An Overview of X.509 Certificates

Chelsea Jean-Mary

Senior Technical Advisor:
Linda Harrison
## Contents

### Background
- What is Public-key Cryptography? 03
- What is a Digital Signature? 04
- Digital Certificates 06
- Types of Certificates 07
- Public-key Infrastructure 08
- Certificate Revocation 09
- Certificate Request 11
- Managing Keys and Certificates 14
- Uses for Certificates in Internet Applications 15
- Authentication using digital certificates 16

### X.509 Certificates
- What Are X.509 Certificates? 18
- Certificate Formats 20
- Protocols Supporting X.509 Certificates 22

### Certificate Usage on z/OS
- Creating and Storing Certificates 26
- z/OS Server Authentication 27
- Common Exploiters of x.509 certificates 28

### Considerations

### Certificate Errors
- Server Cannot Finds its Certificate 30
- Expired x.509 Certificate and Keys 32

### How-to Guide
- Certificate Services with RACDCERT 46
- Generate a certificate 47
- Create a RACF keyring 48
- Create a SITE certificate 50
- Create a new key ring for a shared SITE certificate 52
Background
I've been posting my public key for 15 years now, but no one has ever asked me for it or used it for anything as far as I can tell.

Maybe I should try posting my private key instead.
What is Public-key Cryptography?

Uses two keys - a *public* key potentially known to everyone and a *private* key that is known only to one party in an exchange of information

- Early 70s - First discovered by British government cryptographers
- Mid-70s - Three famous researchers, Ron Rivest, Adi Shamir, and Leonard Adleman, created the popular RSA asymmetric key algorithm using public-key cryptography.

**Benefit:**

- Used to encrypt data sent between two parties and authenticates sender identity

**Cons:**

- Uses complex calculations and are relatively slow. They are not used for encryption of bulk data.
What is a Digital Signature?

An electronic signature created by using a digital certificate’s private key. Provides a unique electronic binding of the identity of the signer to the origin of the object.

To verify the digital signature, we need the signing authority’s public key:

1. A secure hashing algorithm is used to create a digest of the message or certificate’s content
2. The digest is encrypted with the CA’s private key
3. The signature is decrypted with the CA’s public key
4. A new digest of the certificate’s contents is made and compared with the decrypted signature
Digital Certificates

An electronic document used to identify an individual, server, company or some other entity. It contains the public key and is signed with the private key to verify that it belongs to the sender.

- Two purposes:
  - Establishes the owner’s identity
  - Makes the owner’s public key available

- Contains specific pieces of information about the identity of the certificate owner and, if applicable, the Certificate Authority:
  - Owner’s distinguished name of the owner of the public key, aka subject’s name
  - Issuer’s distinguished name, aka issuer’s name
  - Owner’s public key itself
  - The time period during which the certificate is valid, aka the validity period
  - Certificate’s serial number as designated by the issuer
  - Issuer’s digital signature

WSC Blender Program / x.509 certificates / September 21, 2020 / © 2020 IBM Corporation
Only three types of certificates can be stored in RACF keyrings:

1. A *user certificate* is associated with a RACF user ID and is used to authenticate the user’s identity
   - An organization can issue its own certificate with itself as subject and issuer, signing with its own private key. This is called a *self-signed certificate*.
   - A certificate issued by a well-known Certificate Authority (CA) corporation (Verisign, Thawte, etc.) is known as a *CA-signed certificate*.
     - A certificate can be issued by a local CA.
2. A *shared site certificate* is a certificate shared among two or more servers (user IDs).
3. A *Certificate Authority* certificate is associated with a certificate authority and is used to verify signatures in other certificates.
Public-Key Infrastructure

IF SCHOOL KIDS WOULD USE PKI TO PROOF THEIR OWNERSHIP

THIS IS MY PEN
THIS IS MY PEN

LOOK HERE IS MY PEN CERTIFICATE

LET'S ASK AN AUTHORITY

MOM, DAD!

?
Public-Key Infrastructure

- In order to ensure that a public key you have was really created by that specific person, you need some sort of trusted third party.

- 1990s – Public-key infrastructure (PKI) surfaced commercially. This is a set of technologies and standards to manage the creation, storage, and distribution of certificates bound to Certificate Authorities (CAs). This was helped by x.509 certificates.
Certificate Revocation

- Certificate renewal interval is based on risk analysis of potential losses. X.509 renewal interval is usually one year.

