The Digital Certificate Journey from RACF to PKI Services Part 1

Session J9
May 11th 2005

Wai Choi
IBM Corporation
RACF Development
Poughkeepsie, NY

Phone: (845) 435-7623
e-mail: wchoi@us.ibm.com

Trademarks

- **The following are trademarks or registered trade marks of the International Business Machines Corporation:**
  - DB2
  - CICS
  - OS/390
  - RACF
  - S/390
  - z/OS
- **UNIX is a registered trademark of The Open Group in the United States and other countries.**

Agenda

- Public Key Cryptography / Digital certificate Basics

- Basic RACF Support for Certificates

- Sophisticated RACF Support for Certificates

- Some common operations
  1. Renewing certificate with or without new key pair
  2. Sharing a certificate between two servers

Public Key Cryptography Overview

- Secret key cryptography encrypts and decrypts with the same key

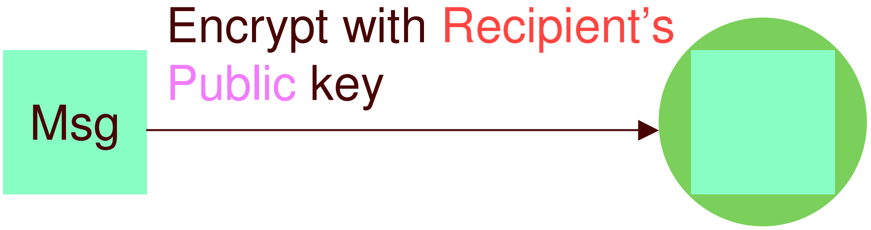
- Public Key cryptography involves a public-private key pair, encrypts with one key and decrypts with its partner key

- To facilitate 2 main goals of secure communication – confidentiality and integrity

Encryption (for confidentiality) Under Public Key Cryptography

Encrypting a message:

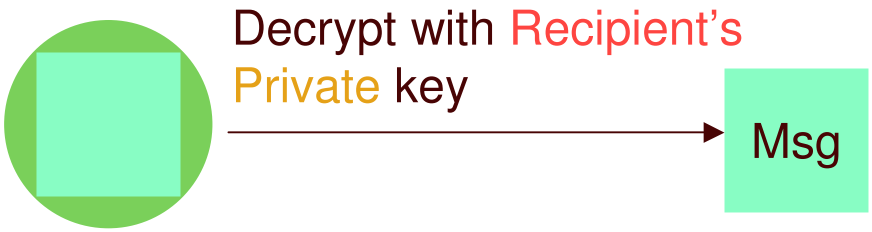
Sender:



Encrypt with Recipient's Public key

Decrypting a message:

Recipient:

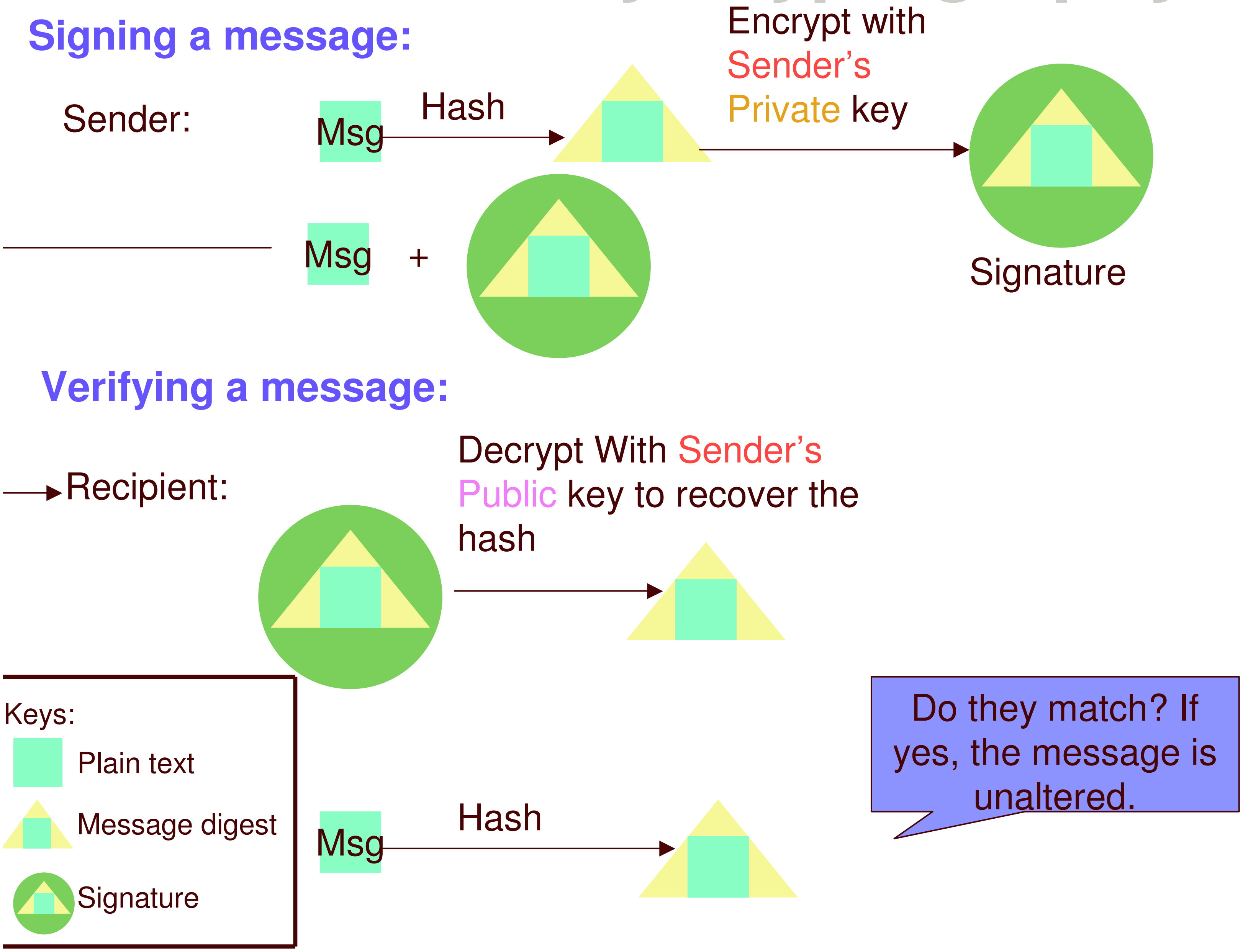


Decrypt with Recipient's Private key

Keys:

- 🟩 Plain text
- 🟢 Encrypted text

Signing (for integrity) Under Public Key Cryptography



Signing a message:

Sender:

Msg → Hash

Encrypt with Sender's Private key

Signature

Msg +

Verifying a message:

Recipient:

Decrypt With Sender's Public key to recover the hash

Msg → Hash

Do they match? If yes, the message is unaltered.

Keys:

🟩 Plain text

▲ Message digest

● Signature

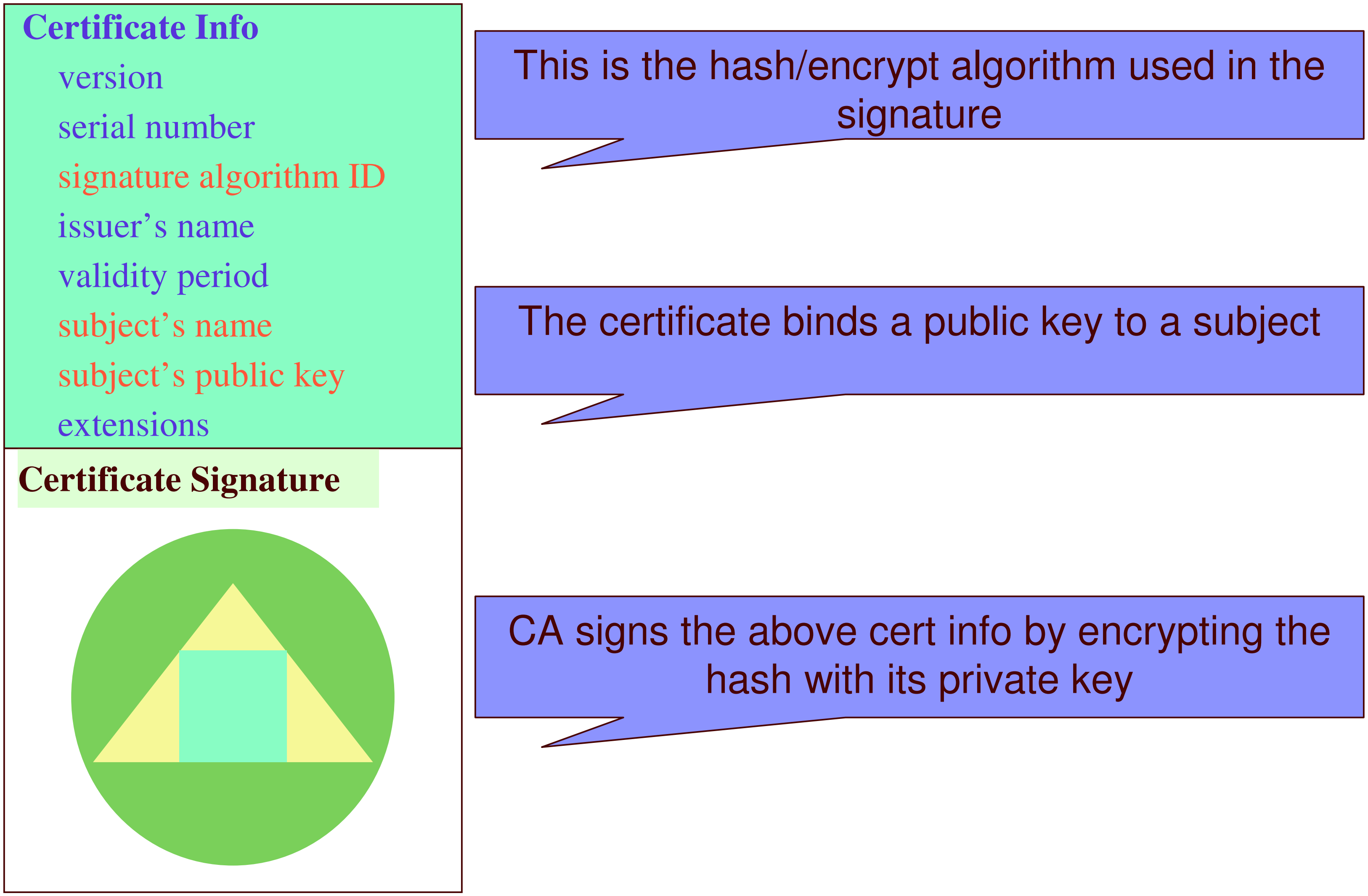
Public key and Certificate

- Public key alone can not tell who the key owner is

- Need a trusted third party, Certificate Authority (CA), to bind a public key to a subject through a certificate

- The Certificate Authority signs the certificate with its private key to prove its authenticity

What’s inside a Certificate?



