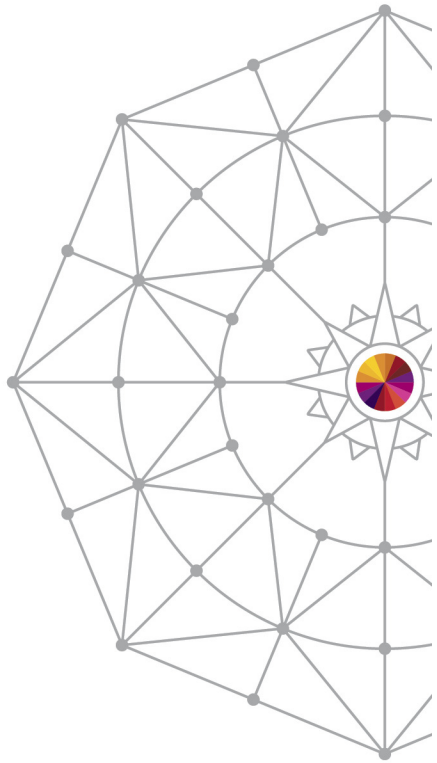
Digital Certificate Goody Bags on z/OS

Wai Choi, CISSP®
IBM Corporation

August 8th, 2014
Session: 15665

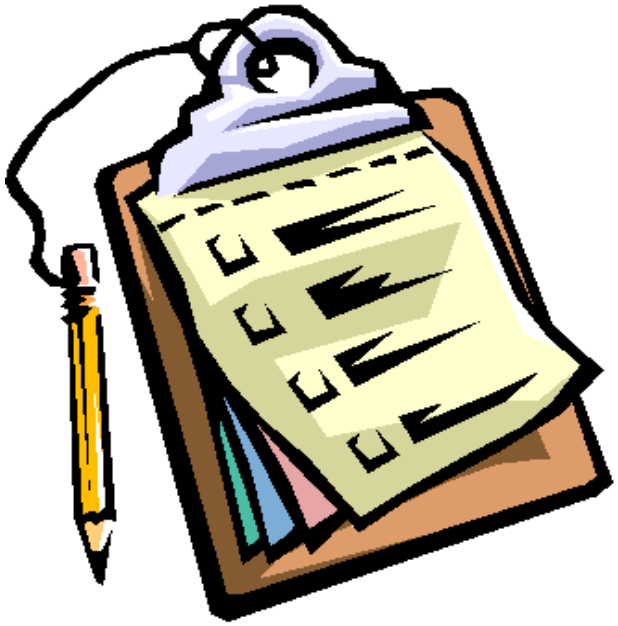


#SHAREorg

SHARE in Pittsburgh 2014

Agenda

- **What is a Digital Certificate?**
- RACF **RACDCERT** Command Overview
  - General tips
  - Generating a certificate request and renewing a certificate
- **RACF Key Rings:**
  - Real and Virtual Key Rings
  - Key Ring Protection
  - Certificate Sharing
  - Server Authentication
  - Client Authentication
- **Certificate Mapping on z/OS:**
  - One-to-one certificate to user ID association
  - Certificate Name Filtering (CNF)
  - Host Id Mapping extensions
- **z/OS Certificate Authority - PKI Services**
- **Certificate Life Cycle Planning**



What is a Digital Certificate?

A Digital Certificate is a digital document issued by a trusted third party which binds an end entity to a public key.

- **Digital document:**
  - Contents are organized according to ASN1 rules for X.509 certificates
  - Encoded in binary or base64 format
- **Trusted third party** aka **Certificate Authority** (CA):
  - The consumer of the digital certificate trusts that the CA has validated that the end entity is who they say they are before issuing and signing the certificate.
- **Binds the end entity to a public key:**
  - **End entity** - Any person or device that needs an electronic identity. Encoded in the certificate as the Subjects Distinguished Name (SDN). Can prove possession of the corresponding private key.
  - **Public key** - The shared half of the public / private key pair for asymmetric cryptography
  - **Digitally signed** by the CA

How is Digital Certificate used?