- Revocation provides a means for a certificate to become invalid prior to its validity end date

- **Reasons for revocation:**
  - Private key associated with the certificate has been compromised
  - Certificates are being used for purpose other than what they are defined

- Revocation information is made available either via
  - Publishing of a Certificate Revocation List (CRL) in an LDAP directory
  - Real-time interrogation of the Certificate Authority via the Online Certificate Status Protocol (OCSP)
Certificate Revocation

- **Certificate Revocation List (CRL)**
  - A time-stamped list of revoked certificates that is signed by a certificate authority
  - Equivalent to 1970s-era credit card blacklist booklets that were based on earlier cheque blacklists

- **How it works:**
  - Fetch certificate
  - Fetch certificate revocation list (CRL)
  - Check certificate against CRL
  - Check signature using certificate
Certificate Revocation

- Online Certificate Status Protocol (OCSP) inquires of the issuing CA whether a given certificate is still valid.
- Returned status values are “good”, “revoked”, or “unknown”
- Instead of obtaining a CRL of everything, the process with OCSP sends a CRL/OCSP response for specific certificates
Certificate Request

- A certificate request that is sent to a certificate authority to be signed contains the owner’s distinguished name, the owners’ public key, and the owner’s own signature over these fields.

- The Certificate Authority verifies the signature with the public key in the digital certificate to ensure that:
  - The certificate request was not modified in transit between the requester and the CA
  - The requester is in possession of the private key that belongs to the public key in the certificate request
Managing Keys and Certificates

- **User requests certificate**
- **CA generates and distributes certificates**
- **Owner uses the certificate**
- **Certificate expires or administrator or user revokes certificate**
- **CA approves the request**
- **User renews certificate**
- **CA rejects the request**
Uses for Certificates in Internet Applications

SSL/TLS
- Used by servers for security for connections between servers and clients
- SSL/TLS uses digital certificates for key exchange, server authentication, and client authentication

Secure electronic mail
- Privacy Enhanced Mail (PEM) or Secure/Multipurpose Internet Mail Extensions (S/MIME)
- Uses digital certificates for digital signatures and for exchange of keys to encrypt and decrypt messages

Virtual Private Networks (VPNs)
- aka secure tunnels that can be set up between firewalls to enable protected connections between secure networks over insecure communication links. The protocols used in tunneling follow the IP Security (IPSec) standard.

Secure Electronic Transaction
- Standard for secure credit card payments in insecure networks
- Digital certificates are used for cardholders to allow for secure private connections between cardholders, merchants, and banks

This is only a subset of certificate usage and not meant to be a comprehensive list.
Authentication using digital certificates

**Client authentication**
- An option in SSL that requires a server to authenticate a client’s digital certificate prior to login
- Server requests and authenticates client’s digital certificate during the SSL handshake
- The identification of a client by a server, or the identification of the person assumed to be using the client

**Server authentication**
- The identification of a server by a client, or the identification of the organization assumed to be responsible for the server at a network address
X.509 Certificates
"X.509 certificates...recognizes that information within a digital certificate are all placed in the same location and in the same order, which makes it possible for all kinds of certificates to be shared across organizations."

All Types of Digital Certificates Are Also A X509 Certificate

Digi-Sign
What Are X.509 Certificates?

• 1998– the first X.509 digital certificate was issued as part of the International Telecommunications Union’s (ITU) Telecommunication Standardization Sector and the X.500 Directory Services Standard.

• X.509 uses the widely accepted international X.509 public key infrastructure standard to verify that a public key belongs to the user, computer or service identify contained within the certificate.

• When a certificate is generated, a private key is also produced, but this private key is not stored inside the x.509 certificate. Only the public key resides in the x.509 certificate.
  – The private key resides in the repository of the end-entity that is represented with the certificate.
  – The private key is used for signing certificates and decrypt data that has been encrypted with the public key. The private key may also be used to encrypt data.

• Every x.509 certificate contains a data section and a signature section.
What Are X.509 Certificates?

