

Understanding Digital Certificates on z/OS

SHARE Boston, MA
August 4, 2010

Saheem Granados, CISSP
IBM Corporation
LDAP Development & Design
Poughkeepsie, NY

e-mail: sgranado@us.ibm.com

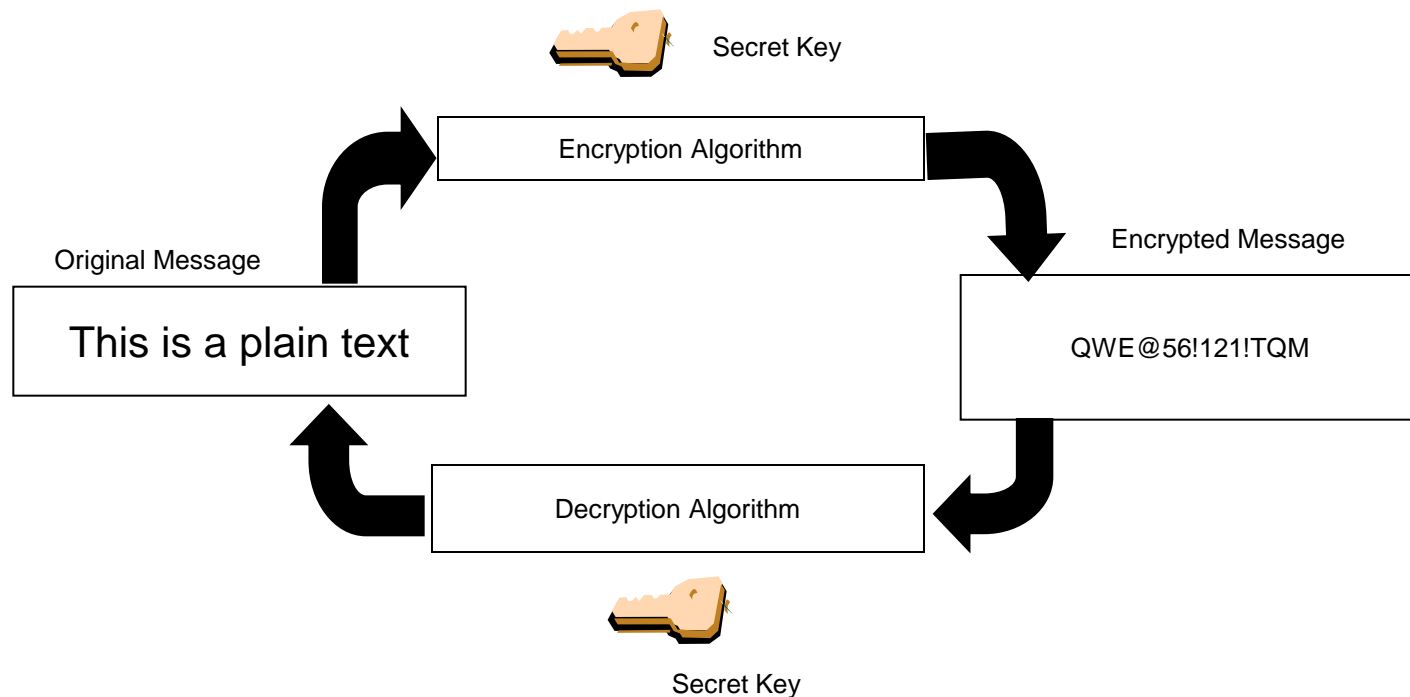


Agenda

- **Symmetric vs. Asymmetric Encryption**
- **What are digital certificates**
- **Certificate types and contents**
- **Overview of certificate utilities available on z/OS**
- **Certificate formats**
- **Summary**

Symmetric Encryption

- Defined keys
- Provide data confidentiality
- Algorithm defines strength of the encryption – DES, Triple DES, AES etc



Asymmetric Encryption

- **Public/private key pairs**
- **A public key and a related private key are numerically associated with each other.**
- **Provide data confidentiality, integrity and non repudiation**
- **Data encrypted/signed using one of the keys may only be decrypted/verified using the other key.**
- **Public key is intended to be given freely**
- **Private key needs to be treated very securely and not distributed**

Encryption (for confidentiality)

Encrypting a message:




Decrypting a message:



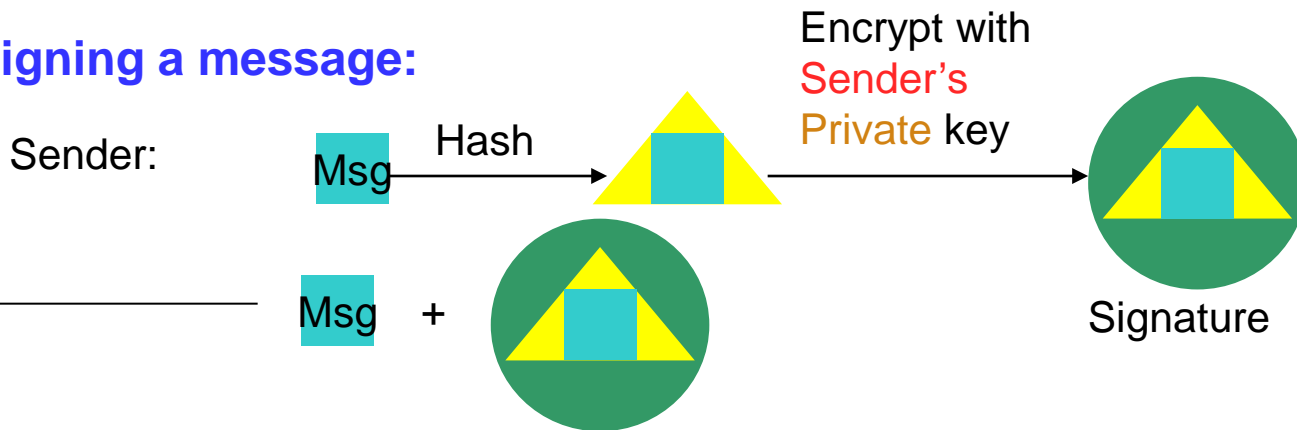
Keys:

 Plain text

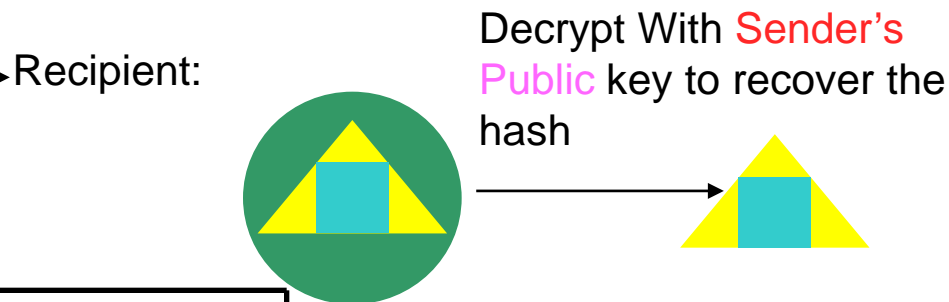
 Encrypted text

Signing (for integrity and non repudiation)

Signing a message:



Verifying a message:



Do they match? If yes, the message is unaltered. Assuming the hashing algorithm is strong. MD5 was demonstrated 'broken' on 12/30/08

Keys:

- Plain text
- Message digest
- Signature

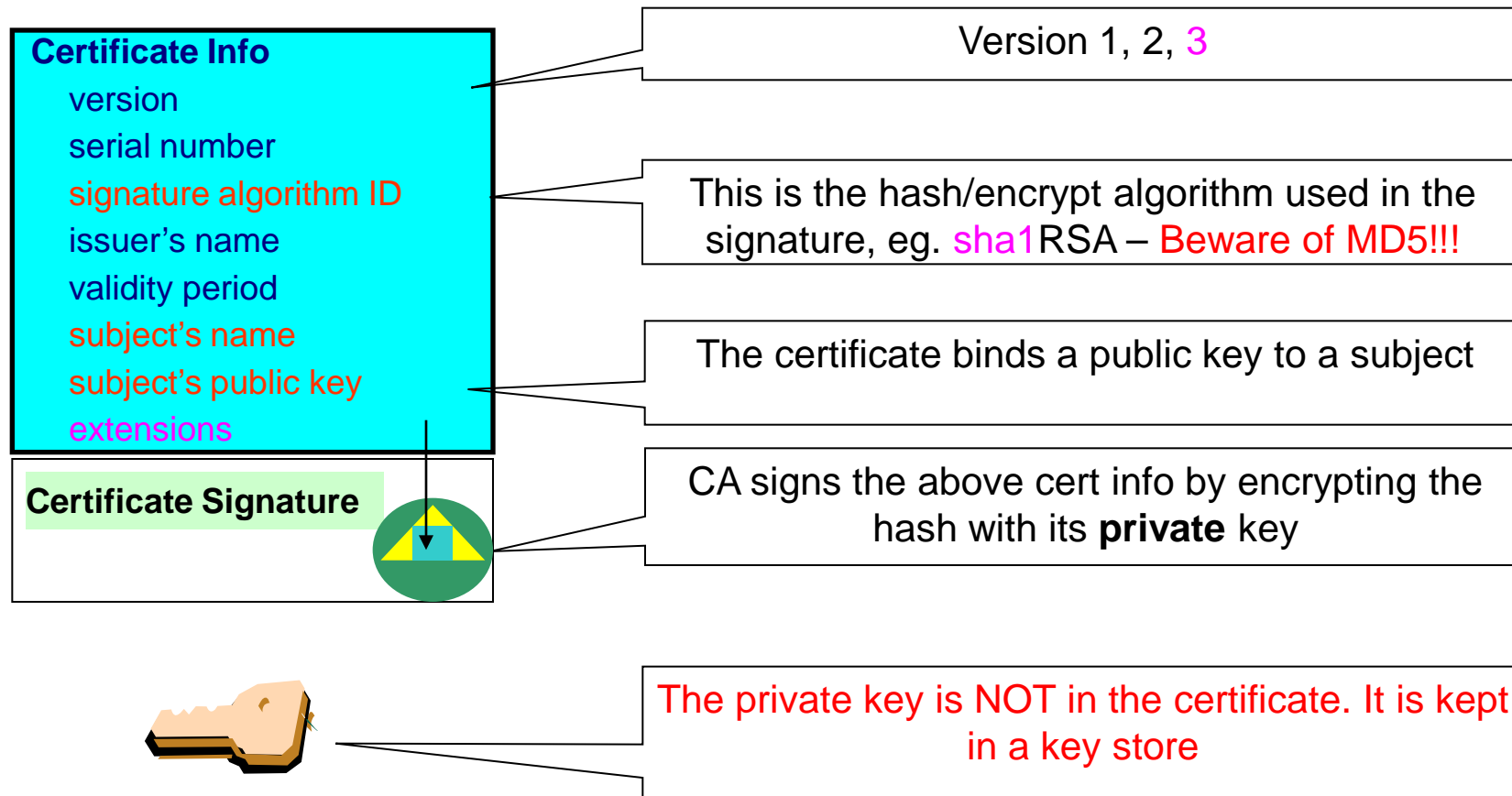
What is a Digital Certificate

- **Best way to think of it is as an ID card, like driver licenses, passports**
- **To establish your identity or credential to be used in electronic transactions**
- **Digital certificates been in existence for over 20 years**
- **It binds public key information to your identity to be used by applications that are based on public key protocols. (e.g. SSL/TLS)**
- **Generally digital certificates provide identity to a person or a server**

What is a Digital Certificate

- **Issued by a trusted third party called Certificate Authority (CA) that can ensure validity**
- **Packaging of the information is commonly known as the x.509 digital certificate. X.509 defines the format and contents of a digital certificate.**
 - IETF RFC 5280
- **Have evolved over time to not only bind basic identity information to the public key but also how public key can be used, additional identity data, revocation etc.**

What's inside a Certificate?



You can NOT change ANY of the certificate information!

Extensions of a x.509 digital Certificate

- Adds additional definitions to a certificate and its identity information
- 15+ currently defined
- Top 6 extensions of interest
 - Authority Key Identifier
 - Subject Key Identifier
 - Key Usage
 - Subject Alternate Name
 - BasicConstraints
 - CRL Distribution Point

Extensions of a x.509 digital Certificate

- **Authority Key Identifier** – Unique identifier of the signer
- **Subject Key Identifier** – Unique identifier of the subject
- **Key Usage** – defines how the public key can used
 - Digital Signature
 - Key Encipherment
 - Key Agreement
 - Data Encipherment
 - Certificate Signing
 - CRL signing
- **Subject Alternate Name** – additional identity information
 - Domain name
 - E-mail
 - URI
 - IP address
- **Basic Constraints** – Certificate Authority Certificate or not
- **CRL Distribution** – Locating of Revoked certificate information

Example of a x.509 digital Certificate

Certificate issued to Server x by CA MyCompany CA to be used for SSL/TLS communication