- **Prove Identity to a peer:**
  - Owner of the certificate can prove possession of the certificate's private key
  - Identity can be validated by checking it is signed by a trusted Certificate Authority
- **Prove origin of a digital document is authentic**:
  - Programs can be signed by code signing certificates
  - E-mail signatures
  - Certificates are signed by CA certificates
- **Establish a secure connection:**
  - Certificates contain a public key which allows protocols such as SSL and AT-TLS to exchange session keys

RACDCERT Overview

- **RACDCERT** is the primary administrative tool for managing digital certificates, key rings, certificate filters using RACF.
- **TSO command** shipped as part of **RACF**
- Command line interface with ISPF panels
- Certificates, rings and filters are protected by RACF profiles
- Learn more:
  - RACF Command Language Reference
  - RACF Security Administrator's Guide

```
RACDCERT ID(FTPServer) GENCERT SUBJECTSDN(CN('Server
Certificate')OU('Production')O('IBM')L('Poughkeepsie') SP('New
York')C('US')) SIZE(1024) WITHLABEL('Server Certificate')
ALTNAME(DOMAIN('mycompany.com'))


RACDCERT ID(FTPServer) ADD('user1.svrcert') WITHLABEL('Server
Certificate')


RACDCERT ID(userid) EXPORT (LABEL('label-name'))  DSN(output-
data-set-name) FORMAT(CERTDER | CERTB64 | PKCS7DER |
PKCS7B64 |   PKCS12DER | PKCS12B64 )   PASSWORD('pkcs12-
password')
```

```
                RACF - Digital Certificate Key Ring Services
OPTION ===> _

   For user: ________

Enter one of the following at the OPTION line:

    1   Create a new key ring
    2   Delete an existing key ring
    3   List existing key ring(s)
    4   Connect a digital certificate to a key ring
    5   Remove a digital certificate from a key ring
```

```
           RACF - Digital Certificate Services
OPTION ===>
                _
  Select one of the following:

       1. Generate a certificate and a public/private key pair.

       2. Create a certificate request.

       3. Write a certificate to a data set.

       4. Add, Alter, Delete, or List certificates or
          check whether a digital certificate has been added to
          the RACF database and associated with a user ID.

       5. Renew, Rekey, or Rollover a certificate.
```

Main RACDCERT Commands

- **Certificate Generation:**
  - RACDCERT **GENCERT** – Generate key pair and certificate
  - RACDCERT **GENREQ** – Generate a certificate request
- **Certificate Installation:**
  - RACDCERT **ADD** – Install a certificate and public/private key
- **Certificate Administration:**
  - RACDCERT **LIST** – Display information on a certificate installed in RACF
  - RACDCERT **LISTCHAIN** – Display information on a certificate chain installed in RACF
  - RACDCERT **ALTER** – Change LABEL or TRUST status of a certificate in RACF
  - RACDCERT **DELETE** – Delete certificate and key pair
  - RACDCERT **CHECKCERT** – Display certificate information from a dataset
  - RACDCERT **EXPORT** – Export a certificate
  - RACDCERT **REKEY** – Renew certificate with new key pair
  - RACDCERT **ROLLOVER** – Finalize the REKEY process

RACDCERT Commands

- **Certificate Ring Administration:**
  - RACDCERT **ADDRING** – Create a key ring
  - RACDCERT **CONNECT** – Place a certificate in a key ring
  - RACDCERT **REMOVE** – Remove a certificate from a key ring
  - RACDCERT **LISTRING** – Display key ring information
  - RACDCERT **DELRING** – Delete a key ring