• Certificates include:
  – Which **X.509 version** applies to the certificate
  – **Serial number** – the identity creating the certificate must assign it a serial number that distinguishes it from other certificates
  – **Algorithm information** – the algorithm used by the issuer to sign the certificate
  – **Issuer distinguished name** – name of the entity issuing the certificate (Usually a CA)
  – **Validity period** of the certificate – start/end date and time
  – **Subject distinguished name** – the identity the certificate is issued to
  – **Subject public key information** – the public key associated

• The signature section includes:
  – The cipher algorithm – the algorithm used by the issuer to create a digital signature
  – The digital signature for the CA – a hash of all the information in the certificate encrypted with the CA private key
Certificate Formats

Certificate are packaged into “formats”.

![Diagram](image.png)
Certificate Formats

Binary

- **.DER** (Distinguished Encoding Rules)
  - A single binary certificate, platform-independent format, the default format for most browsers
  - **Use**: Used for Certificate Requests, which are always DER-encoded and then base64-encoded
- **PKCS#12** (Public-Key Cryptographic Standards)
  - One or more certificates packed together, password-encrypted
  - **Use**: When the CA wants to ship a package confidentially that contains the private key

Base64 (ASCII)

- **.PEM**
  - Default format for OpenSSL. Suitable for sending files as text between systems
  - Prefixed with a “--BEGIN...” line
  - **Use**: When making a Certificate Request in an email
- **PKCS#7**
  - One or more certificate packaged together but not signed or encrypted
  - **Use**: When the CA wants to deliver multiple certificates to a destination
Here are some UNIX commands that will let you output the contents of PEM and DER-encoded certificate formats in a human readable form:

### View
Use the command that has the extension of your certificate replacing `cert.xx` with the name of your certificate.

**View a PEM encoded certificate**
```bash
openssl x509 -in cert.pem -text -noout
```

**View a DER encoded certificate**
```bash
openssl x509 -in certificate.der -inform der -text -noout
```

### Transform
Take one type of encoded certificate to another (i.e. PEM to DER)

**Transform a PEM encoded certificate to DER encoded certificate**
```bash
openssl x509 -in cert.crt -outform der -out cert.der
```

**Transform a DER encoded certificate to PEM encoded certificate**
```bash
openssl x509 -in cert.crt -inform der -outform pem -out cert.pem
```
Protocols Supporting X.509 Certificates

- Transport Layer Security (SSL/TLS)
- IPSec
- Secure Multipurpose Internet Mail Extensions (S/MIME)
- Smartcard
- SSH
- HTTPS
- LDAPv3
Certificate Usage on z/OS
Creating and Storing Certificates

- Create certificates and certificate requests with:
  - RACDCERT in z/OS RACF
  - gskkyman in z/OS UNIX

- Certificates are stored and managed
  - in a RACF keyring
  - in a key database file in z/OS UNIX.
1. Verify server certificate has not expired
2. Verify server certificate is valid using CA’s public key
3. Do optional checks on the server certificate
4. Store server’s public key for later use
5. Generate symmetric key and encrypt under server’s public key

- The server certificate is signed by the CA and is stored on the server side.
- During SSL handshake, the server certificate (not the server private key) is sent to the client.
<table>
<thead>
<tr>
<th>Exploiter</th>
<th>Connecting a Server Certificate to a Keyring</th>
<th>Where to Specify the Keyring Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>TN3270 Server (native SSL/TLS)</td>
<td>Connect server certificate as the DEFAULT on the keyring</td>
<td>Telnet Profile “KEYRING SAF ABCRING” (RACF)</td>
</tr>
<tr>
<td>TN3270 Server (AT-TLS)</td>
<td>Connect as the DEFAULT if the AT-TLS policy does not exploit the Certificate Label Name. If it does, certificate does not need to be the DEFAULT.</td>
<td>AT-TLS Policy</td>
</tr>
<tr>
<td>FTP Server (native SSL/TLS)</td>
<td>Connect server certificate as the DEFAULT on the keyring</td>
<td>FTP.DATA file “KEYRING ABCRING”</td>
</tr>
<tr>
<td>FTP Server (AT-TLS)</td>
<td>Connect as the DEFAULT if the AT-TLS policy does not exploit the Certificate Label Name. If it does, certificate does not need to be the DEFAULT.</td>
<td>AT-TLS Policy</td>
</tr>
<tr>
<td>IP Security (IPSEC) Dynamic VPN in z/OS Comm Server</td>
<td>Connect as the DEFAULT if the AT-TLS policy does not exploit the Certificate Label Name. If it does, certificate does not need to be the DEFAULT.</td>
<td>Iked.conf file “KEYRING ABCRING”</td>
</tr>
<tr>
<td>HTTP Server (native SSL/TLS)</td>
<td>Connect server certificate as the DEFAULT on the keyring</td>
<td>httpd.conf file “KEYRING ABCRING”</td>
</tr>
<tr>
<td>WebSphereMQ (native SSL/TLS)</td>
<td>NOTE: Label of the certificate must start with “ibmWebSphereMQ”</td>
<td>Issue MQ command: “ALTER QMGR SSLKEYR(ABCRING)”</td>
</tr>
</tbody>
</table>
Considerations
Considerations

