example of what happens when caad deals with an externally generated certificate request.

```
%% caad
```

1. Do you want to create a server cert?
2. Do you have a server cert request that you want to sign?
3. Do you want batch create multiple server certs?
4. Do you want to revoke a cert?

```
Please select the number of the option you would like: 2
```

```
Please provide the full path name of the request file: /tmp/certreq.txt
```

```
Please provide the name of the server: host/aserver.sdsc.edu
```

```
Using configuration from /certman/CA_SYSTEM/sdsc.server.cnf
```

```
Check that the request matches the signature
```

```
Signature ok
```

```
The Subjects Distinguished Name is as follows
```

```
commonName          :PRINTABLE:'host/aserver.sdsc.edu'  
organizationalUnitName:PRINTABLE:'SDSC'  
organizationName    :PRINTABLE:'NPACI'  
localityName        :PRINTABLE:'La Jolla'  
stateOrProvinceName :PRINTABLE:'CA'  
countryName         :PRINTABLE:'US'  
Certificate is to be certified until Jul  2 01:47:16 2007 GMT (1461 days)  
Sign the certificate? [y/n]:y
```

```
1 out of 1 certificate requests certified, commit? [y/n]y  
Write out database with 1 new entries  
Data Base Updated
```

%%

After caad has created a server or service certificate the certificate and its private key are put into a server.certs subdirectory within the CA's working directory. The certificate and key can then be copied from the server.certs directory and provided to the system administrator who requested the certificate. In the case of certificates created from externally generated certificate requests only the certificate is put in the server.certs directory because the key remains with program that generated the request. The certificate is also recorded in the index file, ca.db.index, in the CA's working directory.

The third option allows an administrator to create multiple server or service certificates without having to interact with the caad script to create each certificate. This is a useful for feature for situations in which many certificates must be created at one time e.g., when certificates are needed for every node in a cluster system. Here is an example of the contents of an input file which caad would use for creating multiple certificates:

```
host/tg-c001.sdsc.teragrid.org  
host/tg-c002.sdsc.teragrid.org  
host/tg-c003.sdsc.teragrid.org  
host/tg-c004.sdsc.teragrid.org  
host/tg-c005.sdsc.teragrid.org  
host/tg-c006.sdsc.teragrid.org  
host/tg-c007.sdsc.teragrid.org  
host/tg-c008.sdsc.teragrid.org  
host/tg-c009.sdsc.teragrid.org  
host/tg-c010.sdsc.teragrid.org  
.  
.  
.
```

Here is an example of the use of caad to batch create certificates using input from the example file above.

%% caad

1. Do you want to create a server cert?
2. Do you have a server cert request that you want to sign?
3. Do you want batch create multiple server certs?
4. Do you want to revoke a cert?

Please select the number of the option you would like: 3

Enter the name of file containing the list of hosts: /tmp/hosts

1. SDSC
2. UMICH
3. TACC
4. CALTECH

Please pick a number for the NPACI organization: 1

Please enter your private key encryption password:
Verifying private key password, please reenter password:

Would you like a decrypted copy of the private key? [y/n] y

```
Using configuration from sconfig
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'host-tg-c001.sdsc.teragrid.org.privkey.pem'
-----
Using configuration from sconfig
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'US'
organizationName      :T61STRING:'NPACI'
organizationalUnitName:PRINTABLE:'SDSC'
commonName            :PRINTABLE:'host/tg-c001.sdsc.teragrid.org'
Certificate is to be certified until Jul 31 19:24:51 2007 GMT (1461 days)
Sign the certificate? [y/n]:
```

```
1 out of 1 certificate requests certified, commit? [y/n]
Write out database with 1 new entries
Data Base Updated
read RSA key
writing RSA key
```

Information on the creation of each certificate will be written to the administrator's terminal, just as you can see above for the first certificate in the batch.

The fourth option that caad provides is certificate revocation. To identify the certificate to be revoked caad has to be provided with the serial number of the certificate. A certificate's serial number is a hexadecimal number that can be found within the certificate or in an index file of all certificates issued by the CA system. The index file will be described in the section covering the CA daemon and its operating environment. Caad will continue to prompt for the serial number of a certificate to be revoked until none is provided and a *< return >* is input, at which point caad will terminate.