Certificate Info

- version
- serial number
- signature algorithm ID
- issuer’s name
- validity period
- subject’s name
- subject’s public key
- extensions

Certificate Signature

This is the hash/encrypt algorithm used in the signature

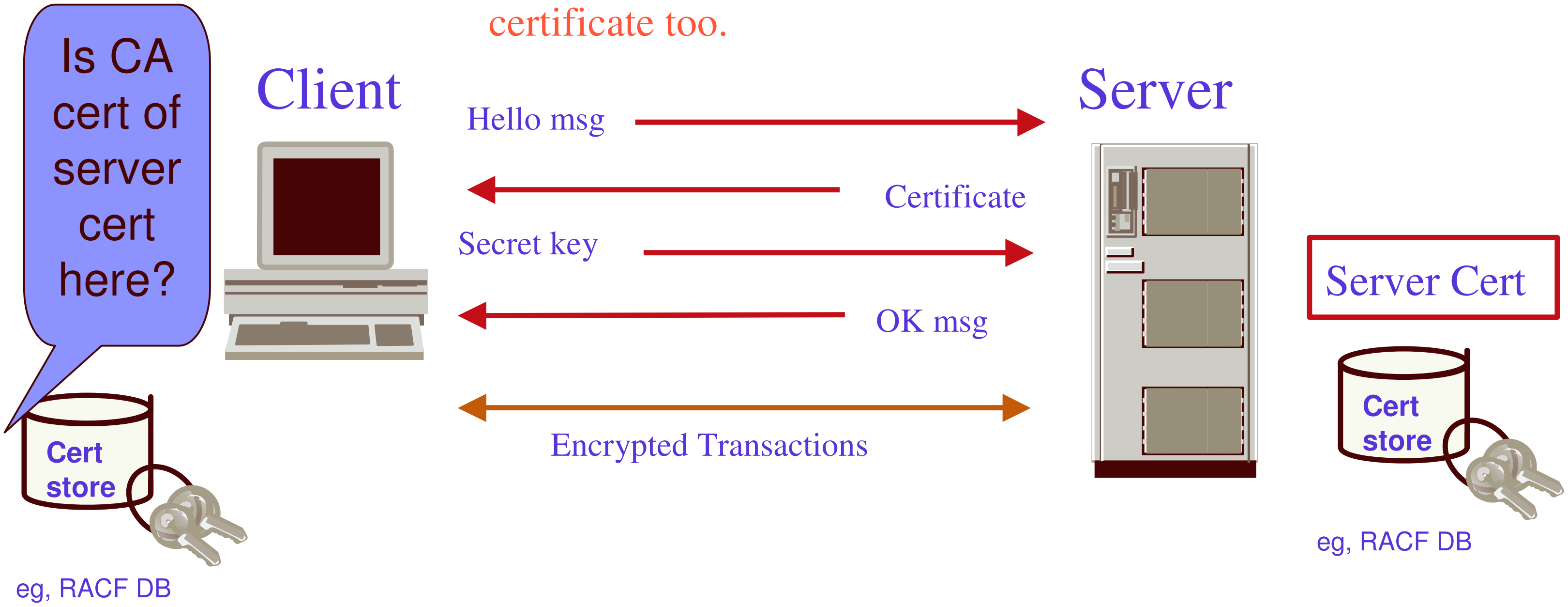
The certificate binds a public key to a subject

CA signs the above cert info by encrypting the hash with its private key

A common use of Certificate - handshake

1. Client sends a ‘hello’ msg to server
2. Server sends its certificate to client
3. Client validates the server’s certificate
4. Client encrypts a secret key with server’s public key and sends it to server
5. Server decrypts the secret key with its private key
6. Server encrypts a ‘handshake OK’ msg with the secret key and sends it to client
7. Client trusts server, business can be conducted

* Note the above steps illustrate server authentication. For client authentication, server needs to validate client’s certificate too.



A Quiz

Given:

- CA1 is the CA cert which signed the server cert S
- CA2 is the CA cert which signed the client cert C
- Ring X is the server's key ring, ring Y is the client's key ring

Question:

What cert(s) needed in ring X? in ring Y?