• A digital certificate alone can never be proof of anyone’s identity. It simply verifies the identity of the digital certificate owner by providing the public key that is needed to check the digital certificate owner’s digital signature. Therefore the digital certificate owner must protect the private key that belongs to the public key in the digital certificate.

• An application that authenticates the owner of a certificate cannot accept just the certificate. A message signed by the certificate owner should accompany the certificate. This allows the application to verify that the message is a “fresh” signature from the certificate owner and not a replayed message from an imposter.

• The larger the public/private key pair size, the greater the security but more computationally intensive.

• When transferring certificates, use a format acceptable to the receiving side. Be sensitive to binary and text modes to ensure proper transfer!
Certificate Errors
Certificate Errors

• For z/OS Communications Server, it is rarely necessary to take an SSL trace for diagnosing problems with SSL/TLS or AT-TLS. A simple look in SYSLOGD or the MVS console can reveal what has gone wrong with the secured connection you are testing.

• **Common Problems:**
  - Wrong Owner of Key Ring that Certificates reside on
  - Wrong Owner of Personal Certificate
  - Expired Certificate
  - Signing authority certificate not available to validate a certificate received during negotiation of secured session
  - Exporting certificates without understanding certificate formats
Error Example #1: Server Cannot Find its Certificate

The following flow is seen when a client connects to an AT-TLS secured port.
Error Example #1: Server Cannot Find its Certificate

The following flow is seen when a client connects to an AT-TLS secured port.

In this case, the FTP server fails to provide a certificate.
Error Example #1: Server Cannot Find its Certificate

The following shows the error messages. The server reset the connection, not the client.

```
234 Security environment established - ready for negotiation
FC2838 authServerAttls: Start Handshake
FC2847 authServerAttls: ioctI() failed on SOCTTLSCTL - EDC8121I Connection reset. (errno2=0x77A9733D) <<<<<<<<<<<<<< Different errno2 from previous example
EZA28971 Authentication negotiation failed
EZA1534I *** Control connection with 192.168.20.93 dies.
```

```
AT MVS3 (SERVER) : EZend871 TTLS Error RC: 6 Initial Handshake
AT MVS1 (CLIENT) : Ezend871 TTLS Error RC: 438 Initial Handshake
```
Error Example #1: Server Cannot Find its Certificate

The following shows the error messages. The server reset the connection, not the client.
Error Example #1: Server Cannot Find its Certificate

The following shows the error messages. The server reset the connection, not the client.

```
234 Security environment established - ready for negotiation
FC2838 authServerAtls: Start Handshake
FC2847 authServerAtls: ioctl() failed on SiOCTTLSCTL - EDC8121I Connection reset. (errno2=0x77A9733D) Different errno2 from previous example
EZA2897I Authentication negotiation failed
EZA1534I *** Control connection with 192.168.20.93 died
```

“Key label is not found.”

“Internal error reported by remote partner.”
Error Example #1: Server Cannot Find its Certificate

The following shows the error messages. The server reset the connection, not the client.

MVS (server) indicates a problem with finding the correct key label in the key database or ring.

“Key label is not found.”