- **Certificate Map Administration:**
  - RACDCERT **MAP** – Create a certificate filter
  - RACDCERT **ALTMAP** – Change the certificate filter
  - RACDCERT **DELMAP** – Delete a certificate filter
  - RACDCERT **LISTMAP** – Display certificate filter information

RACDCERT ID

- **RACDCERT** commands specified without the ID keyword will normally default to the user ID issuing the command:
  - **User1's certificate is displayed if user1 issues the following command**
    - `RACDCERT LIST(LABEL('cert1'))`
  - **User2's certificate is displayed if user1 issues the following command (assuming user1 has the authority to list other's certificate)**
    - `RACDCERT ID(user2) LIST(LABEL('cert2'))`
- Good practice to specify ID/CERTAUTH/SITE explicitly, and put it right after 'RACDCERT' to avoid confusion

RACDCERT CONNECT

- **RACDCERT CONNECT** connects a Certificate to a key ring.
- Uses two **different** user IDs:
  - **Certificate owner** – Defaults to ring owner, see example 2) below
  - **Ring owner** – Defaults to command issuer
- **Syntax:**

  RACDCERT ID(<ring-owner>) CONNECT(ID(<certificate-owner>) LABEL('<cert-label>')...RING(<ring-name>…)
- **Which is the best practice? Which is confusing?**

  1) `RACDCERT ID(Mary) CONNECT(ID(John) LABEL('JCert') RING(MRing)…)`
  - Ring owner: Mary, Cert owner: John

  2) `RACDCERT ID(Mary) CONNECT(LABEL('MCert') RING(Mring)…)`
  - Ring owner: Mary, Cert owner: Mary

  3) `RACDCERT CONNECT(ID(John) LABEL('JCert') RING(IRing)…)`
  - Ring owner: Issuer of command, Cert owner: John

  4) `RACDCERT CONNECT(LABEL('ICert') RING(IRing)…)`
  - Ring owner: Issuer of command, Cert owner: Issuer of command

RACDCERT GENREQ (1 of 2)

- **RACDCERT GENREQ** generates a certificate request for obtaining a certificate from a Certificate Authority.
- **GENREQ** requires an existing certificate. If a certificate does not exist, use **GENCERT** to create a self signed certificate first:
    - RACDCERT GENCERT (usually a self-signed one)
        - This is a stepping stone to get the request, will be replaced once the certificate is fulfilled by the CA
        - RACDCERT ID(ftpd) GENCERT SUBJECTSDN(CN('ftpcert') OU('RACF')…) WITHLABEL('ftpcert')
    - RACDCERT GENREQ <use the certificate label from GENCERT above >
        - RACDCERT ID(ftpd) GENREQ(LABEL('ftpcert')) DSN('user1.ftpreq')
    - **Send the request** to external CA for signing
    - When the certificate is returned from the external CA, install it in RACF with **RACDCERT ADD.** This will replace the RACDCERT GENCERT certificate.

RACDCERT GENREQ (2 of 2)

WARNING: Do not delete the self-signed certificate after the certificate request has been generated. You will lose the private key.

GOOD NEWS: V2R1, you can't delete the certificate which has been used for GENREQ unless you specify FORCE in the RACDCERT DELETE command

Certificate stored as a profile

- A certificate profile in the **DIGTCERT** class is created for a certificate added or created
  - Certificate
  - Private key (may or may not be present)
  - Last serial number used (initial value is 0)
  - More…
- This certificate profile represents the **certificate**, **NOT a protection profile**
  - Can not be managed by the resource management commands, like **RALTER**, **RDELETE**...
  - Managed though **RACDCERT** commands
  - Unlike the other protection files, the **owner** field in the certificate profile has no authority relevance
- There are specific profiles in the **FACILITY** class for **RACDCERT** authority checking

  IRR.DIGTCERT.<function>, eg.