1. **For Server authentication**

Ring X: CA1, S   Ring Y: CA1

2. **For Client authentication (implies server authentication too)**

Ring X: CA1, S, CA2   Ring Y: CA2, C, CA1

Further thinking:

Would it be simpler (for which case?) if both the server and client certs were signed by the same CA cert, say CA1? How do the rings look like?

RACF support for Certificates

1. Certificate generation – RACDCERT GENCERT, GENREQ

- **RACDCERT GENCERT:**
  - GENerates a CERTificate, and optionally public-private key
  - Can utilize z/OS hardware crypto for private key generation, storage and operation
  - Certificate can be self signed or signed by another certificate's corresponding private key
  - May create certificates for other servers, includes those on non-z/OS platform

RACF support for Certificates…

- RACDCERT GENCERT examples:

►Create a self signed certificate and public-private key pair using ICSF for a CA -- CERTAUTH represents the ‘ID’ of a CA, no ‘SIGNWITH’ needed for self signed cert

```
–RACDCERT CERTAUTH GENCERT SUBJECT(...) ICSF…
```

►Create a certificate and public-private key pair using PCICC for ID WEBSRV, signed with a CA certificate named ‘theCA cert’

```
–RACDCERT ID(WEBSRV) GENCERT SUBJECT(…) PCICC
 SIGNWITH(CERTAUTH LABEL(‘theCA cert’)…
```

►Create a certificate based on a certificate request specified in the dataset ‘CERTREQ.B64’ for ID MYID – you get the request from other system and you want your local CA to sign it. This generates certificate only, no key pair is created.

```
–RACDCERT ID(MYID) GENCERT(‘CERTREQ.B64’)
 SIGNWITH(CERTAUTH LABEL(‘theCA cert’)…
```

RACF support for Certificates…

- ## RACDCERT GENREQ:
  - GENerates a REQuest by copying information, including the public key, from a previously created certificate
  - Usually issued after GENCERT a self signed certificate
  - The request is signed with the private key associated with the specified certificate
  - The request is saved to a data set in the Base64 format which can be used in cut and paste
  - The created certificate request can be submitted (e.g. e-mail it, paste it into a web page, etc) to a CA for issuance
  - Note there is no key generation in GENREQ
  - RACDCERT GENREQ example:

►Generate a request based on an existing certificate named ‘My Self Signed Cert’ in RACF for ID OUTSRV and put the request in the data set ‘CERTREQ.B64’

–
```
RACDCERT ID(OUTSRV) GENREQ(LABEL(‘My Self Signed
Cert')) DSN(CERTREQ.B64)
```

An example of a Base64 encoded certificate request

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBfzCB6QIBADAQMQ4wDAYDVQQDEwV0ZXN0YTCBnzANBgkqhkiG9w0BAQEFAAOB
jQAwgYkCgYEA3qy4qFTb97+kefbgMysxIXpaRQVwqT0I2XeDmI0WmXF6GkvK+i4k
7wr/pto+cGtqCzHsQrm6aRKiJF6pdizYkf4xew0DqdeVArOydr/4HESVNlJRqxJ/
jqY4IJ0uphnsbKKyfl7ny77u4M50YZBXRGq9VDAFpCaQbNW8xVkIUPECAwEAAaAw
MC4GCSqGSIb3DQEJDjEhMAAwHQYDVR0OBBYEFFX5QcyxyCL6q+NFFGjQpoCnP9mB
MA0GCSqGSIb3DQEBBQUAA4GBAMyKoRZvGJAyVPummMMiRJgMQ4KMcYrraI79rz7L
SWAq5/lPpbkue9enn1xaS3eJZOQNXGaVk4Rem3rGM740/PpQIF/qMN1pJZfOEzyL
rYxIaO6riEXM3Q2Y80m6C+X+Vk69eRClLTvc8I5l5uz+EMCTd5x5PaGuzhXgjxkE
Q5vt
-----END NEW CERTIFICATE REQUEST-----
```

RACF support for Certificates…

2. Certificate installation – RACDCERT ADD, ADDRING, CONNECT

- **RACDCERT ADD:**
  - **Install a certificate to RACF with or without private key**

RACF support for Certificates...

- RACDCERT ADD examples:

► Add a trusted CA certificate under ID CERTAUTH