Here is an example of caad being run to revoke a certificate:

```
%% caad

1. Do you want to create a server cert?
2. Do you have a server cert request that you want to sign?
3. Do you want batch create multiple server certs?
4. Do you want to revoke a cert?

Please select the number of the option you would like: 4

Enter the cert serial number: 0133
Using configuration from /certman/CA_SYSTEM/sdsc.cnf
Revoking Certificate 0133.
Data Base Updated
Using configuration from /certman/CA_SYSTEM/sdsc.cnf

Enter the cert serial number:

%%
```

When caad has revoked a certificate the CA system's CRL (Certificate Revocation List) is updated, the newly revoked certificate's serial number is added to the CRL. The entry for the certificate in the index file is also changed to reflect the new status of the certificate. Both the CRL and the index file are copied by caad to an NFS mounted directory which is also mounted by other

computer systems SDSC. The CRL and index file are both given world read permission so that they can be checked by processes running on SDSC systems which are dependent on certificates for authentication purposes.

4 The Certificate Authority Daemon:

The certificate authority daemon running on the CA machine listens to a chosen port number. The daemon is, in production usage, started by a cron job which is run every ten minutes, this job checks to see that the CA daemon is running and, if it is not, restarts it. The CA daemon can also be run from a terminal, in which case, information on its activities is sent to stdout. This is useful when one is making changes to the CA system because the daemon tells of its interactions with the cacl client program. The CA daemon is always run from a special account, called certman, which is used only for production CA activities and which exists only on the CA system. The certman account is also used by the CA administrator when running the caad program described in the previous section of this paper.

When the CA daemon begins execution the first thing it does is set up a socket bound to a specific port number. This will provide the communications channel between the cacl and the CA daemon. The CA daemon next goes into an infinite loop. At the top of the loop it reads the next certificate serial number to be used from a disk file and puts it into a variable. The daemon then calls a subroutine which waits to hear from a cacl program and receive a certificate request.

The subroutine which waits for and receives incoming certificate requests through the socket connection reads the CA system's private key encryption password off of disk and stores it in a variable. This password will later be used when decrypting certificate requests which have been encrypted with the CA system's public key, from its certificate. The subroutine next listens on the open socket for a certificate request. When a request is received the IP address from which the request originated is stored. The request is read off of the socket and put into a request working file. An additional operation, not tied directly to certificate request processing is also carried out at this point. A subroutine is called which goes through the CA's index file and looks for entries tied to certificates which have expired, when such entries are found it marks them as expired. This will ensure that when a new certificate is requested which will have the same DN as a certificate which has expired, there will be no question that the previous entry is expired and a new certificate with that DN can be issued.

The next subroutine called performs the initial certificate request processing operations. The first thing done in the subroutine is to decrypt the certificate request. If the decryption is unsuccessful the message "request decrypt failed" is sent back through the socket connection to cacl and will be displayed for the user. When a certificate request file is successfully decrypted the contents of the request are then checked.

The request file will contain a PKCS#10 format certificate request. In the case of the cacl CA system run at SDSC the country field must be C=US, the organization field must be O=NPACI, the organizational unit field must be OU=SDSC, for all user certificate requests. In each case if the content of a field is missing or incorrect an error message is sent back to cacl describing the problem and it is displayed for the user. At this point the contents of the common name field and userid field are read from the request and put into program variables, they will be used in the next step in request processing.

At SDSC user account information is managed by PAS (Password Administration System). This puts all of the user account information usually found in a Unix system's /etc/passwd and /etc/shadow files into a single master password file. The PAS master password file is available to the CA daemon and used by it to authenticate users making certificate requests. The userid supplied in the certificate request is used to lookup the user account entry in the master password file and then the CN provided in the certificate request is compared with the name found in the GECOS field of the password file. Next the salt from the encrypted password in the master password file entry is run, along with the plain text password provided in the challenge password field in the certificate request, through crypt to produce an encrypted password. The encrypted password from the request is compared with the encrypted password in the user's master password file entry. If the encrypted passwords match then the user making the certificate request has been authenticated to the same level of certainty as would be accomplished by Unix system login authentication. A certificate can now be signed by the CA daemon for the request if no other valid i.e., not expired or revoked, certificate with the same distinguished name exists, which was previously signed by the CA. If a signed certificate is created, it is sent back to the requesting cacl program through the socket connection. If the request failed because the userid could not be found in the master password file, the CN did not match, or the password did not match, a message explaining the cause of the failure is sent back through the socket and displayed by cacl for the user.