Version	V3
Serial Number	150
Signature Algorithm	RSA with SHA1
Issuer	CN=MyCompany CA,OU=Onsite CA ,O=CA Company,C=US
Validity	
From	Wednesday, May 31, 2008 10:41:39 AM
To	Wednesday, May 31, 2009 10:41:39 AM
Subject	CN=Server x,OU=z/OS,O=IBM,ST=New York,C=US
Public Key	RSA (1024)
Extensions	
Key Usage	Digital Signature, Key Encipherment
Authority Key Identifier	8014 91C1 73B0 73D5 D992 7467 CD1B F151 1434 31B6 2C5A
Subject Key Identifier	0414 7CA8 9E87 AA37 5D70 0301 7FDA 996C 1238 A20D 4FDE
Basic Constraints	Certificate issued to a certificate authority= FALSE
Subject Alternate Name	IP Address=9.1.2.3

Types of digital certificates

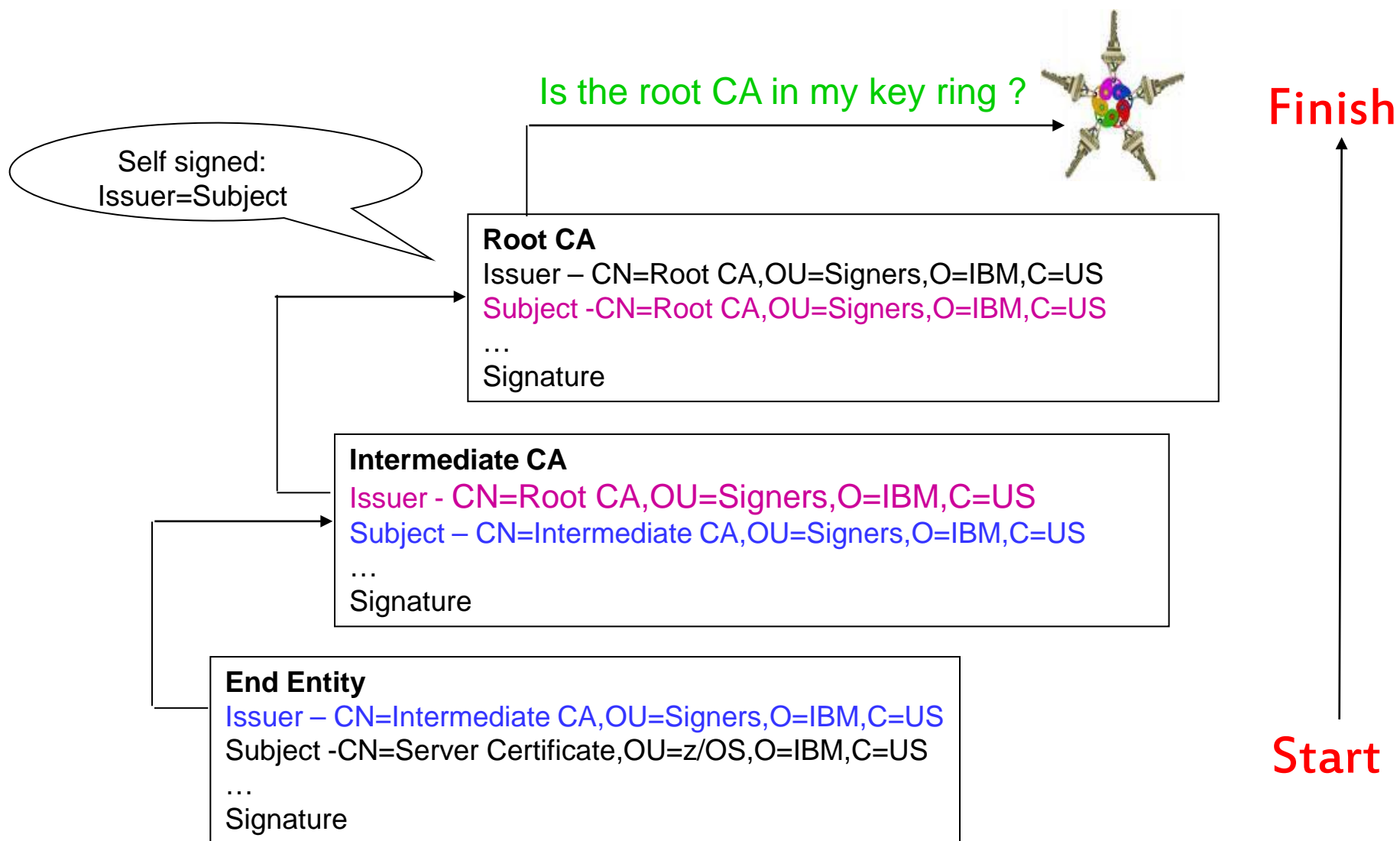
- **Self signed**

- Self-issued
- Issuer and subject names identical
- Signed by itself using associated private key

- **Signed Certificates**

- Signed/issued by a trusted Certificate Authority Certificate using its private key.
- By signing the certificate, the CA certifies the validity of the information. Can be a well-known commercial organization or local/internal organization.