  IRR.DIGTCERT**.GENCERT**

  IRR.DIGTCERT**.ADD** …

Certificate stored as a profile

- Certificate rings, and filters are also stored in RACF profiles (DIGTRING, DIGTNMAP)
- The **RACF User** profile contains information about certificates associated with the user. **DELUSER** will remove digital certificates associated with a user.
- Precaution needed for CERTAUTH certificate when you plan to preserve the certificate and the private key by exporting them in a pkcs12 package
    - When re-add this package, the last issued serial number value is not preserved, it starts with the initial value
    - For example, if this CA certificate has issued 100 certificates, the next certificate to be issued should have serial number 101; but after re-adding it, the certificate to be issued will have serial number 1, which is already used – all the certificates issued by the same CA should have a unique serial number!
- Before deleting CERTAUTH certificate, find out the last certificate's serial number it issued
- After re-adding, use R_Datalib's **IncSerialNum** function to bump up the serial number field to the appropriate number

Renewing a Certificate: With the Same Key Pair

Steps:

1) Create a new certificate request from the original certificate and save the request in a dataset 'request_dsn':

```
RACDCERT CERTAUTH GENREQ(LABEL('original
cert')) DSN(request_dsn)
```

2) Send the request to the original certificate CA

Warning: Don't delete the 'original cert'!!!

3) After you receive the new certificate and save it in a dataset 'cert_dsn', add it back under the same ID:

```
RACDCERT CERTAUTH ADD(cert_dsn)
```

Renewing a Certificate: With a New Key Pair (1 of 2)

- The longer a key pair is used, the more likely it is to be cracked. The key pair should be periodically changed. Two **RACDCERT** functions are provided:
- **RACDCERT REKEY**
  - Make a self-signed copy of the original certificate with a new public-private key pair
- **RACDCERT ROLLOVER**
  - Finalize the **REKEY** operation
  - Private key of the old certificate is deleted so that it may not be used again for signing or encryption
  - Cert with usage **PERSONAL**: all keyring occurrences of the old certificate will be replaced with the new one
  - Cert with usage **CERTAUTH** or **SITE**: the new cert will be added to all keyring occurrences of the old one

Renewing a Certificate: With a New Key Pair (2 of 2)

Steps:

1) Make a self copy of the original certificate

   RACDCERT ID(ftpid) REKEY(LABEL('original cert')) WITHLABEL('original cert2')

2) Create a certificate request from the copied certificate in step 1:

   RACDCERT ID(ftpid) GENREQ(LABEL('original cert2')) DSN(request_dsn)

3) Send the request to the original certificate CA

   Warning: Don't delete the 'original cert'!!!

4) After you receive the new certificate and save it in a dataset 'cert_dsn', add it back under the same ID:

   RACDCERT ID(ftpid) ADD(cert_dsn)

5) Roll over the original certificate to the new one:

   RACDCERT ID(ftpid) ROLLOVER(LABEL('original cert'))
   NEWLABEL('original cert2')

6) You may use RACDCERT ALTER to change the label back to the original one if you want

RACF Key Rings

- A key ring is a collection of certificates that **identify a networking trust relationship**. Key Rings are used to **identify the certificates required to establish a connection to a peer**.
- A certificate must be placed in a key ring before it can be used by middleware applications though the RACF **R_DataLib** callable service.
- Key Ring Syntax indicated on the application's configuration file:
  - **<ring owner-id>/<ring-name> or <ring-name> , eg**
    - **FTP.DATA : KEYRING** ftpid/ftpRing OR ftpRing
    - **AT-TLS policy : KEYRING SAF** tcpid/tcpRing OR tcpRing
  - **Recommendation:** to avoid confusion, specify the ring owner-id to _

- **Key Rings contain Certificate Usage –** assigned to a certificate when it is connected to a key ring indicates its intended purpose, default to the owner type, override by the USAGE keyword
  - **PERSONAL –** Used to identify a local server application. Personal usage must be used to get access to the private key.
  - **CERTAUTH –** Used to identify CA certificates to verify the peer entity's certificate
  - **SITE** – Certificate associated with an off-platform server or other network identity. SITE certificate bypass the normal certificate chain validation IF the validation application honors it.