```
–RACDCERT CERTAUTH ADD(‘dataset containing CA cert’)
 TRUST…
```

► Add a trusted peer certificate under ID SITE

```
–RACDCERT SITE ADD(‘dataset containing site cert’)
 TRUST…
```

► Add a certificate package that contains the private key under ID myid

```
–RACDCERT ID(myid) ADD(‘dataset containing my cert’)
 password(‘secret’)…
```

► Replace a previous self signed certificate under ID OUTSRV

```
–RACDCERT ID(OUTSRV) ADD(‘dataset containing a CA
 signed cert issued for the request based on the self
 signed cert’) TRUST
```

- GENCERT a self signed cert
- GENREQ based on the self signed cert
- Send request to CA and get back a new cert
- ADD the new cert back under the same ID

RACF support for Certificates…

2. Certificate installation – RACDCERT ADD, ADDRING, CONNECT

- **RACDCERT ADDRING**
  - Create a key ring (for handshake process)
  - Certificate must be placed in a key ring before it can be used by other middleware, e.g. SSL
  - RACDCERT example:
  - ►Create a key ring called ‘SSLring’ for ID WEBSRV
    - `RACDCERT ID(WEBSRV) ADDRING(SSLring)`
- **RACDCERT CONNECT**
  - Place a certificate in a key ring
  - You may place your own certificates and other’s certificate in your key ring
  - The TRUST status of the connected certificate in the ring tells if the certificate can be used

RACF support for Certificates…

- A certificate can be placed in more than one key ring
- The usage of the certificate in the ring depends on the owner ID, unless overridden by the USAGE keyword:
  - Ordinary ID - PERSONAL
  - CERTAUTH - CERTAUTH
  - SITE – SITE
- You can pick one of the certificates to be the DEFAULT, some applications will depend on this attribute to indicate the signing certificate
- RACDCERT CONNECT example:

►Connect WEBSRV’s own certificate called ‘SSL cert’ to WEBSRV’s key ring called SSLring

```
–RACDCERT ID(WEBSRV) CONNECT(LABEL(‘SSL cert’)
 RING(SSLring) DEFAULT…)
```

RACF support for Certificates…

3. Certificate administration – RACDCERT LIST, ALTER, DELETE, REMOVE, …

- **RACDCERT LIST:**
  - Display certificate information for a userid
- **RACDCERT ALTER**
  - Change the TRUST/NOTRUST status or the label of a certificate
- **RACDCERT DELETE**
  - Remove a certificate from RACF
- **RACDCERT REMOVE**
  - Remove a certificate from a key ring
- **And more…**

Another common use of Certificate – Log in a system without userid/password

- Handshake process requires one’s own certificate and the issuer’s certificate of the communicating party in each other’s certificate store/key ring in the system.

- The system in which the user logs in needs to have the user’s certificate installed.

- It will be a huge administrative burden to add and manage thousands of user certificates for all the users who use certificate to log in.

More Sophisticated Support from RACF

- Solution 1: Certificate Name Filtering
  - Supported by RACF

- Solution 2: HostIdMapping
  - Supported by RACF and PKI Services

- No user certificate needs to be installed.

More Sophisticated Support from RACF…

- **Certificate Name Filtering – RACDCERT MAP**
  - Create the definition of a set of rules (‘filters’) based on the subject’s or issuer’s distinguished names (or both)
  - Can map one or more certificates to a filter
  - Examples:

►Create a filter to associate ID VUSER to any user presenting a certificate issued by Verisign Class 1 Individual Subscriber