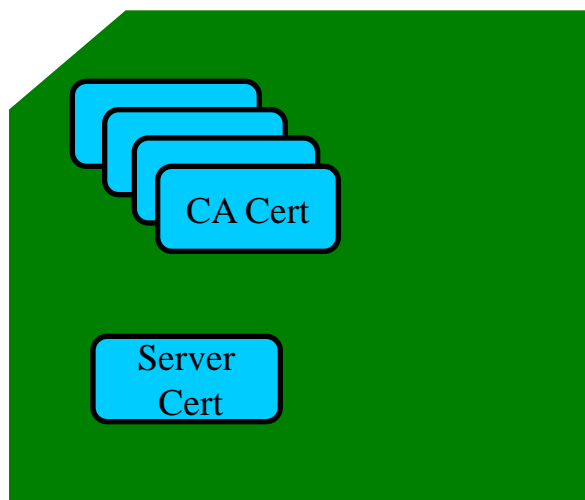
Certificate Chain Validation



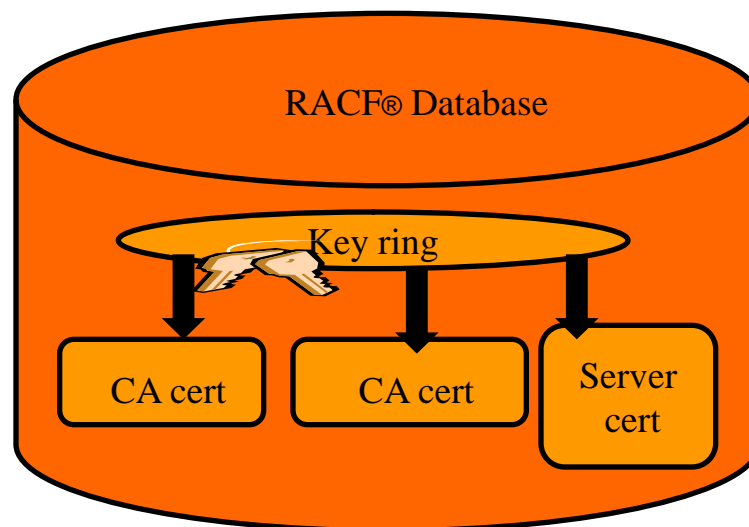
Certificate Stores on z/OS

- gskkyman manages certificates stored in a key database file
- RACDCERT manages certificates stored in a RACF key ring.

GSKKYMAN



RACDCERT



Certificate Store Protection

- gskkyman key database files
 - Protected by the file system's permission bits and password
 - Upon creation, permission bits are 700 giving the issuer of gskkyman read and write to the file only.
 - Applications using these files need at least read to the file

- RACF Key Rings
 - RACF key rings are protected by resource profiles.
 - Users rings need read access to IRR.DIGTCERT.LISTRING or <ring owner>.<ring name>.LST to be able to read the contents of their key ring.

Certificate Utilities

- **gskkyman** is a Unix based utility shipped as part of the System SSL product in the z/OS Cryptographic Services Element
- **RACDCERT** is a TSO command shipped as part of RACF
- Provide basic certificate functions
 - ▶ Create/delete certificate store (HFS key database file / SAF key ring)
 - ▶ Create certificate requests (to be signed by trusted Certificate Authority)
 - ▶ Import/Export certificates (with and without private keys)
 - ▶ Create self-signed certificates
- Do not have all the functions of a real Certificate Authority

Certificate Authority on z/OS

- **PKI Services** provides full certificate life cycle management
 - ▶ Request, create, renew, revoke certificate
 - ▶ Provide certificate status through Certificate Revocation List(CRL) and 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

Defining a Certificate

- **How will the certificate be used?**
- **What certificate store is to be used?**
- **Who will be the certificate authority?**
- **What is the identities' subject name?**
- **What is the size of the public/private keys?**
- **Whether additional identity information is to be added to the certificate?**
- **What label or nickname will the certificate be known by?**

Defining a Certificate Request to be signed by a CA

- A **certificate signing request** (also **CSR**) is a message sent from the certificate requestor to a certificate authority to obtain a signed digital certificate
- Contains identifying information and public key for the requestor
- Corresponding private key is not included in the CSR, but is used to digitally sign the request to ensure the request is actually coming from the requestor
- CSR may be accompanied by other credentials or proofs of identity required by the certificate authority, and the certificate authority may contact the requestor for further information.
- If the request is successful, the certificate authority will send back an identity certificate that has been digitally signed with the private key of the certificate authority.