In the past, SDSC had production systems which were not covered by PAS and consequently the master password file did not contain account information for those systems. These were Cray Unicos systems, which had a UDB (User Data Base) containing the needed information. A program, which ran as a root cron job, was put on those systems to extract data from the UDB and construct a password file containing all of the information needed by the CA daemon for user certificate request authentication. These password files were made available to the CA daemon. When a certificate request came to the CA daemon the IP address from which the request originated was checked in order to determine which password file would be used for authentication purposes. The CA daemon can be easily modified to support the use of multiple password files from different machines for authentication purposes, if no single account management system will provide the information needed for all of the machines on which cacl is installed.

When the content of the certificate request has been determined to be correct

and, the user who made the request by running cacl has been authenticated by checking the userid, CN, and login password then, the next step is to create a certificate and sign it with the CA's private key. This process involves the use of an OpenSSL command to create a certificate based on the certificate request. OpenSSL checks its index file, ca.db.index, to see if a valid i.e., not revoked or expired, certificate exists with the same DN as that in the certificate which is about to be signed. If a valid certificate with the same DN does exist then the certificate request will be rejected and the reason will be provided by cacl to the user. If no DN matching that in the request can be found the certificate will be created and signed.

At SDSC "Guest" accounts exist which are provided to individuals who are taking training classes. The CA daemon checks the CN from the certificate request to find if it is for a guest account, if it is, a certificate which will expire in seven days is issued. If the request is not for a guest account the issued certificate will expire in four years.

After a certificate has been signed it is sent through the socket connection to the cacl program that requested it. The index file is updated with an entry for the new certificate, the file containing the next serial number to be used in a certificate is incremented by one, and the new certificate is put into a subdirectory within the CA's working directory. Next the certificate and index file are copied to an NFS mounted directory to be made available to other processes which will need the information they provide. Lastly, the CA requests a copy of the user encrypted private key from the cacl program that made the certificate request and stores in in a subdirectory within its working directory. The CA daemon then waits for another certificate request to be sent to it.

A The Cacl Man Page

cacl(1)

cacl(1)

NAME

cacl - Certificate Authority CLient, request a digital certificate and create a Globus user environment

SYNOPSIS

/usr/local/apps/pki_apps/cacl

DESCRIPTION

The cacl program is a user utility for requesting a digital certificate. The program creates a certificate request which is sent to a CA (Certificate Authority) for signing. The user is authenticated by the CA which checks the account name and password in the request file against the account name and password in a password file. When a signed certificate is returned by the CA it is put into a .globus directory within the user's home directory. The matching private key for the certificate, a PKCS#12 version of the certificate, and a symbolic link to a Grid Security Infrastructure directory is also put into the .globus directory.

When you run the cacl program you will be prompted for your login password. The password together with account information extracted from the /etc/passwd file will be put into the certificate request. Next you will be prompted twice for a private key encryption password. This will be used to encrypt your private key which will be stored in your .globus directory. After the CA has processed your request and returned your signed certificate cacl will convert that certificate from PEM format into PKCS#12 format so that the certificate can be imported into a Netscape browser. The private key is contained within the PKCS#12 certificate file, the key is encrypted with the same password used for the key file.

When you have successfully run cacl you will find a .globus

directory in your home directory. That directory will contain the following files:

usercert.p12 - Your digital certificate and private key in a PKCS#12 format certificate