“Internal error reported by remote partner.”
Error Example #1: Server Cannot Find its Certificate

The following shows a display of the key ring at MVS3 (server).

```
racdcert id(FTPD) listring(Server_RING)
Digital ring information for user FTPD:
    Ring:
        >Server_RING<

Certificate Label Name       Cert Owner   USAGE   DEFAULT
-----------------------------------------------
MVS1 LABS Certificate Authority     CERTAUTH    CERTAUTH   NO
```
Error Example #1: Server Cannot Find its Certificate

The following shows a display of the key ring at MVS3 (server).

```
racdcert id(FTPD) listring(Server_RING)
Digital ring information for user FTPD:

Ring:  >Server_RING<

<table>
<thead>
<tr>
<th>Certificate Label Name</th>
<th>Cert Owner</th>
<th>USAGE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVS1 LABS Certificate Authority</td>
<td>CERTAUTH</td>
<td>CERTAUTH</td>
<td>NO</td>
</tr>
</tbody>
</table>
```

“Where is the FTP Server Certificate? It is not on the server’s key ring!”
Error Example #1: Server Cannot Find its Certificate

Solution:

1. Connect FTP Server Certificate to Key Ring.

2. Change Policy Instance number and UPDATE PAGENTT to reinstall changes to the Key ring for FTP Server.

3. Recycle FTP Server to reinstall changes to Key Ring.

Digital ring information for user FTPD:

<table>
<thead>
<tr>
<th>Certificate Label Name</th>
<th>Cert Owner</th>
<th>USAGE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVS1 LABS Certificate Authority</td>
<td>CERTAUTH</td>
<td>CERTAUTH</td>
<td>NO</td>
</tr>
<tr>
<td>FTP Server on MVS1-MVS7</td>
<td>ID(FTPD)</td>
<td>PERSONAL</td>
<td>YES</td>
</tr>
</tbody>
</table>
Error Example #2: Expired x.509 Certificate and Keys

This visual shows the SSL Return code that you would see if your certificate cannot be found due to an invalid status of one type or another.

**MVS Console Log at FTP Server LPAR**

```
EZD1287I TTLS Error RC: 401 Initial Handshake 036
LOCAL: 192.168.20.113..21
REMOTE: 192.168.20.111..1038
JOBNAME: FTPPT1 RULE: FTPPT@192.168.20.113 2
USERID: TCPIP GRPID: 00000002 ENVID: 00000009 CONNID: 000000AD
```

**SYSLOGD Output with AT-TLS Policy Trace Level of 255 at Client LPAR**

```
EZD1284I TTLS Flow GRPID: 00000001 ENVID: 00000006 CONNID: 000000CA
RC: 401 Call GSK_SECURE_SOCKET_INIT - 7EB3C318

EZD1283I TTLS Event GRPID: 00000001 ENVID: 00000006 CONNID: 000000CA
RC: 401 Initial Handshake 00000000 7EB9C798

EZD1286I TTLS Error GRPID: 00000001 ENVID: 00000006 CONNID: 000000CA
LOCAL: 192.168.20.113..1038 REMOTE: 192.168.20.113..21 JOBNAME:
USER201 USERID: USER301 RULE: FTPPTClient@192.168.20.11n~5
RC: 401 Initial Handshake 00000000 7EB9C798
```
Error Example #2: Expired x.509 Certificate and Keys

This visual shows the SSL Return code that you would see if your certificate cannot be found due to an invalid status of one type or another.

"Cryptographic Services SYSTEM SECURE SOCKETS LAYER Programming (SC24-5901-10)

At MVS1 and MVS3 ... RC 401:
401 Certificate is expired or is not valid yet.

Explanation: The current time is either before the Certificate start time or after the Certificate end time.

User response: Obtain a new Certificate if the Certificate is expired or wait until the Certificate becomes valid is not valid yet.

Digital certificate information for user TCPIP:
Label: FTPServer31 EXP
Certificate ID: 2QXjw9fJ18bj1+KFmaWFmfLyQMXn10BA
Status: TRUST
Start Date: 2017/06/07 00:00:00
End Date: 2020/06/07 23:59:59
Error Example #2: Solution

To renew the expiration dates of a Certificate, follow these steps:

1. Generate a Certificate Request for the Certificate with the invalid dates ("RACDCERT GENREQ" command)

2. Generate a new Certificate, keeping the original old date, but extending the new data by one year. ("RACDCERT GENCERT" command)

3. Mark the Certificate as TRUSTED – since the old date will cause it to default to UNTRUSTED. ("RACDCERT ALTER" command)
How-to Guides
Certificate Services with RACDCERT

One method of creating and managing certificates and asymmetric keys is to use the RACF command RACDCERT.