Steps to request a CA signed Certificate

▪Steps:

- ▶ Create a key database file or SAF key ring
- ▶ Receive CA certificate, if not already in database
- ▶ Create a new certificate request and send to CA
- ▶ Receive signed certificate
- ▶ Indicate to the application that this certificate is to be used
 - ▶ Mark it as 'default'
 - ▶ Name it with a specific required label

If you use gskkyman...

Create a key database

Database Menu

- 1 - Create new key database**
 - 2 - Open key database**
 - 3 - Change database password**
 - 4 - Change database record length**
 - 5 - Delete database**
 - 6 - Create key parameter file**
 - 7 - Display certificate file (Binary or Base64 ASN.1 DER)**
- 0 - Exit Program**



Name of key database

Enter your option number: **1**

Enter key database name (press ENTER to return to menu: **/tmp/my.kdb**)

Enter database password (press ENTER to return to menu: **password**)

Re-enter database password: **password**

Enter password expiration in days (press ENTER for no expiration): **<enter>**

Enter database record length (press ENTER to use 2500): **<enter>**

This will add a number of well-known trusted CA certificates to the key database.

Importing a signing Certificate Authority Certificate

Key Management Menu

Database: /tmp/my.kdb

- 1 - Manage keys and certificates
- 2 - Manage certificates
- 3 - Manage certificate requests
- 4 - Create new certificate request
- 5 - Receive requested certificate or a renewal certificate
- 6 - Create a self-signed certificate
- 7 - Import a certificate
- 8 - Import a certificate and a private key
- 9 - Show the default key
- 10 - Store database password
- 11 - Show database record length

- 0 - Exit program

Enter option number (press ENTER to return to previous menu): **7**

Importing a signing Certificate Authority Certificate Continued

File contains the CA
certificate

Enter import file name (press ENTER to return to menu): **cacert.b64**

Enter label (press ENTER to return to menu): **CA Certificate**

Certificate imported.

Creating a new certificate request

Key Management Menu

Database: /tmp/my.kdb

- 1 - Manage keys and certificates
- 2 - Manage certificates
- 3 - Manage certificate requests
- 4 - Create new certificate request
- 5 - Receive requested certificate or a renewal certificate
- 6 - Create a self-signed certificate
- 7 - Import a certificate
- 8 - Import a certificate and a private key
- 9 - Show the default key
- 10 - Store database password
- 11 - Show database record length

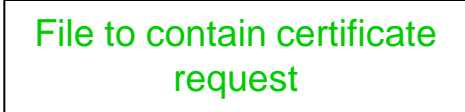
- 0 - Exit program

Enter option number (press ENTER to return to previous menu): 4

Fill in the information about the requestor

Certificate Type

- 1 - Certificate with 1024-bit RSA key
- 2 - Certificate with 2048-bit RSA key
- 3 - Certificate with 4096-bit RSA key
- 4 - Certificate with 1024-bit DSA key



