

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

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

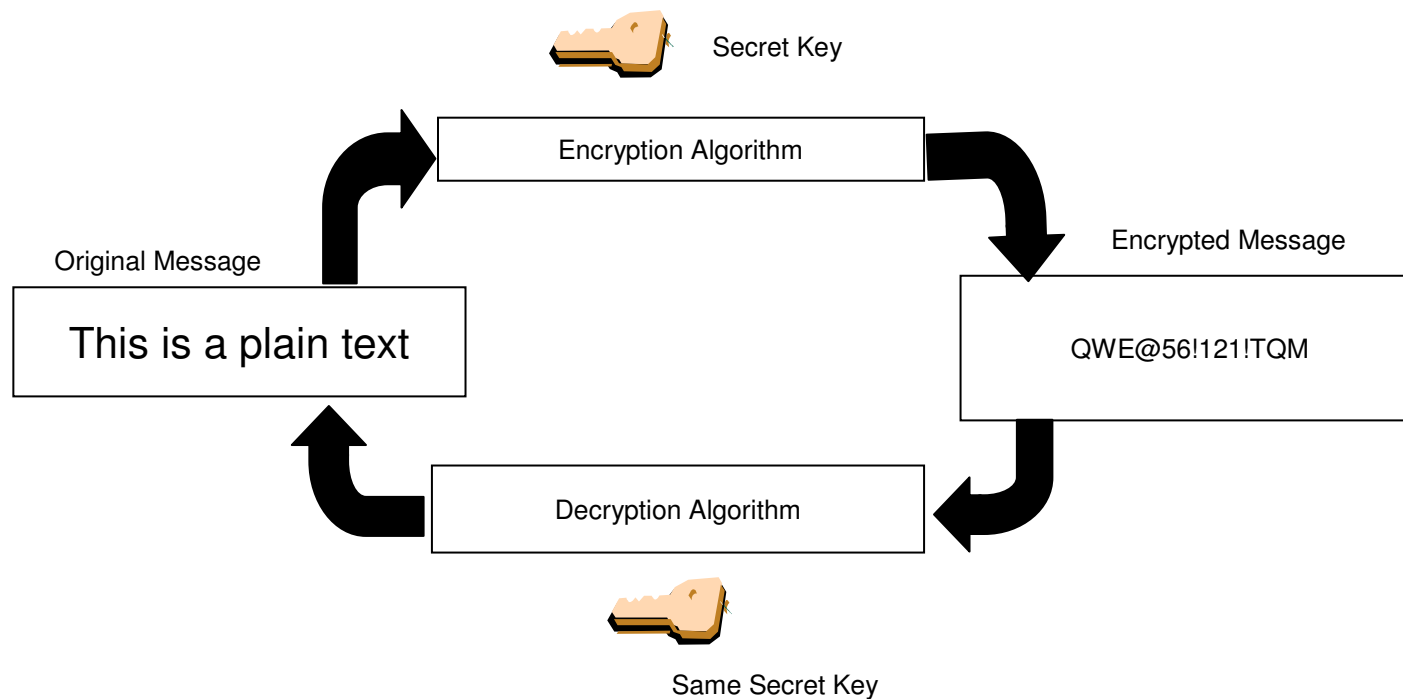
Same key used for both encryption and decryption

Provide data confidentiality

Fast, used for bulk encryption/decryption

Securely sharing and exchanging the key between both parties is a major issue

Common algorithms: DES, Triple DES, AES



Asymmetric Encryption

2 different keys - Public/private key pairs

A public key and a related private key are numerically associated with each other.

