#SHAREorg

SHARE
Technology · Connections · Results

Digital Certificate Goody Bags on z/OS

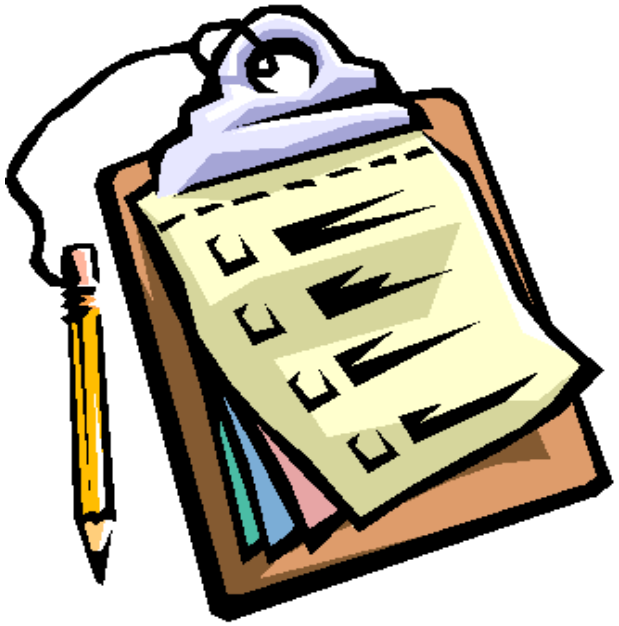
Ross Cooper, CISSP
IBM Corporation
RACF/PKI Development
Poughkeepsie, NY
Email: rdc@us.ibm.com

August 6th, 2012
Session 11623

SHARE in Anaheim 2012

Agenda

- **What is a Digital Certificate?**
- RACF **RACDCERT** Command Overview
  - RACDCERT CONNECT Tips
  - Tips for Generating a certificate request and renewing a certificate
  - Certificates stored as a profile
- **RACF Key Rings:**
  - Virtual Key Rings
  - Key Ring Protection
  - Sharing a Private Key with SITE
  - Key Ring exploiters
  - Server Authentication
  - Client Authentication
- **RACF Digital Certificate APIs**
- **Certificate Mapping on z/OS:**
  - One-to-one certificate to user ID association
  - Certificate Name Filtering (CNF)
  - Host Id Mapping extensions
- **PKI Services**



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 using RACF.
- **TSO command** shipped as part of **RACF**
- Command line interface with ISPF panels
- Certificates and Rings are protected by RACF profiles
- Learn more:
  - RACF Command Language Reference

```
                    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
```

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 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.
```

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 certificate information from an installed certificate
  - RACDCERT **ALTER** – Change certificate installation information
  - 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'))`

RACDCERT CONNECT

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

  RACDCERT ID(<ring-owner>) CONNECT(ID(<certificate-owner>) label...)
- **Which case has the exception?**
  - `RACDCERT ID(Mary) CONNECT(ID(John) LABEL…)`
    - Ring owner: Mary, Cert owner: John
  - `RACDCERT ID(Mary) CONNECT(LABEL…)`
    - Ring owner: Mary, Cert owner: Mary
  - `RACDCERT CONNECT(ID(John) LABEL…)`
    - Ring owner: Issuer of command, Cert owner: John
  - `RACDCERT CONNECT(LABEL…)`
    - Ring owner: Issuer of command, Cert owner: Issuer of command

RACDCERT GENREQ

- **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.

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

Renewing a Certificate: Same Key Pair

- **Eventually all certificates expire. To avoid application outages, certificate should be renewed before they expire.**
- **Renew a certificate with the original key pair:**
- **If the certificate is a self-signed certificate:**

  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) Create the new certifcate using the request in step 1:

     ```
     RACDCERT CERTAUTH GENCERT(request_dsn) SIGNWITH(CERTAUTH
     LABEL('original cert'))
     ```

- **If the certificate is not a self-signed certificate:**

  1) Same as step 1 above
  2) Send the request to the original certificate CA
  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)`

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

Renewing a Certificate: New Key Pair (1 of 3)

- **Renew a certificate with a new key pair**

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: New Key Pair (2 of 3)

- **Renew a certificate with a new key pair…**
- **If the certificate is a self-signed certificate:**

1) Make a self copy of the original certificate:

```
RACDCERT CERTAUTH REKEY(LABEL('original cert'))
WITHLABEL('original cert2')
```

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

```
RACDCERT CERTAUTH ROLLOVER(LABEL('original cert'))
NEWLABEL('original cert2')
```

Renewing a Certificate: New Key Pair (3 of 3)

- **Renew a certificate with a new key pair…**
- **If the certificate is not a self-signed certificate:**

1) Make a self copy of the original certificate

   RACDCERT ID(myid) REKEY(LABEL(‘original cert’)) WITHLABEL(‘original cert2’)

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

   RACDCERT ID(myid) GENREQ(LABEL(’original cert2’)) DSN(request_dsn)

3) Send the request to the original certificate CA

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

   RACDCERT ID(myid) ADD(cert_dsn)

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

   RACDCERT ID(myid) ROLLOVER(LABEL(‘original cert’)) NEWLABEL(‘original cert2’)

Certificate stored as a profile (1 of 2)

- A certificate profile in the **DIGTCERT** class is created for a certificate added or created
  - The profile name is in the form:

    <Certificate Serial #>.<Issuer's distinguished name>
  - Example:

    RACDCERT CERTAUTH GENCERT SUBJECTDN(OU(‘Master CA’) O(‘IBM’) C(‘US’)) WITHLABEL(‘MyCA’)

    Profile created: 00.OU=Master¢CA.O=IBM.C=US

    RACDCERT ID(testid) GENCERT SUBJECTDN(OU(‘Test Dept’) O(‘IBM) C(‘US’)) WITHLABEL(‘TestCert’) SIGNWITH(CERTAUTH LABEL(‘MyCA’))

    Profile created: 01.OU=Master¢CA.O=IBM.C=US
- Serial number of a self-signed certificate is 0
- Subsequent serial numbers will be incremented in order by 1
- Blanks in the DN are substituted with '**¢**' in the profile name
- If the CA's DN name is too long to be stored in a profile (246 characters), a hash of the name is used in the profile

Certificate stored as a profile (2 of 2)

- This profile represents the **certificate**, **NOT a protection profile**
  - The certificate profile can not be managed by the resource management commands, like **RALTER**, **RDELETE**...
  - Managed though **RACDCERT** commands
- There are specific profiles in the **FACILITY** class for **RACDCERT** authority checking
  - **IRR.DIGTCERT.<function>**
  - **IRR.DIGTCERT.GENCERT**
  - **IRRDIGTCERT.ADD** …
- 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.

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 for applications:
  - **<user-id>/<ring-name>**
- **Types of Certificates in RACF:**
  - **User** – Directly Associated with one z/OS user ID.
  - **CERTAUTH** – Trusted CA certificate used to verify the peer entity's certificate.
  - **SITE** – Certificates associated with an off-platform server or other network identity. SITE certificates bypass the normal certificate chain validation. Private keys can be shared.
- **Key Rings contain Certificate Usage –** The usage assigned to a certificate when it is connected to a key ring indicates its intended purpose.
  - **PERSONAL –** Used to identify a local server application. Personal usage must be used to get access to the private key.
  - **CERTAUTH –** Used to verify the peer entity's certificate. Used to identify the local server's CA certificate.
  - **SITE** – Certificate associated with an off-platform server or other network identity. SITE certificates bypass the normal certificate chain validation.

Virtual Key Rings

- A Virtual Key Ring is a set of certificates which are logically associated, 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**
  - **<virtual ring owner>.IRR_VIRTUAL_KEYRING.LST**
    - **READ** access – Read all certificates and own private key
    - **UPDATE** access – Read other user's private keys
    - **CONTROL** access – Read CA / SITE private keys
- **Global** FACILITY class profiles:
  - **IRR.DIGTCERT.LISTRING**:
    - **READ** access – Read own key rings and own private keys. Read SITE and CA Virtual key rings.
    - **UPDATE** access – Read other user's rings (Can not read others user's private keys)
  - **IRR.DIGTCERT.GENCERT:**
    - **CONTROL** access – Read CA / SITE private keys
- **Note:** Private keys are only returned when certificate usage is **PERSONAL**
- **Remember:** When switching from Global FACILITY class profiles to Ring Specific RDATALIB class profiles, the Ring Specific will be checked first.

Share a Private Key with SITE or CERTAUTH

- Applications can **share the private key** of a certificate which is added under **SITE** or **CERTAUTH**
- Create a keyring under one ID, say SRV1:

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

- Create a certificate under CERTAUTH or SITE, not a personal ID:

```
RACDCERT SITE GENCERT... WITHLABEL('Share Cert')
```

- Connect the cert to this ring:

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

- Permit both IDs to use this ring:

```
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ACCESS(READ) ID(SRV1)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ACCESS(UPDATE)
     ID(SRV2)
```

- Permit both IDs to use this private key:

```
RDEF FACILITY IRR.DIGTCERT.GENCERT UACC(NONE)
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ACCESS(CONTROL)
     ID(SRV1 SRV2)
```

- **Warning:** This access to these profiles allows these user IDs to access ANY private keys in SITE or CERTAUTH

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 **system 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?

z/OS Key Ring exploiters

Exploiter	Connect the server cert to the ring, eg. 'MYRING'	Where/How to specify the RACF Key Ring
FTP Server	RACDCERT ID(FTPSVR) CONNECT(LABEL('FTP Cert') RING(MYRING) DEFAULT) Note1	FTP.DATA file KEYRING MYRING or AT-TLS policy
TN3270 Server	RACDCERT ID(TNSVR) CONNECT(LABEL('TN Cert') RING(MYRING) DEFAULT) Note1	Telnet profile file KEYRING SAF MYRING or AT-TLS policy
IP Security (IPSEC)	RACDCERT ID(IPSEC) CONNECT(LABEL('IPSEC Cert') RING(MYRING) DEFAULT) *Note1*	Iked.conf file KEYRING MYRING or AT-TLS policy
HTTP Server	RACDCERT ID(WEBSVR) CONNECT(LABEL('WEB Cert') RING(MYRING) DEFAULT) *Note: must be connected as default*	httpd.conf file Keyfile MYRING SAF
Websphere MQ	RACDCERT ID(QM1) CONNECT(LABEL ('ibmWebSphereMQMQ1') RING(MYRING)) *Note: label of the cert must start with 'ibmWebSphereMQ'*	MQ command ALTER QMGR SSLKEYR (MYRING)

Key Ring Setup: Server authentication

- **Example:** A user wants to establish a secure FTP connection between their workstation and an FTP server, but NOT use client authentication.
- **User Key Ring:**
  - CA certificate which signed the FTP Server identity certificate
  - **Notes:**
    - No End entity certificate required
      - *(Other authentication method used such as User ID & Password)*
    - No Private keys required
    - On z/OS the CERTAUTH Virtual Key Ring can be used if the FTP server is signed by a CERTAUTH certificate
- **The FTP Server Key Ring:**
  - FTP Server Identity Certificate (with access to private key)
  - CA Certificate which signed the FTP Server Identity Certificate