```
–RACDCERT MAP ID(VUSER) IDNFILTER(‘OU=Verisign
Class 1 Individual Subscriber.O=Verisign,
Inc.L=Internet’)…
```

►Create a filter to associate ID RACFGP to any user presenting a certificate with subject’s distinguished name OU=RACF.O=IBM

```
–RACDCERT MAP ID(RACFGP) SDNFILTER
(‘OU=RACF.O=IBM’)…
```

More Sophisticated Support from RACF…

- **HostIdMapping**
  - A client can present a certificate containing a HostIdMapping extension to the server
  - This extension contains a host name and a subject id
  - RACF will honor this extension if
    - the issuing CA cert is marked HIGHTRUST
    - the host name in the extension matches a profile IRR.HOST.<host name> in the SERVAUTH class
  - PKI Services can create this extension

Common exploiters of certificates on z/OS

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)	FTP.DATA file KEYRING MYRING
TN3270 Server	RACDCERT ID(TNSVR) CONNECT(LABEL('TN Cert') RING(MYRING) DEFAULT)	PROFILE.TCPIP file KEYRING SAF MYRING
HTTP Server	RACDCERT ID(WEBSVR) CONNECT(LABEL('WEB Cert') RING(MYRING) 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)

Two ways to renew a certificate

- ## Renew a certificate with the original key pair

  - Eventually a certificate will expire. To avoid complications, you should renew it before it expires.
  - Steps:
  - **If the certificate is a self-signed certificate:**

    1. Create a new certificate request from your 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(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)
       ```

Two ways to renew a certificate…

- **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

Two ways to renew a certificate…

- ## Renew a certificate with a new key pair…

  - **Steps:**
  - **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’)
```

Two ways to renew a certificate…

- ## 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’)
```

How to share a certificate’s private key in a keyring

- Create a certificate under CERTAUTH or SITE

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

- Setup permission for the two IDs to use this private key

```
RDEF FACILITY IRR.DIGTCERT.GENCERT UACC(NONE)
```

```
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ACCESS(CONTROL) ID(SRV1 SRV2)
```

- Create a keyring under one ID, say SRV1

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

- 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)
```

References

- PKI Services web site:

http://www.ibm.com/servers/eserver/zseries/zos/pki

- PKI Services Red Book:

http://www.redbooks.ibm.com/abstracts/sg246968.html

- RACF web site:

http://www.ibm.com/servers/eserver/zseries/zos/racf

- Cryptographic Services
  - PKI Services Guide and Reference (SA22-7693)
  - OCSF Service Provider Developer's Guide and Reference (SC24-5900)
  - ICSF Administrator's Guide (SA22-7521)
  - System SSL Programming (SC24-5901)
- Security Server Manuals:
  - RACF Command Language Reference (SC28-1919)
  - RACF Security Administrator's Guide (SC28-1915)
  - RACF Callable Services Guide (SC28-1921)
  - LDAP Administration and Use (SC24-5923)
- IBM HTTP Server Manuals:
  - Planning, Installing, and Using (SC31-8690)
- Other Sources:
  - PKIX - http://www.ietf.org/html.charters/pkix-charter.html

Questions???

See you in the next session J10 for the next part of the journey – PKI Services

Disclaimer

- The information contained in this document is distributed on as "as is" basis, without any warranty either express or implied. The customer is responsible for use of this information and/or implementation of any techniques mentioned. IBM has reviewed the information for accuracy, but there is no guarantee that a customer using the information or techniques will obtain the same or similar results in its own operational environment.
- In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used. Functionally equivalent programs that do not infringe IBM's intellectual property rights may be used instead. Any performance data contained in this document was determined in a controlled environment and therefore, the results which may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.
- It is possible that this material may contain references to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM Products, programming or services in your country.
- IBM retains the title to the copyright in this paper as well as title to the copyright in all underlying works. IBM retains the right to make derivative works and to republish and distribute this paper to whomever it chooses.