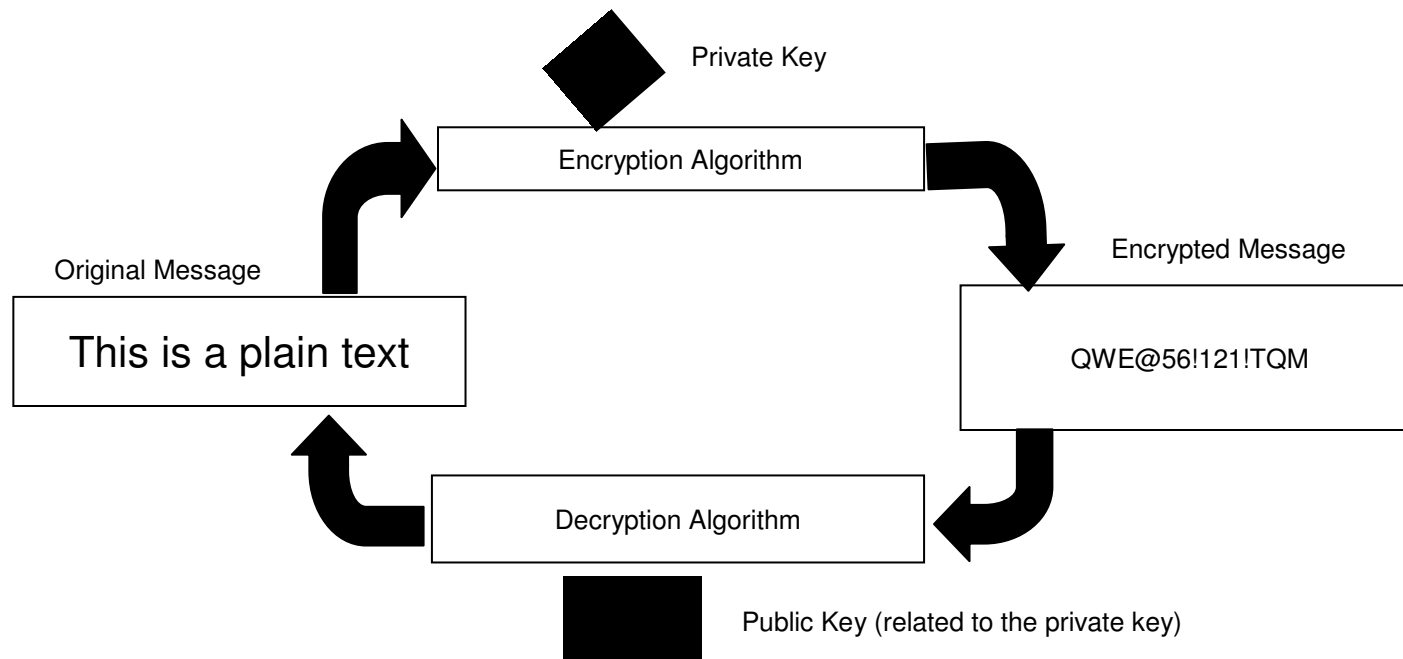
Provide data confidentiality, integrity and non repudiation

Data encrypted/signed using one of the keys may only be decrypted/verified using the other key.

Very expensive computationally

Public key is freely distributed to others, private key is securely kept by the owner

Common algorithms: RSA, DSA, ECC



Message Digest (Hash)

A fixed-length value generated from variable-length data

Unique:

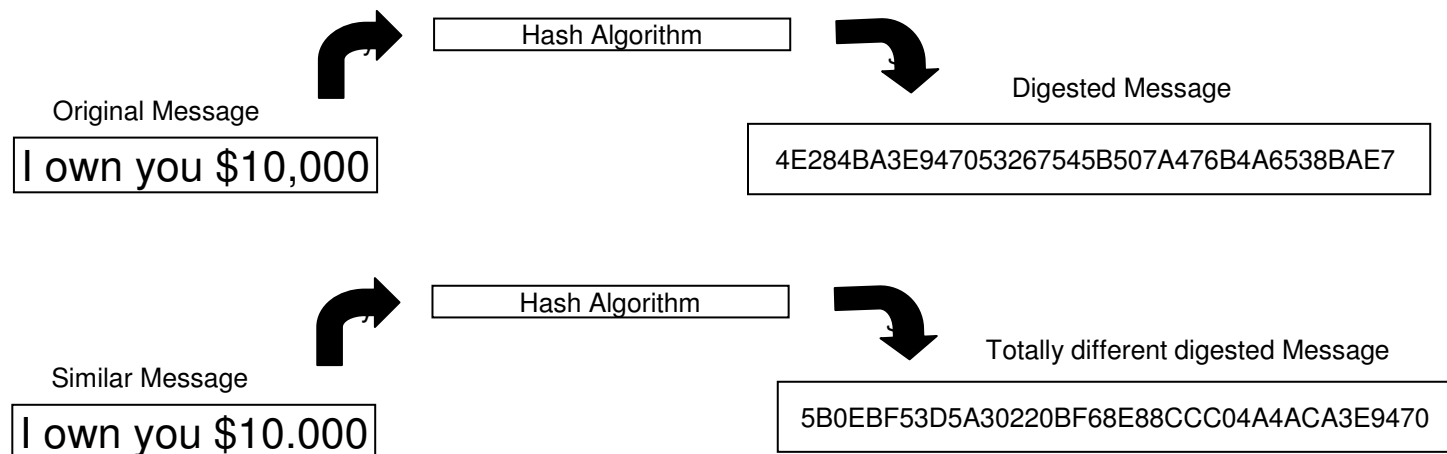
- the same input data always generates the same digest value
- tiny change in data causes wide variation in digest value
- Theoretically impossible to find two different data values that result in the same digest value

