#SHAREorg

SHARE
Technology · Connections · Results

Understanding Digital Certificates on z/OS

Saheem Granados, CISSP
IBM
sgranados@us.ibm.com
Thursday, March 15, 2012
10423

Visit www.SHARE-SEC.com for more information on the SHARE Security & Compliance Project

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

- CICS*
- DB2*
- IBM*
- IBM (logo)*
- OS/390*
- RACF*
- Websphere*
- z/OS*

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Identrus is a trademark of Identrus, Inc

VeriSign is a trademark of VeriSign, Inc

Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Agenda

- Cryptography Primer
  - For your eyes only: Encryption
  - Things are as they were intended: Hashing
  - “John Hancock”: Digital Signatures
- What are Digital Certificates?
- Digital Certificate details
- Trust and Digital Certificates
- Digital Certificates on z/OS
- Summary
- Appendix – z/OS commands

For your eyes only: Encryption

- Fundamental social need to keep secrets/protect artifacts
  - A person's diary of personal thoughts
  - Government secrets
  - Social security number
  - Money
- Sensitive/secretive artifacts should only be accessible by appropriate individuals, i.e., Access must be authorized!
- Typically, an authorized party must satisfy the requirements set by a protection mechanism to access secured artifacts
  - Use key to open locked diary
  - Run through the proper combination to safe
  - Provide PIN number to ATM
- Encryption = an approach to protect/keep data secret

Things are as they were intended: Hashing

- Another key social function is detecting change
  - Change in weather patterns
  - Change in one's bank accounts
  - Change in weight
- Typically, one detects change by observing an artifact two different moments in time and comparing
  - Compare weather from last year on a given date with the weather from the current year
  - Compare two different monthly statements
- Hashing = an approach to detecting change in data

“John Hancock”: Digital Signatures

- Another key social function, essential to many of our operating institutions, is using written signatures as a means for a person to bind to agreement or prove access to documents.
  - Sign a contract to purchase a large item
  - Sign a credit card receipt
  - Sign a document to prove you have read it
- This function implies three things
  - Person signing proves their identity
  - Once signed the agreement or document cannot change (unless agreed upon by you)
  - Person signing cannot later deny they signed to the agreement or document
- Digital Signatures = an approach to have entities (people, computers, organizations, etc..) bind to agreements or prove access to data.

Encryption

- Simplest form: Math function (algorithm) that takes input data and transposes into different output data, such that output data cannot easily be transposed back to the original data.
  - A → EncryptionFunction (input+7) -> G
    G → DecryptionFunction (input-7) -> A
    - Access to the original data, i.e., decryption is only possible if algorithm (input+7) is known.
- Realistic form: Math function (algorithm) that takes **input data and key** and transposes into different output data, such that output data cannot easily be transposed back to the original data.
  - A, key → EncryptionFunction (AES) -> G
    G, key → DecryptionFunction (AES) -> A
    - Access to the original data, i.e., decryption is only possible if algorithm (AES) and KEY are known

Encryption

- Common approach to encryption now is: algorithm is known and key material is protected
  - Seems to help security since algorithm can be scrutinized and weaknesses can be evaluated before deployment.
- Two different types of encryption: Symmetric and Asymmetric
- Symmetric: Same key is used for both encryption and decryption:
  - A, key → EncryptionFunction (AES) -> G
    G, key → DecryptionFunction (AES) -> A
    - *key must be the same for both*
- Asymmetric: Key pair used, where one half of pair is used for encryption and the other half is used for decryption
  - A, key → EncryptionFunction (RSA) -> G
    G, key' → DecryptionFunction (RSA) -> A
    - *key and key' constitute a key pair, i.e. mathematically related such that if key is used for one part of the process, key' is the only key that can do the reverse.*

Hashing

- Simplest form: Math function (algorithm) that takes input data and generates a block of data that uniquely identifies the input while making it very difficult to determine the input given the output:
  - ABCDFBBDE… → HashFunction (SHA-1) -> 1288…
- Hash of a block of data allows one to prove that data has not changed
  - Determine hash when data originally created/sent
  - Determine hash at a later time. If they don't match, data has changed.