usercert.pem - Your digital certificate, signed by the CA daemon

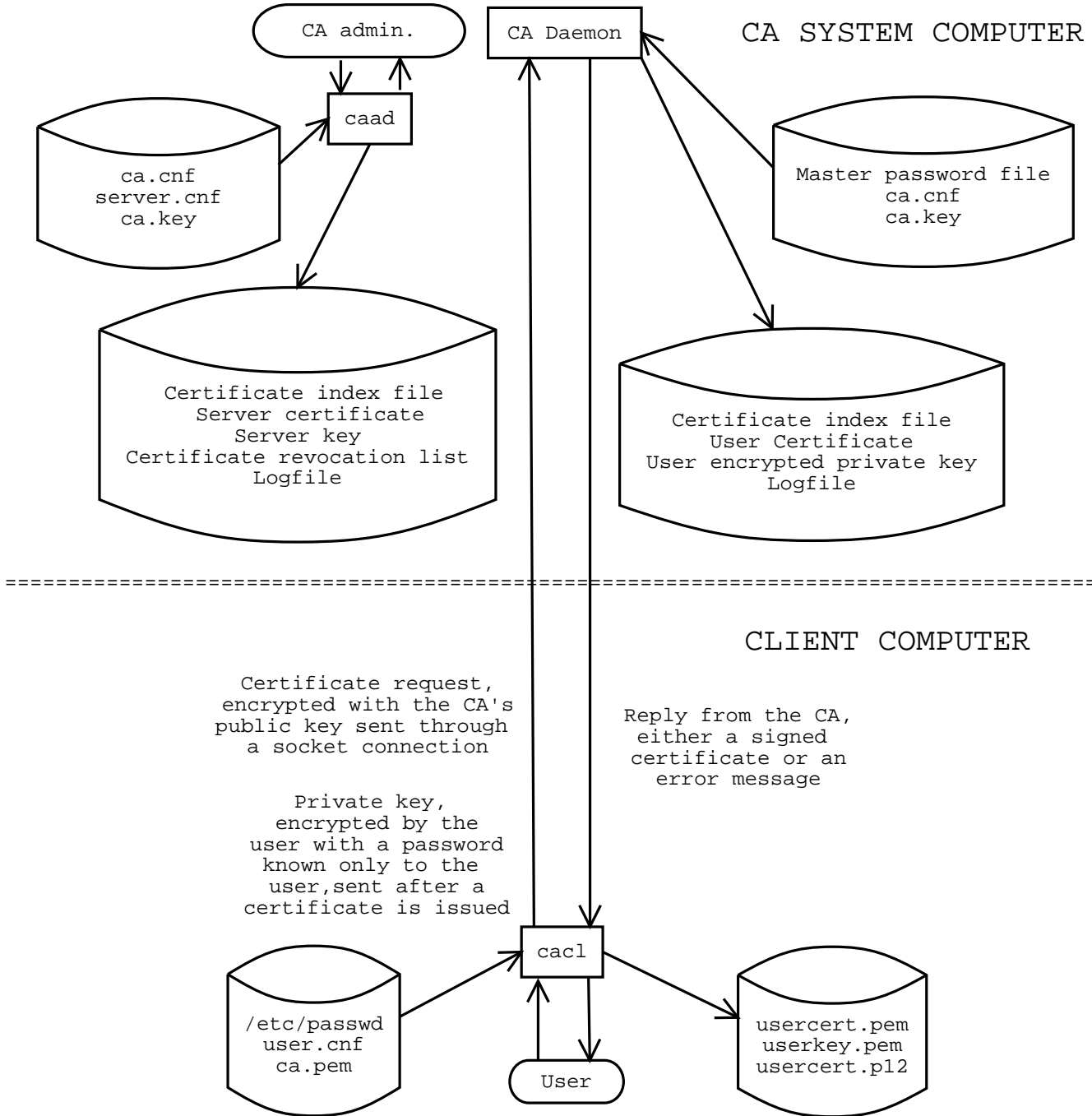
userkey.pem - Your private key matching the public key contained in your usercert.pem