Virtual Key Rings

- A Virtual Key Ring is a set of certificates which are associated with the certificate owner, but not connected to a 'real' RACF key ring.
- There are three types of virtual key rings:
    - **CERTAUTH** – All trusted CA certificates
        - Syntax: ***AUTH***/***
    - **SITE** – All site certificates
        - Syntax: ***SITE***/***
    - **User** - All certificates owned by a single user ID
        - Syntax: **<owning-id>**/*

- Most common usage is the **CERTAUTH** virtual key ring.
    - It is used when an application validates the certificates of others but has no need for its own certificate and private key.
    - **Example:** An FTP user who wants to establish a SSL encrypted connection to a FTP server. As long as the CA certificate which issued the FTP server's SSL certificate is a trusted CA certificate in RACF, the CERTAUTH virtual key ring can be used.

RACF Key Ring Protection

- RACF Key Rings are protected by resource profiles
- Two types of profiles are checked: **Ring Specific** or **Global**
- **Ring Specific** RDATALIB class profiles:
  - **<ring owner>.<ring name>.LST**
    - **READ** access – Retrieve all certificates and private ke
    - **UPDATE** access – Retrieve all certificates and private keys of others
    - **CONTROL** access – Retrieve all certificates and private keys of SITE and CA
  - **<virtual ring owner>.IRR_VIRTUAL_KEYRING.LST**
    - **READ** access – Retrieve all certificates and private keys of his own
    - **UPDATE** access – Retrieve all certificates and private keys of others
    - CA / SITE private keys can not be retrieved from a virtual key ring

RACF Key Ring Protection

- **Global** FACILITY class profiles:
  - **IRR.DIGTCERT.LISTRING**:
    - **READ** access – Retrieve certificates and private keys from one's own real or virtual key rings. Retrieve SITE and CA certificates (not private keys) from SITE and CA from their real or virtual key rings.
    - **UPDATE** access – Retrieve certificates from other user's real or virtual rings (not others user's private keys)
  - **IRR.DIGTCERT.LISTRING (READ) + IRR.DIGTCERT.GENCERT (CONTROL):**
    - Retrieve SITE and CA certificates and private keys from SITE and CA from their **real** key rings.
- **Note:** Private keys are only returned when certificate usage is **PERSONAL**
- **Remember:** If both types of profiles exist, the **Ring Specific profile** will be checked first.

Key Ring Setup: Server authentication

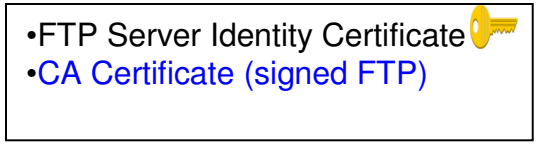
- **Example:** A client wants to establish a secure FTP connection between their workstation and an FTP server, but NOT use client authentication.
- **Client Key Ring:**
  - CA certificate which signed the FTP Server identity certificate
- **The FTP Server Key Ring:**
  - FTP Server Identity Certificate (with access to private key)
  - CA Certificate which signed the FTP Server Identity Certificate

Client Key Ring

- CA Certificate (signed FTP)

Note: No client certificate is needed. Use CERTAUTH's virtual key ring, no need to create a real key ring