- Hashing + Encryption basically equals Digital Signature

More on Asymmetric Encryption

- Remember key pairs:
  - Input, key-> EncryptAlgorithm (RSA)-> EncryptedOutput
  - EncryptedOutput, key' -> DecryptAlgorithm(RSA) -> Input
- Mathematically speaking, which of the pair is used for the encrypt, decrypt doesn't really matter.
- By convention:
  - One half the key pair is called **Private Key**
    - One keeps their private key protected/secret
  - The other half is called the **Public Key**
    - One makes their public key widely available (hint: Digital Certificates)
- By convention: (Encrypting for the sake of exchange)
  - One encrypts using the recipient's **Public Key**
    - Help ensures only private key owner can decrypt
  - The recipient decrypts using it's **Private Key**

Digital Signatures

- Simplest form: Math function (algorithm) that takes input data and generates a Hash and then encrypts the Hash using its private key:
  - ABCDFBBDE… → HashFunction (SHA-1) -> 1288…, **Private Key** -> EncryptionAlgorithm (RSA) -> 573453….
- Those interested in **verifying** digital signature would:
  - ABCDFBBDE… → HashFunction (SHA-1) -> 1288…
    573453…,**Public Key** -> DecryptionAlgorithm (RSA) -> 1288…
    Compare results
    - *Results match = Data has not changed since signature AND Signer has bound/produced original data*

Questions…

- Encryption
- Hash
- Digital Signatures

Digital Certificates

- Recall “John Hancock”:
  - Person signing proves their identity
  - The agreement or document cannot change (unless agreed upon by you)
  - Person signing cannot later deny they signed the agreement or document
- Digital Signatures address most of the above:
  - The agreement or document cannot change (unless agreed upon by you)
    - Hashing step of digital signature processing enforces this
  - Person signing cannot later deny they signed the agreement or document
    - Private key is assumed to be protected and only accessible by signer
- The one requirement not completely addressed by digital signatures is:
  - Person signing proves their identity
- Digital Certificates provides a framework to better address this requirement

What are Digital Certificates?

- Simplest form: A digital certificate is a digital document that contains among other information, an entity's public key and is used as a means to prove the entity's identity.
  - Analogous to a person's driver license or passport
  - A few types of digital certificate formats exist, most notably **X.509**
- X.509 certs contain important identification information for its owner
  - Entity's distinguished name – unique identifier of owner
  - Entity's Public Key
- X.509 certs contain important information regarding the contents of the digital certificate
  - Dates of Validity
  - How certificate can be used, e.g.,
    - Encryption only
    - Digital signatures only
    - Certificate Authority (more on this later)
  - Issuer's Identification
- X.509 allows for Certificate Revocation (e.g., private key gets compromised)

X.509 Certificate Example

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

Why Digital Certificates?

- Since Digital Certificates contain public key material, they are available to anyone
- Essential for
  - Verifying digital signatures
  - Protocols to securely exchange data
- Common secure exchange data protocols rely on digital certs
  - Sender generates a random symmetric key (since symmetric encryption is faster)
  - Sender retrieves a recipient's Public Key from a digital cert to encrypt the random symmetric key
  - Sender uses random symmetric key to encrypt payload
  - Sender sends encrypted payload and encrypted symmetric key to recipient
  - Recipient decrypts symmetric key using its private key
  - Recipient decrypts payload with symmetric key
- Often backup of data leverages digital certs to ensure data encrypted
- Required for SSL/TLS communication
  - Major web browsers handle digital certificate processing for users
  - Warns if problems occur that could result in diminished security
    - Certificate's signature doesn't validate
    - Certificate has expired or been revoked
    - Certificate was not issued by a **trusted party**