RACDCERT can be used to create public and private keys and store them in the RACF database along with X.509 certificates containing the public keys.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>Add certificate into the RACF database</td>
</tr>
<tr>
<td>ADDRING</td>
<td>Create a RACF keyring</td>
</tr>
<tr>
<td>ALTER</td>
<td>Change the trust status of a certificate</td>
</tr>
<tr>
<td>CHECKCERT</td>
<td>Check whether a certificate has already been added to the RACF database</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Connect a certificate to a keyring</td>
</tr>
<tr>
<td>DELETE</td>
<td>Delete a certificate from the RACF database</td>
</tr>
<tr>
<td>DELRING</td>
<td>Delete a keyring</td>
</tr>
<tr>
<td>EXPORT</td>
<td>Export a certificate as a PKCS#7 package or as a binary DER certificate</td>
</tr>
<tr>
<td>GENCERT</td>
<td>Generate a key pair and certificate</td>
</tr>
<tr>
<td>GENREQ</td>
<td>Generate a PKCS# 10 certificate request</td>
</tr>
<tr>
<td>LIST</td>
<td>List the contents of a certificate</td>
</tr>
<tr>
<td>LISTRING</td>
<td>List the contents of a keyring</td>
</tr>
<tr>
<td>REMOVE</td>
<td>Remove a certificate from a keyring</td>
</tr>
</tbody>
</table>
Generate a certificate

**a. Create a server certificate and a private/public key pair:**
- ID(USERID) – the started task user ID of your server

**b. Generate a request to have the certificate signed by an external CA**
- Send the request to the CA
- Receive from the CA

**c. Add the signed certificate into RACF**

```
RACDCERT ID(certificate-owner) GENCERT +
   SUBJECTSDN( +
      CN('common-name') +
      OU('organization-unit-name') +
      O('organization-name') +
      L('locality') +
      SP('state-or-province') +
      C('country') ) +
      SIZE(key-size) +
      NOTBEFORE(DATE/yyyy-mm-dd)) +
      NOTAFTER(DATE/yyyy-mm-dd)) +
      WITHLABEL('label-name') +
      ALTNAME( +
         DOMAIN('internet-domain-name'))
RACDCERT ID(certificate-owner) GENREQ (LABEL('label-name')) +
      DSN(output-data-set-name)
---
RACDCERT ID(certificate-owner) +
   ADD(data-set-name) +
   TRUST +
   WITHLABEL('label-name')
```
Generate a certificate (with JCL)

```plaintext
//CERT01 JOB MSGCLASS=X,NOTIFY=&SYSUID
//CERT01 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//********************************************************************
//* Create Individual Personal Certificate for server *
//********************************************************************
//SYSTSPRT DD SYSOUT=* 
//SYSTSPRT DD *
RACDCERT ID(certificate-owner) GENCERT
   SUBJECTSDN (CN('Common Name')
   OU('organization-unit-name1')
   C('country'))
   NOTBEFORE(DATE(yyyy-mm-dd))
   NOTAFTER(DATE(yyyy-mm-dd))
   WITHLABEL('label-name')
setropts raclist(DIGTCERT) refresh
racdcert ID(certificate-owner) list(label('label-name'))
/*
```

a. The ID parameter identifies this certificate is being generated as a server certificate
b. SUBJECTSDN identifies several components of the x.509 Distinguished Name (DN)
c. Timeframe for certificate’s validity
d. A label for organizing the certificate within a RACF database
e. Refreshes the DIGTCERT class so that the changes are known immediately
f. Displays the certificate with its attributes. Verifies that certificate has been properly created.
Create a RACF key ring

a. Create a keyring:
   - ADDRING(KeyRing) – the name of the key ring

b. Connect the Certificate Authority certificate to the keyring
   - ADD(data-set-name) – the dataset contains the CA certificate

c. Connect the server certificate to the keyring
   - ADD(data-set-name) – the dataset contains the server certificate

d. Show the contents of the keyring
Create a RACF key ring (with JCL)