File to contain certificate request

Enter certificate type (press ENTER to return to menu): **1**

Enter request file name (press ENTER to return to menu): **certreq.arm**

Enter label (press ENTER to return to menu): **Server Certificate**

Enter subject name for certificate

Common name (required): **Server Certificate**

Organizational unit (optional): **Production**

Organization (required): **IBM**

City/Locality (optional): **Endicott**

State/Province (optional): **New York**

Country/Region (2 characters - required): **US**

Enter 1 to specify subject alternate names or 0 to continue: **1**

Content of the certificate request

Contents of certreq.arm file:

```
-----BEGIN NEW CERTIFICATE REQUEST-----  
MIIB3jCCAUcCAQAwezELMAkGA1UEBhMCVVMxETAPBgNVBAgTCE5ldyBZb3JrMREw  
DwYDVQQHEwhFbmRpyY290dDEMMAoGA1UEChMDSUJNMRMwEQYDVQQLEwpQcm9kdWN0  
aW9uMRswGQYDVQQDExJTZXJ2ZXIgcQ2VydGhmaWNhdGUwgZ8wDQYJKoZIhvcNAQEB  
BQADgY0AMIGJAoGBAMTiaO7czZdi8IU+eCL23xtrqhXBqnksHBwdW8zeCjnqxq1l  
ump9GY4Jw9Wyqp9a2J85bWJD06TaHhFALru5pgOl+jMOQTbB+wZoSolbIrwoWl6l  
pLx1cqJOn53mBmv6ruP/d055jjgKTczYhOa2JdhmfpAvf+C6tUkn7qMWlRzNAGMB  
AAGgKzApBgkqhkiG9w0BCQ4xHDAaMBGGA1UdEQQRMA+CDW15Y29tcGFueS5jb20w  
DQYJKoZIhvcNAQEFBQADgYEAAxCvLl4Cq+YVdJuHGnVr28ySnPz8E1uMT/k9Y6qM  
EE+3Hiy2aD2mUREyeljehF5VNSbHwG5VcFrFVV0tuVomeJgY8bYmlE45Z4oJoyqFG  
HdQVUQO5E+W3UvKYv698KQTpl668BV51F3x1BwNx6K1PLl40i0fq8gFMfB8nP0KM  
LOs=  
-----END NEW CERTIFICATE REQUEST-----
```

Receiving a signed certificate request

Key Management Menu

Database: /tmp/my.kdb

- 1 - Manage keys and certificates
- 2 - Manage certificates
- 3 - Manage certificate requests
- 4 - Create new certificate request
- 5 - Receive requested certificate or a renewal certificate
- 6 - Create a self-signed certificate
- 7 - Import a certificate
- 8 - Import a certificate and a private key
- 9 - Show the default key
- 10 - Store database password
- 11 - Show database record length

- 0 - Exit program

File contains cert
returned from CA

Enter option number (press ENTER to return to previous menu): **5**

Enter certificate file name (press ENTER to return to menu): **svrcert.arm**

Marking a certificate as the default

Key and Certificate Menu

Label: Server Certificate

- 1 - Show certificate information
- 2 - Show key information
- 3 - Set key as default
- 4 - Set certificate trust status
- 5 - Copy certificate and key to another database
- 6 - Export certificate to a file
- 7 - Export certificate and key to a file
- 8 - Delete certificate and key
- 9 - Change label
- 10 - Create a signed certificate and key
- 11 - Create a certificate renewal request

- 0 - Exit program

Enter option number (press ENTER to return to previous menu): **3**

If you use RACDCERT... (ISPF Panel or Command)

RACDCERT Panel on Key Ring

```
                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 Panel on Certificate

```
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.

Create a key ring

Name of key ring

```
RACDCERT ID(FTPserver) ADDRING(MyRACFKKeyRing)
```

Importing a signing Certificate Authority Certificate

Dataset contains the CA certificate


```
RACDCERT CERTAUTH ADD('user1.cacert') TRUST  
WITHLABEL('CA Certificate')
```

```
RACDCERT ID(FTPServer) CONNECT (CERTAUTH LABEL('CA  
Certificate') RING(MyRACFKKeyRing) USAGE(CERTAUTH))
```

Creating a new certificate request

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

```
RACDCERT ID(FTPServer) GENREQ(LABEL('Server Certificate'))  
DSN('user1.certreq')
```



Dataset to contain
certificate request

Receiving a signed certificate request

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

Dataset contains cert
returned from CA

```
RACDCERT ID(FTPServer) CONNECT(ID(SUIMGTF) LABEL('Server  
Certificate') RING(MyRACFKeyRing) USAGE(PERSONAL)  
DEFAULT)
```

Listing a RACF Key Ring

RACDCERT ID(FTPServer) LISTING(MyRACFKeyRing)

Ring:

```

>MyRACFKeyRing<
Certificate Label Name      Cert Owner      USAGE      DEFAULT
-----
CA Certificate              CERTAUTH       CERTAUTH   NO
Server Certificate         ID(FTPServer)  PERSONAL   YES
  
```

Note: RACF key rings allow for a certificate's private key to be stored into ICSF's (Integrated Cryptographic Service Facility) PKDS (Public Key Dataset) for added security.