Trust & Proof of Identity

- Recall
  - Proof of Identity is critical for effective use of signatures
  - Digital Cert analogous to a person's identification e.g., a driver license or passport
- Often a person produces identification when a signature is required
  - If I require a signature, do I **TRUST** the issuer verified the person's identify?
  - Same applies to Digital Certs
- X.509 Digital Certificates include
  - Issuer's identification
  - Issuer's Digital Signature of the certificate
    - Issuer will not sign certificate unless it has sufficiently verified the cert owner's identify
- Issuer = Certificate Authority (CA)
- Well known certificate Authorities exist, e.g., Verisign
  - Web browsers, RACF, etc.. come pre-loaded with many well known CA certs
- Typically a digital certificate is **TRUSTED** if
  1. The CA is **trusted,** i.e., the CA's certificate has been validated/accepted
  2. The CA's signature of the certificate validates

CA & Chaining

- CAs issue and sign digital certificates
- Recall X.509 Digital Certificates can define the uses of the certificate/public key
  - Encrypt only
  - Sign Only
  - **Certificate Authority**
- Not always practical to have a well know CA issue all the certs need for an organization
  - X.509 allows for CA chain to be associated with a digital cert
- A CA chain establishes a set of digital certificates that must be validated when validating a digital cert
- A root CA (can be a well known third party, e.g., Verisign) issues 1 or more intermediate CA digital certs to an org
  - Root CA is self signed. Great CARE must be taken to verify the root CA cert.
  - Main browsers, RACF, etc… come pre-loaded with most common root CA certs
- The intermediate CA digital certs can be used to sign other intermediate CA certs (if a longer CA chain is needed) or a user's or system's digital cert
- When a CA chain is encountered, the certificate is validated by recursively validating the CA signatures for all of the digital certificates within the CA chain

CA & Chaining

Is the root CA in my key ring ?

Finish

Self signed:
Issuer=Subject

Root CA
Issuer – CN=Root CA,OU=Signers,O=IBM,C=US
Subject -CN=Root CA,OU=Signers,O=IBM,C=US
…
Signature

Intermediate CA
Issuer - CN=Root CA,OU=Signers,O=IBM,C=US
Subject – CN=Intermediate CA,OU=Signers,O=IBM,C=US
…
Signature

End Entity
Issuer – CN=Intermediate CA,OU=Signers,O=IBM,C=US
Subject -CN=Server Certificate,OU=z/OS,O=IBM,C=US
…
Signature

Start

When defining a digital certificate…

- How will the certificate be used?
- What certificate store is to be used?
- Who will be the certificate authority?
- What is the identity's 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?

Questions…

- Digital Certificates
  - What?
  - Why?
- Trust
- Certificate Authority and Chaining

Digital Certificates on z/OS

- Digital Certificates and Signatures are CRITICAL to day-day IT activity
  - SSL/TLS
  - Encrypting data for backup
  - Securing Data for exchange with partners
- Key and Certificate Management can be non-trivial
- Certificate Authority related work can be non-trivial (consider Intermediate CAs)
- Services available to facilitate digital cert usage on z/OS
  - IBM ICSF
  - IBM RACF
  - IBM System SSL Unix commands - gskkyman
  - IBM PKI Services

ICSF

- IBM ICSF – Cryptographic services facility on z/OS
  - Provides Key repositories –
    - CKDS – Symmetric keys
    - TKDS – PKCS#11 related objects
    - PKDS - Asymmetric key pairs
  - Provides Cryptographic algorithm implementations, including hardware accelerated crypto
  - PKDS does not store X.509 certs.
  - TKDS can have X.509 cert objects.