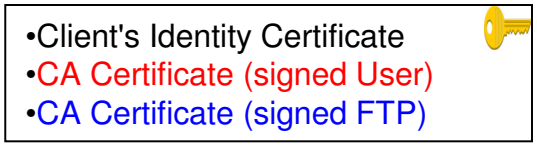
FTP Server Key Ring



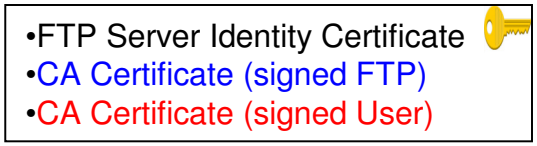
Key Ring Setup: Client authentication

- **Example:** A client wants to establish a secure FTP connection between their workstation and an FTP server and use client authentication to authenticate to the server.
- **Client Key Ring:**
  - Client Identity Certificate (with access to private key)
  - CA Certificate which signed the Client Identity Certificate
  - CA Certificate which signed the FTP Server Identity Certificate
- **The FTP Server Key Ring:**
  - FTP Server Identity Certificate (with access to private key)
  - CA certificate which signed the FTP Server Identity Certificate
  - CA certificate which signed the Client Identity Certificate

Client Key Ring



FTP Server Key Ring



One certificate to serve multiple servers

- Need a certificate for SSL handshake for two servers with domain names
  - server1.**ibm.com**
  - server2.**ibm.com**
- **Steps:**

1) Create a certificate with wild card in the Subject Distinguished Name

```
RACDCERT ID (SRV1) GENCERT SUBJECTSDN(CN('*.ibm.com')
 O('IBM')) WITHLABEL('Share Cert') SIGNWITH(CERTAUTH
  LABEL('CACert'))
```

2) Create a keyring under one ID, say SRV1

```
RACDCERT ID(SRV1) ADDRING(ShareRing)
```

3) Connect the cert to this ring

```
RACDCERT ID(SRV1) CONNECT(ID(SRV1) LABEL('Share Cert')
  RING(ShareRING) USAGE(PERSONAL) DEFAULT)
```

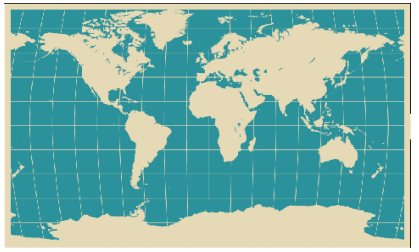
4) Permit both IDs to access the ring, the cert and the private key

```
PERMIT SRV1.SHARERING.LST CLASS(RDATALIB) ACCESS(READ) ID(SRV1)
PERMIT SRV1.SHARERING.LST CLASS(RDATALIB) ACCESS(UPDATE)ID(SRV2)
Note: The class RDATALIB must be RACLISTed
```

5) Indicate the keyring name in the application's configuration file, eg. FTP.DATA
   KEYRING SRV1/ShareRing

Certificate Mapping on z/OS

- Applications can call RACF to **map a digital certificate to a RACF user ID**
- **InitACEE** is the main RACF API for performing this mapping
- Some applications which can use these mappings:
  - **WAS**
  - **HTTP Server**
  - **FTP Server**
- Certificate Mapping options (evaluated in this order):
  - **One-to-one certificate to user ID association**
  - **Certificate Name Filtering (CNF)**
  - **Host Id Mapping extensions**



Certificate Mapping on z/OS: 1. One-to-one certificate to user ID association

- **Setup:**
  - Either generate (RACDCERT GENCERT) or add (RACDCERT ADD) a certificate to RACF.
- This establishes a **direct one-to-one mapping** between a certificate and a user ID.
- **Advantages:**
  - Simple – One certificate = one user id
- **Disadvantages:**
  - Administrative cost of this approach could be high if a large number of users is required

Certificate Mapping on z/OS: 2. Certificate Name Filtering

- **Setup:**
- Create the filter with the **RACDCERT MAP** command, eg
    - RACDCERT ID(VUSER) MAP IDNFILTER('OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet')…
- Filters can map a large number of certificates to a limited number of user Ids with little administrative cost.
- Appropriate when a large number of users need to be mapped to a single role, such as a group of bank tellers.
- **Auditing accountability remains** since the IDN/SDN in the end-entity's certificate will appear in SMF audit records.
- **Advantages:**
    - Less administrative setup for a large number of certificates
- **Disadvantages:**
    - Planning required