User Key Ring

- CA Certificate (signed FTP)

FTP Server Key Ring

- FTP Server Identity Certificate
- CA Certificate (signed FTP)

Key Ring Setup: Client authentication

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

User Key Ring

- User's Identity Certificate
- CA Certificate (signed User)
- CA Certificate (signed FTP)

FTP Server Key Ring

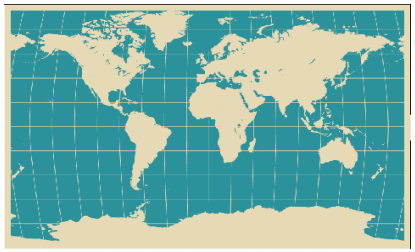
- FTP Server Identity Certificate
- CA Certificate (signed FTP)
- CA Certificate (signed User)

RACF Digital Certificate APIs

- Applications can get access to digital certificates though APIs
- **Java RACF KeyStore:** Allows Java programs to access RACF Key Rings
- **System SSL:** Allows UNIX applications to access RACF Key Rings
- **R_DataLib Callable service:** The lowest level API used by applications on z/OS to access RACF keyrings.
- **R_DataLib Functions:**
  - **DataGetFirst / DataGetNext** – Return certificates from a RACF keyring.
  - **CheckStatus** – Get certificate trust status
  - **IncSerialNum** – Increment a CA certificate's last used serial number
  - **NewRing** – Create a key ring
  - **DelRing** – Delete a key ring
  - **DataPut** – Add a certificate to RACF and connect to key ring
  - **DataRemove** – Remove a certificate from a key ring and/or from the Database

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: One-to-one certificate to user ID association

- When a certificate is either generated (RACDCERT GENCERT) or added to RACF, it is registered to a user ID and added to the RACF database.
- This establishes a **direct one-to-one mapping** between a certificate and a user ID.
- Certificates added to RACF are stored in certificate profiles in the DIGTCERT class. Can optionally contain the private key, or a link to the private key in ICSF.
- **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: Certificate Name Filtering

- Associates **many certificates with one user ID** based on filters covering portions of the subject's and/or issuer's distinguished names in the certificate.
- Filters can map a large number of certificates to a limited number of user Ids with little administrative cost.
- Filters are created with the **RACDCERT MAP** command
- 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: Certificate Name Filtering - Example

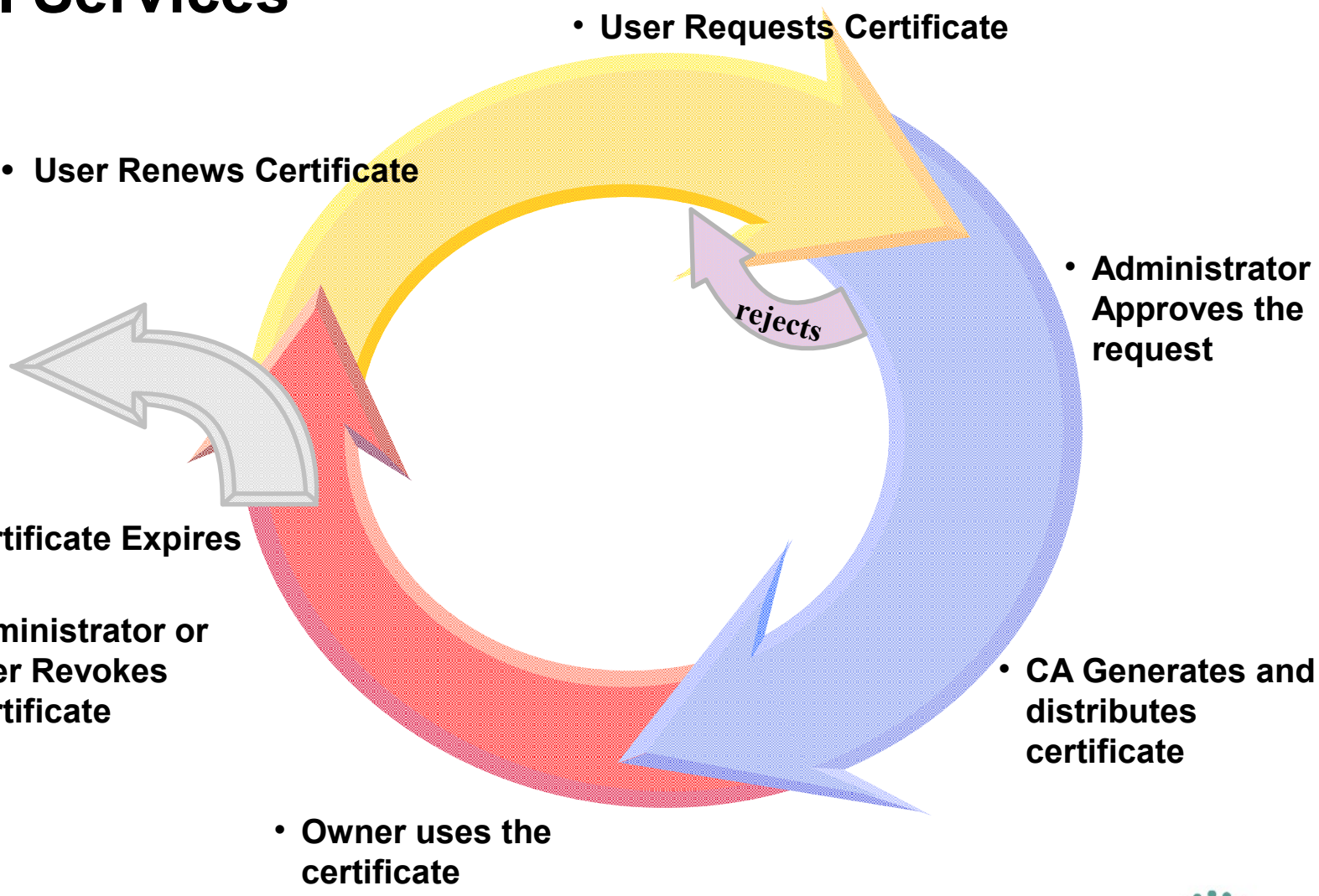