ERRORS

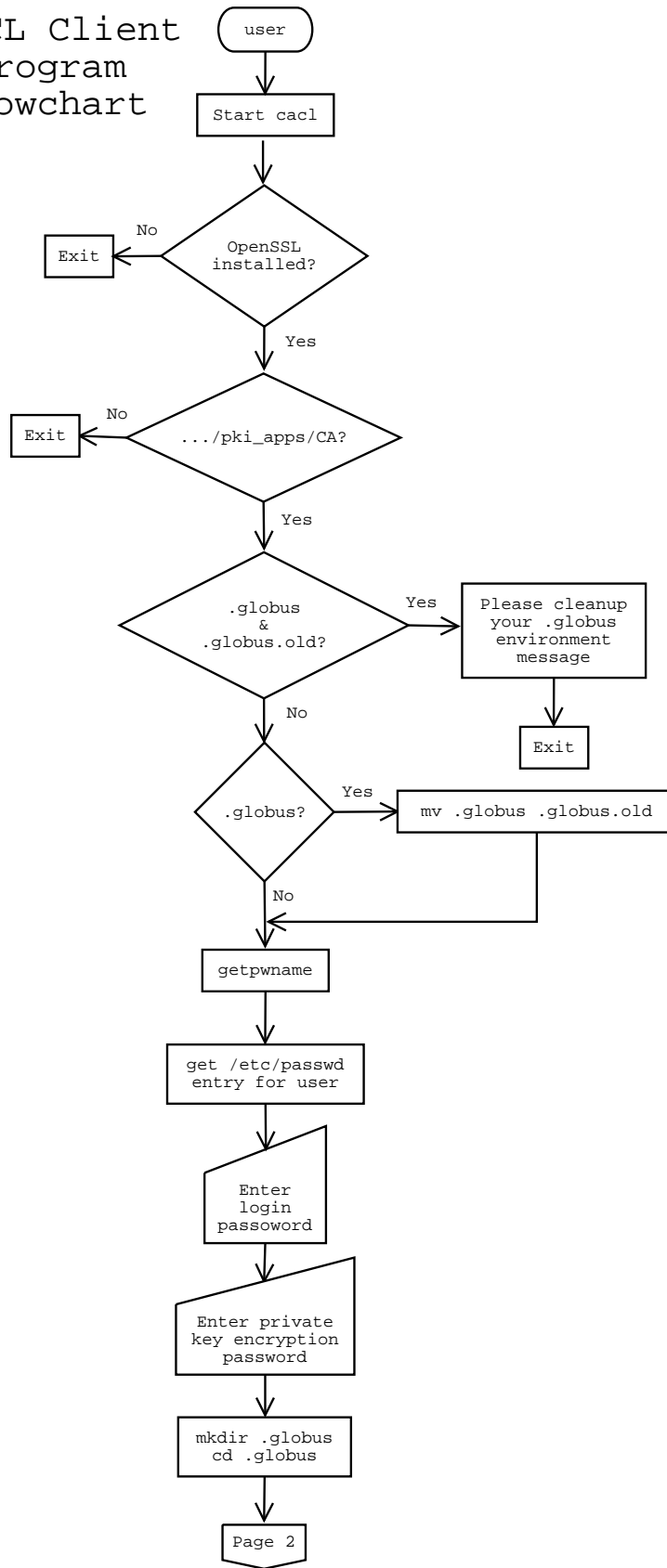
Before issuing a new certificate the CA will check see if a valid certificate already exists for the user, if such a certificate exists a new certificate will not be issued.

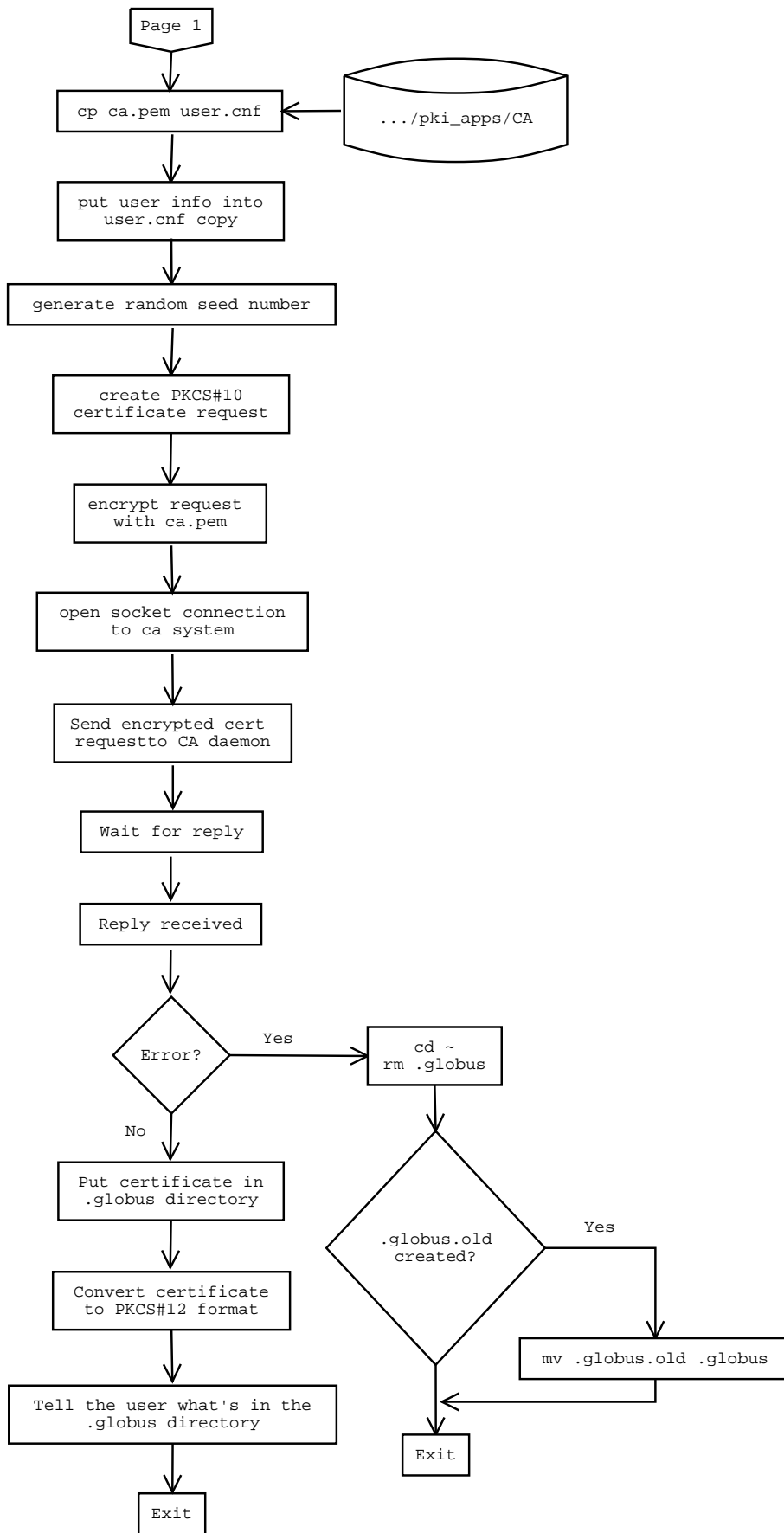
When the cacl program is run it checks to see if a .globus directory exists in the user's home directory. If a .globus directory exists it is moved to .globus.old. If both a .globus and a .globus.old exist cacl will quit. Cacl can be rerun after the .globus directory has been renamed or removed.