Certificate Mapping on z/OS: 3. Host Id Mappings extensions

- The hostIdMappings certificate extension is used to communicate the **end entity's user ID on a particular system**
- The extension contains a list of host name and user ID value pairs:
  - **userID1@hostName1.com**
  - **userID2@hostName2.com**
- **Setup:**
  - CA Cert must be marked **HIGHTRUST**
  - **SERVAUTH** class profile: IRR.HOST.<HOSTNAME>, where HOSTNAME= host name in thehostIdMappings extension
  - Permit Id which presents the certificate **READ access** to the above profile
  - If appID presents a certificate with ext value ftpID@server and appID has READ access to IRR.HOST.server, ftpID will be used to access the resource
- **Advantages:**
  - End entity certificates or filters need not be added to RACF
- **Disadvantages:**
  - Certificates can not be changed, therefore changes in user IDs will require a new certificate

Certificate Authority on z/OS: PKI Services

- **PKI Services** provides full certificate life cycle management
  - **Request**, **create**, **renew**, **revoke** certificates
  - Provides certificate status:
    - **Certificate Revocation List (CRL)**
    - **Online Certificate Status Protocol (OCSP)**
  - **Automatic notifications** or renewal of expiring certificates
- Generation and administration of certificates via customizable web pages
- Support **Simple Certificate Enrollment Protocol (SCEP)** for routers to request certificates automatically
- Support **Certificate Management Protocol (CMP)** for programmatic interface from other platforms

Certificate Life Cycle Planning (1 of 2)

- To set up a certificate for secure traffic the first time is **only the beginning**
- Must plan for the **certificate life cycle**
- Certificate expiration causes **application outage**
- Things to consider:
  - **How many** certificates are actively used in the system?
  - Categorize them:
    - Certs **locally created** VS Certs by **external provider**
    - Certs used to authenticate the incoming requests VS certs to identify your servers to the other parties
      - What CA certs will you trust?
      - Each server will have its own ring and own cert or shared?

Certificate Life Cycle Planning (2 of 2)

- If you are a local CA which issues certs to the other systems:
  - Who should be responsible to **keep track of the expiry date**? 'You' as the issuer or 'They' as the requestors?
    - When to **renew your CA** cert?
    - A 10 year validity CA cert should not issue 2 year validity cert after the 8th year

- How to **keep track of the expiration dates** of all the certificates in the system?
  - Spreadsheets?
  - Utilities?
  - Automation for renew?
  - Use certificate management vendor products?

Summary

- **What is a Digital Certificate?**
- RACF **RACDCERT** Command Overview
  - General tips
  - Generating a certificate request and renewing a certificate
- **RACF Key Rings:**
  - Real and Virtual Key Rings
  - Key Ring Protection
  - Certificate Sharing
  - Server Authentication
  - Client Authentication
- **Certificate Mapping on z/OS:**
  - One-to-one certificate to user ID association
  - Certificate Name Filtering (CNF)
  - Host Id Mapping extensions
- **z/OS Certificate Authority - PKI Services**
- **Certificate Life Cycle Planning**

References

- **IBM Education Assistant web site:**
  http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp
- **RACF web site:**
  http://www.ibm.com/servers/eserver/zseries/zos/racf
- **PKI Services web site:**
  http://www.ibm.com/servers/eserver/zseries/zos/pki
- **IBM Redbooks**
  - **z/OS V1 R8 RACF Implementation**
- **Security Server Manuals:**
  - **RACF Command Language Reference**
  - **RACF Security Administrator's Guide**
- **Cryptographic Server Manual**
  Cryptographic Services System Secure Sockets Layer Programming
- **RFCs**
  - **RFC2459 - Internet X.509 Public Key Infrastructure Certificate and CRL Profile**
  - **RFC5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile**

Questions ?

Session 15665