- **End Entity Certificate:**
  - **SDN**: CN=Ross Cooper,OU=Bank Tellers,O=Big Bank,C=US
  - **IDN**: CN=Some CA Root, OU=Some CA,O=Some CA Inc,C=US
- **Filter:**
  - RACDCERT ID(**BANKT**) MAP
    SDNFILTER('OU=Bank Tellers,O=Big Bank,C=US')
    IDNFILTER('CN=Some CA Root, OU=Some CA,O=Some CA Inc,C=US')
- **Search Order:**

1) Subject's-full-name.issuer's-full-name:
   CN=Ross Cooper,OU=Bank Tellers,O=Big Bank,C=US.CN=Some CA Root, OU=Some CA,O=Some CA Inc,C=US
2) Subjects-partial-name.issuer's-full-name:
   OU=Bank Tellers,O=Big Bank,C=US.CN=Some CA Root, OU=Some CA,O=Some CA Inc,C=US
3) Subject-full-name:
   CN=Ross Cooper,OU=Bank Tellers,O=Big Bank,C=US
4) Subjects-partial-name:
   OU=Bank Tellers,O=Big Bank,C=US
5) Issuer's-full-name:
   CN=Some CA Root, OU=Some CA,O=Some CA Inc,C=US
6) Issuer's-partial-name:
   OU=Some CA,O=Some CA Inc,C=US

Certificate Mapping on z/OS: 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**
- RACF uses the extension to find the local system's host name and then determine the local user ID for the ACEE
- **Setup:**
  - CA Cert must be marked **HIGHTRUST**
  - Host name **matches SERVAUTH** class profile: IRR.HOST.<HOSTNAME>
  - Id which presents the certificate must have **READ access** to the SERVAUTH class profile
- **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



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)**
  - Generation and administration of certificates via customizable web pages
  - Support **Simple Certificate Enrollment Protocol (SCEP)** for routers to request certificates automatically
  - **Automatic notifications** or renewal of expiring certificates

Review

- **What is a Digital Certificate?**
- RACF **RACDCERT** Command Overview
  - RACDCERT CONNECT Tips
  - Tips for Generating a certificate request and renewing a certificate
  - Certificates stored as a profile
- **RACF Key Rings:**
  - Virtual Key Rings
  - Key Ring Protection
  - Sharing a Private Key with SITE
  - Key Ring exploiters
  - Server Authentication
  - Client Authentication
- **RACF Digital Certificate APIs**
- **Certificate Mapping on z/OS:**
  - One-to-one certificate to user ID association
  - Certificate Name Filtering (CNF)
  - Host Id Mapping extensions
- **PKI Services**

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 11623