RACF

- Manage X.509 certificates stored in Key Rings
- Key rings/certs are protected via RACF profiles
- Stores Related key material
  - Key material can be stored in ICSF as an alternative
- RACDCERT – RACF command to manage Key rings
  - TSO command
  - ISPF panels available
- Learn more:

RACF Command Language Reference (SC22-7687)

```
                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.
```

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')

gskkyman

- gskkyman – Unix utility shipped with System SSL
  - Manage X.509 certs stored in USS file system key database files
    - Files protected by file system File **permission bits** and **password**
  - Learn more:

    Cryptographic Services System Secure Sockets Layer Programming (SC24-5901)

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

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

PKI

- IBM PKI Services – Handles complete cert life cycle management
  - Manage X.509 certificates
  - Perform Certificate Authority related functions
- Closely tied to RACF
  - The CA cert must be **installed** in RACF's key ring
  - **Authority checking** goes through RACF's callable service
  - Most of the **auditing** work done through RACF
- CA cert private key can be stored in **ICSF**
- Generation and administration of certificates via customizable **web pages**
- Keys can be generated by requestor, or generated by PKI (**Key escrow**)

Questions…

- ICSF
- RACF
- gskkyman
- PKI Services

Certificate Related Tips

- TRUST is key to leveraging Certificate Related Technology
  - Private key MUST be protected!!!
  - Certificate Authorities must be reliable
- Larger public/private key pairs increase security, however also increase computational cost
- Business Practices needed for checking a certificate's revocation/validity status

Referencces

- **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

Summary

- Cryptography Primer
  - Encryption
  - Hashing
  - Digital Signatures
- Defined Digital Certificates
- Why Digital Certificates
- Trust, Certificate Authorities, and Chaining
- Digital Certificates on z/OS
  - ICSF
  - RACF
  - SSL's gskkyman
  - PKI Services

Appendix

- Performing Common Certificate Related Tasks on z/OS

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-----
MIIB3jCCAUcCAQAwczELMAkGA1UEBhMCVVMxETAPBgNVBAgTCE5ldyBZb3JrMREw
DwYDVQQHEwhFbmRpY290dDEMMAoGA1UEChMDSUJNMRMwEQYDVQQLEwpQcm9kdWN0
aW9uMRswGQYDVQQDExJTZXJ2ZXIgQ2VydGlmaWNhdGUwgZ8wDQYJKoZIhvcNAQEB
BQADgY0AMIGJAoGBAMTiaO7czZdi8IU+eCL23xtrqhXBqnksHBwdW8zeCjnqxq1l
ump9GY4Jw9Wyqp9a2J85bWJD06TaHhFALru5pgOl+jMOQTbB+wZoSOlbIrwoWl6l
pLx1cqJOn53mBmv6ruP/d055jjgKTczYhOa2JdhmfpAvf+C6tUkn7qMWlRzNAgMB
AAGgKzApBgkqhkiG9w0BCQ4xHDAaMBgGA1UdEQQRMA+CDW15Y29tcGFueS5jb20w
DQYJKoZIhvcNAQEFBQADgYEAAxCvLl4Cq+YVdJuHGnVr28ySnPz8E1uMT/k9Y6qM
EE+3Hiy2aD2mUREyeljehF5VNSbHwG5VCrFVVOtuVomeJgY8bYmlE45Z4oJoyqFG
HdQVUQO5E+W3UvKYv698KQTpl668BV51F3xlBwNx6K1PLl40i0fq8gFMfB8nP0KM
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