a. Adds a key ring to the RACF database
b. Connects a certificate to the created keyring
c. Indicates this certificate may be used as a single entity (server) only
d. Connects the CA certificate to the key ring
e. Indicates that this is a CA certificate
f. Protects the key ring with RACF control commands
g. Protects the private key and permits the user ID of the server to access it
h. This refreshed classes in MVS storage.
i. This command lists the contents of the keyring with the label identified

```sql
//KEYRNG01 JOB MSGCLASS=X,NOTIFY=&SYSUID
//KEYRNG01 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*********************************************************************
//* Add a separate keyring for ring-owner for server *
//*********************************************************************
//SYSTSPRT DD SYSOUT=*  
//SYSTSIN DD *
racdcert ID(ring-owner) addring(ring-name) a  
racdcert ID(ring-owner) CONNECT(ID(certificate-owner) - b  
  LABEL('label-name')  
  RING(ring-name)  
  DEFAULT) c  
racdcert ID(ring-owner) CONNECT(CERTAUTH - d  
  LABEL('CA-label-name')  
  RING(ring-name)  
  USAGE(CERTAUTH)) c  
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(USER) ACCESS(READ)  
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(USER) ACCESS(READ)  
setropts raclist(DIGTRING) refresh  
setropts raclist(DIGTCERT) refresh  
setropts raclist(FACILITY) refresh  
racdcert listring(ring-name) id(ring-owner) i  
```
Create a SITE certificate

a. Create a SITE certificate

b. Lists names of all site certificates in the RACF database

c. Connect the site certificate to the new keyring using the RACDCERT CONNECT command

**RACDCERT SITE GENCERT SUBJECTSDN**
- cn(‘common-name’)
- o(‘organization-name’)
- ou(‘organizational—unit-name’)
- C(‘country’))
- withlabel(‘label-name’)
- signwith(certauth label(‘label-name’))

**RACDCERT SITE LIST**

**RACDCERT ID**
- ring-owner)
- CONNECT(SITE -
- LABEL(‘label-name’)
- RING(ring-owner) –
- DEFAULT
- USAGE(PERSONAL))
Create a SITE certificate \textit{(with JCL)}

\begin{verbatim}
//CERTSITE JOB MSGCLASS=X,NOTIFY=&SYSUID
//CERTSITE EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*****************************************************************
//* CREATE SITE AUTHORITY CERTIFICATE FOR ALL SERVERS (SHARED) *
//*****************************************************************
//SYSTSPRT DD SYSOUT=* //SYSTSIN DD *
RACDCERT SITE GENCERT SUBJECTSDN(CN('common-name') - a
    O('organization-name') -
    OU('organizational-unit-name') -
    C('country') ) -
     WITHLABEL('label-name') b
RACDCERT SITE LIST c
/*
\end{verbatim}

\begin{itemize}
  \item[a.] The SITE ID is used to indicate that this certificate is to be used as a site certificate.
  \item[b.] The label name implies that the certificate is a shared site certificate.
  \item[c.] The command lists the names of all known site certificates in the RACF database.
\end{itemize}
Create a new key ring for a shared SITE Certificate (with JCL)

a. Create a new RACF shared keyring using the RACDCERT ADDRING command

b. Connects the internal CA to the new keyring using the RACDCERT CONNECT command

c. Connect the site certificate to the new keyring using the RACDCERT CONNECT command

```
//KEYRNGS JOB MSGCLASS=X,NOTIFY=&SYSUID
//KEYRNGS EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//*********************************************************************
// Add a new keyring to the various clients’ RACF ID, then ...
// Add the SITE certificate to the servers’ keyring.
//*********************************************************************
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
racdcert ID(ring-owner) ADDRING(ring-name) a
racdcert ID(certificate-owner) CONNECT(CERTAUTH) - b
  LABEL('label-name') -
  RING(ring-name) -
  USAGE(CERTAUTH)
racdcert ID(ring-owner) CONNECT(SITE c
  LABEL('label-name') -
  RING(ring-owner) -
  DEFAULT -
  USAGE(PERSONAL))
setropts raclist(DIGTRING) refresh
setropts raclist(DIGTCERT) refresh
racdcert listring(*) id(ring-owner)
```
Appendix
Certificate Authorities

- A company that is considered trustworthy and produces digital certificates for other individuals and companies (i.e. subjects) bearing that subject’s public key

- A Certificate Authority (CA) can be either a:
  - well-known corporation (Verisign, Thawte, etc.) or
  - local CA that is established within a corporation to sign server or client certificates

- They are only issued for a limited time.