Certificate Formats

- **X.509 certificates can exist in many different forms**
 - Single certificate
 - PKCS #7 certificate package
 - Contains 1 or more certificates
 - PKCS #12 certificate package
 - A password encrypted package containing 1 or more certificates and the private key associated with the end-entity certificate.
 - **Only package type that contains a private key**
- **Can be in binary or Base64 encoded format**

Base64 encoding

- **Converting binary data to displayable text for easy cut and paste.**

-----BEGIN CERTIFICATE-----

```
MIICPTCCAaagAwIBAgIIR49S4QANLvEwDQYJKoZIhvcNAQEFBQAwNzELMAkGA1UE
BhMCMVVMxDTALBgNVBAoTBFRlc3QxGTAXBgNVBAMMEFRlc3Rfc2VsZl9zaWduZWQw
HhcNMDgwMTE3MTMwNjQxWhcNMDkwMTE2MTMwNjQxWjA3MQswCQYDVQQGEwJVUzEN
MA5GA1UEChMEVGVzdDEZMBcGA1UEAwwQVGVzdF9zZWxmX3NpZ25lZDCBnzANBgkq
hkiG9w0BAQEFAAOBjQAwgYkCgYEA9tK0v5gLaceozMfMeVd891fCjBVoR+dpzhwK
R2B/QcQYBGLfqS4YM/wGSh6YrmVyg00VxocriySbcxRuBayw3pE4/3JI2myINmLp
bFIdPCnqk/qvFK+1N+nrEnBK9yls7NmxDIuQQfFsX/o/DpoxwzxwXf+JbWDwirQR
NyLiTGMCaAwEAAaNSMFAwHQYDVR0OBBYEFawDFLjOUcRa62BVs3jVyHewuOWEMB8G
A1UdIwQYMBaAFAwDFLjOUcRa62BVs3jVyHewuOWEMA4GA1UdDwEB/wQEAwIE8DAN
BgkqhkiG9w0BAQUFAAOBgQAC5sW1f3EdE0k9zc8wKNt1sczWkQBrVy4Rdr17ERqN
D2OfkBJQuXiNwN18pF6WPWFYG80MNwhP4oJSVePnzElh4Wzi2wl/zI8rINSW7px3
w16lz+8jEI84q/N0q0toPTAtEb6fIzwjkLtctt3oF+IjunvE5QoRsXRJbbTMD/EG
jw==
```

-----END CERTIFICATE-----

Exporting Certificates through gskkyman

Key and Certificate Menu

Label: Server Certificate

- 1 - Show certificate information
- 2 - Show key information
- 3 - Set key as default
- 4 - Set certificate trust status
- 5 - Copy certificate and key to another database
- 6 - Export certificate to a file
- 7 - Export certificate and key to a file
- 8 - Delete certificate and key
- 9 - Change label
- 10 - Create a signed certificate and key
- 11 - Create a certificate renewal request

- 0 - Exit program

Enter option number (press ENTER to return to previous menu):

Exporting Certificates through gskkyman

Option 6 – Public Certificate Information

Export File Format

- 1 - Binary ASN.1 DER
- 2 - Base64 ASN.1 DER
- 3 - Binary PKCS #7
- 4 - Base64 PKCS #7

Option 7 – Public Certificate Information and Private Key

Export File Format

- 1 - Binary PKCS #12 Version 1 (Few very old applications still use V1)
- 2 - Base64 PKCS #12 Version 1
- 3 - Binary PKCS #12 Version 3
- 4 - Base64 PKCS #12 Version 3

Exporting Certificates through RACDCERT

- **RACDCERT ID(userid) EXPORT**

(LABEL('label-name'))

DSN(output-data-set-name)

FORMAT(CERTDER | CERTB64 | PKCS7DER | PKCS7B64 |
PKCS12DER | PKCS12B64)

PASSWORD('pkcs12-password')

- **Example - Export Server Certificate with its private key**

- RACDCERT ID(FTPServer) EXPORT

- LABEL('Server Certificate') DSN('USER1.SERVER.CERT')

- FORMAT(PKCS12DER) PASSWORD('passwd')

Summary

- **Digital certificates provide electronic identity and public key information to be utilized through public key protocols (ie. SSL/TLS)**
- **Utilizing trusted CAs is key to ensure validity of the digital certificate**
- **Protect the private key!!!**
- **Larger the public/private key pair size, greater security, but more computation intense**

Summary

- **Certificate source usage is application defined.**
- **When transferring certificates, use a format acceptable to the receiving side.**
- **When transferring certificates, be sensitive to binary and text modes to ensure proper transfer**

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 (SG24-7248)
- **Security Server Manuals:**
RACF Command Language Reference (SC22-7687)
RACF Security Administrator's Guide (SC28-1915)
- **Cryptographic Server Manual**
Cryptographic Services System Secure Sockets Layer Programming (SC24-5901)
- **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