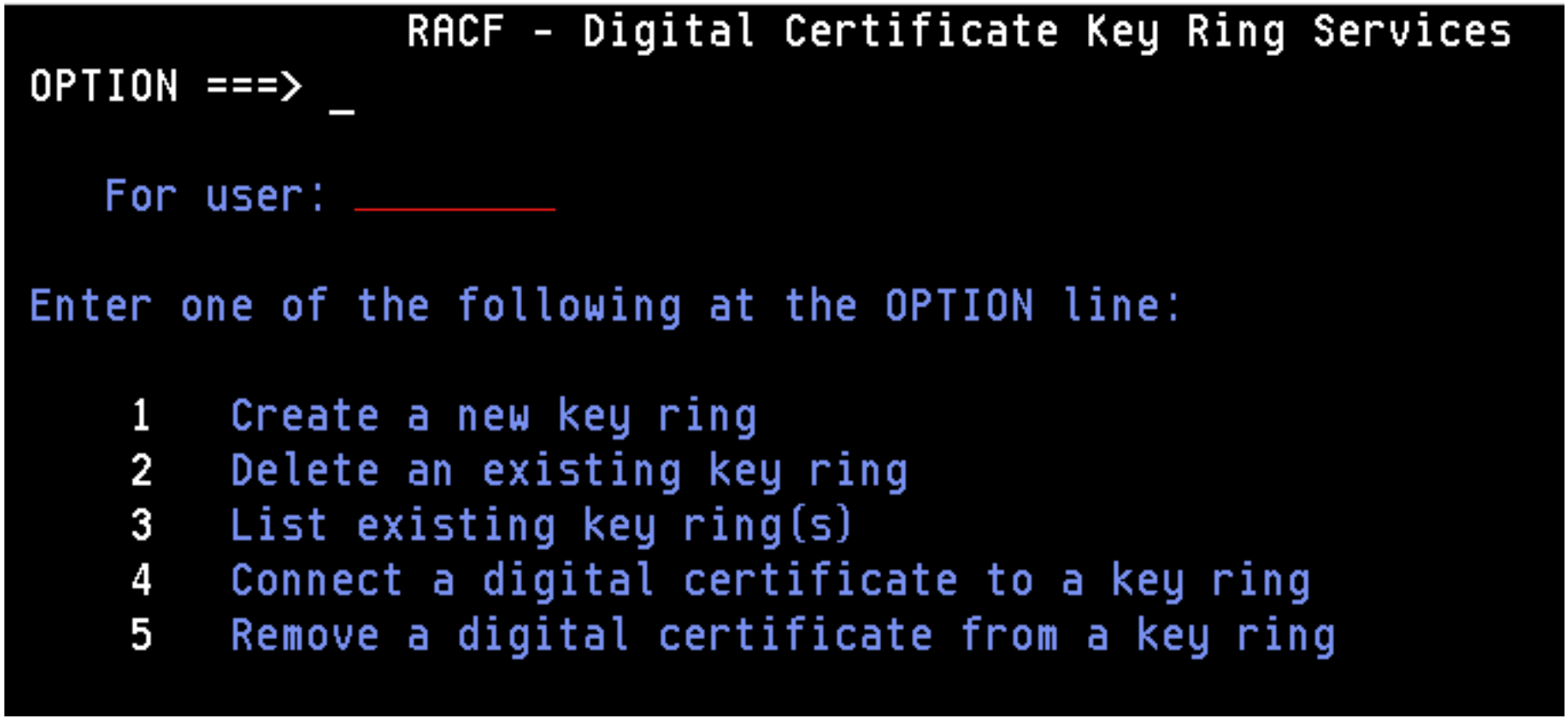
1 - Show certificate information
2 - Show key information
3 - Set key as default
4 - Set certificate trust status
5 - Copy certificate and key to another database
6 - Export certificate to a file
7 - Export certificate and key to a file
8 - Delete certificate and key
9 - Change label
10 - Create a signed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

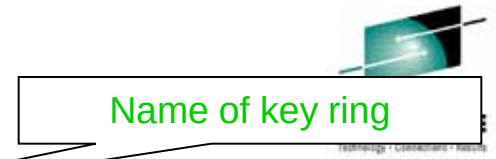
RACDCERT Panel on Key Ring



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



RACDCERT ID(FTPserver) ADDRING(MyRACFKeyRing)

Importing a signing Certificate Authority Certificate

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

RACDCERT ID(FTPServer) CONNECT (CERTAUTH LABEL('CA Certificate') RING(MyRACFKeyRing) 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.

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’)