B CA System Overview Flowchart

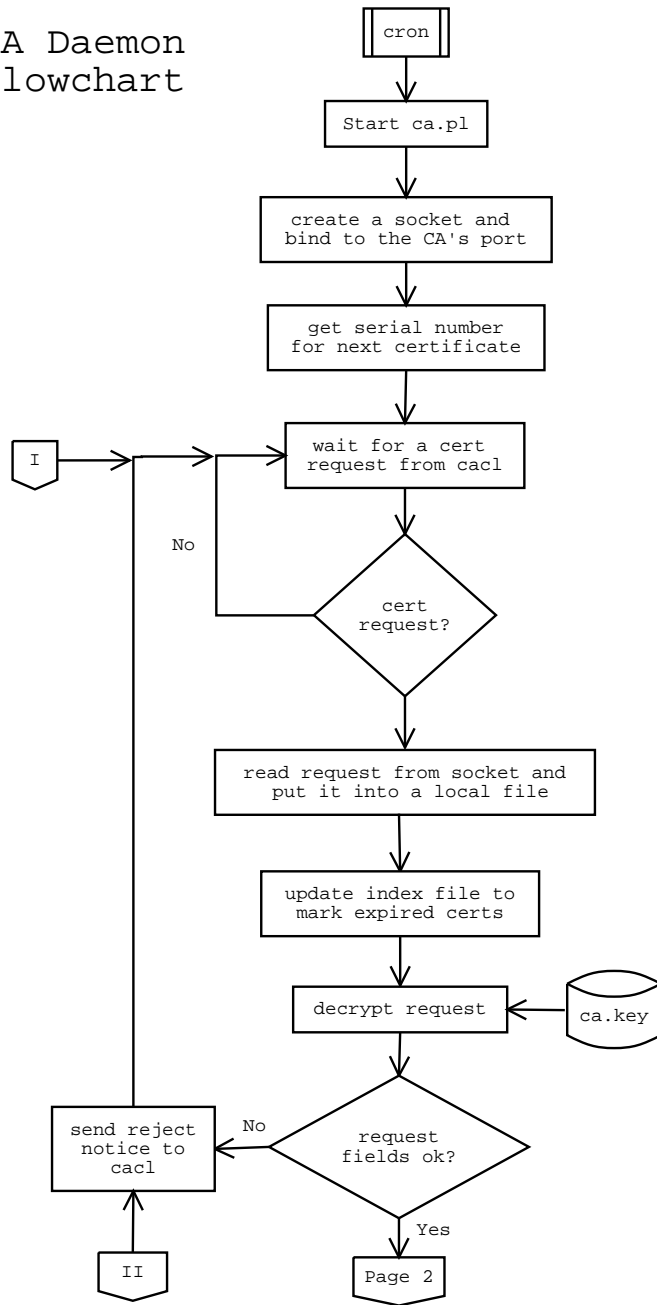


C CACL Client Program Flowchart





D CA Daemon Flowchart



CA DAEMON