One-way: can't reverse a digest value back into the original data

NOT based on a key

Play a part in data integrity and origin authentication

Common algorithms: SHA1, SHA256

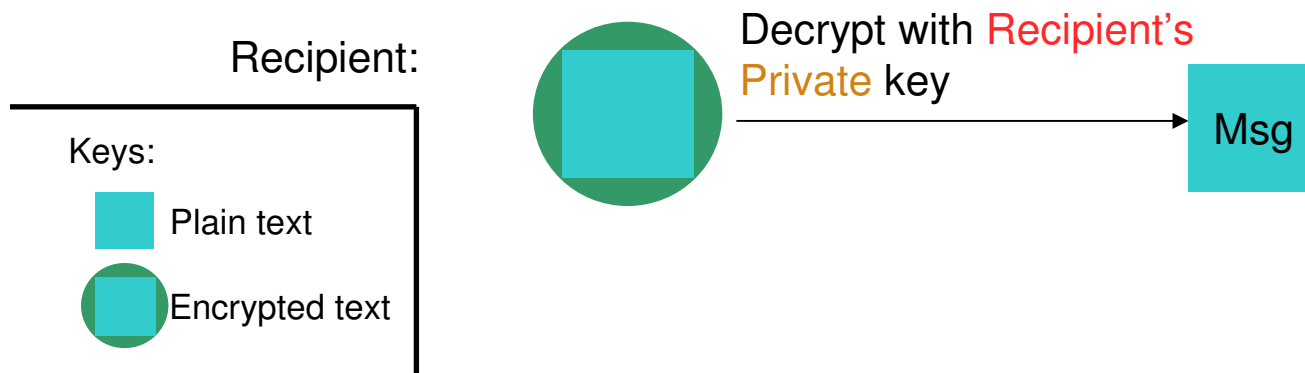


Encryption (for confidentiality)

Encrypting a message:



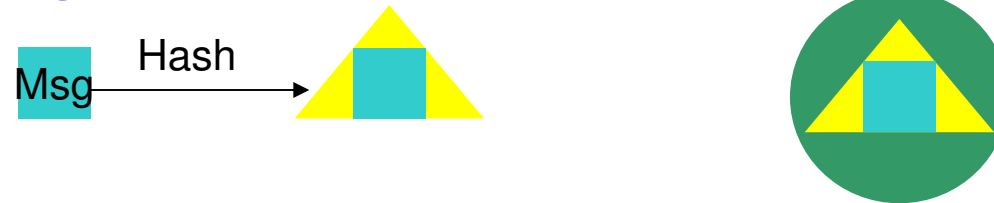
Decrypting a message:



Signing (for integrity and non repudiation)

Signing a message:

Sender:



What is a Digital Certificate (1 of 2)

Generally digital certificates provide identity to a person or a server

- Person - like an ID card
- Server – like a business license

To establish an identity or credential to be used in electronic transactions

It binds the public key to the identity to be used by applications that are based on public key protocols. (e.g. SSL/TLS)

Issued by a trusted third party called Certificate Authority (CA) that can ensure validity

What is a Digital Certificate(2 of 2)

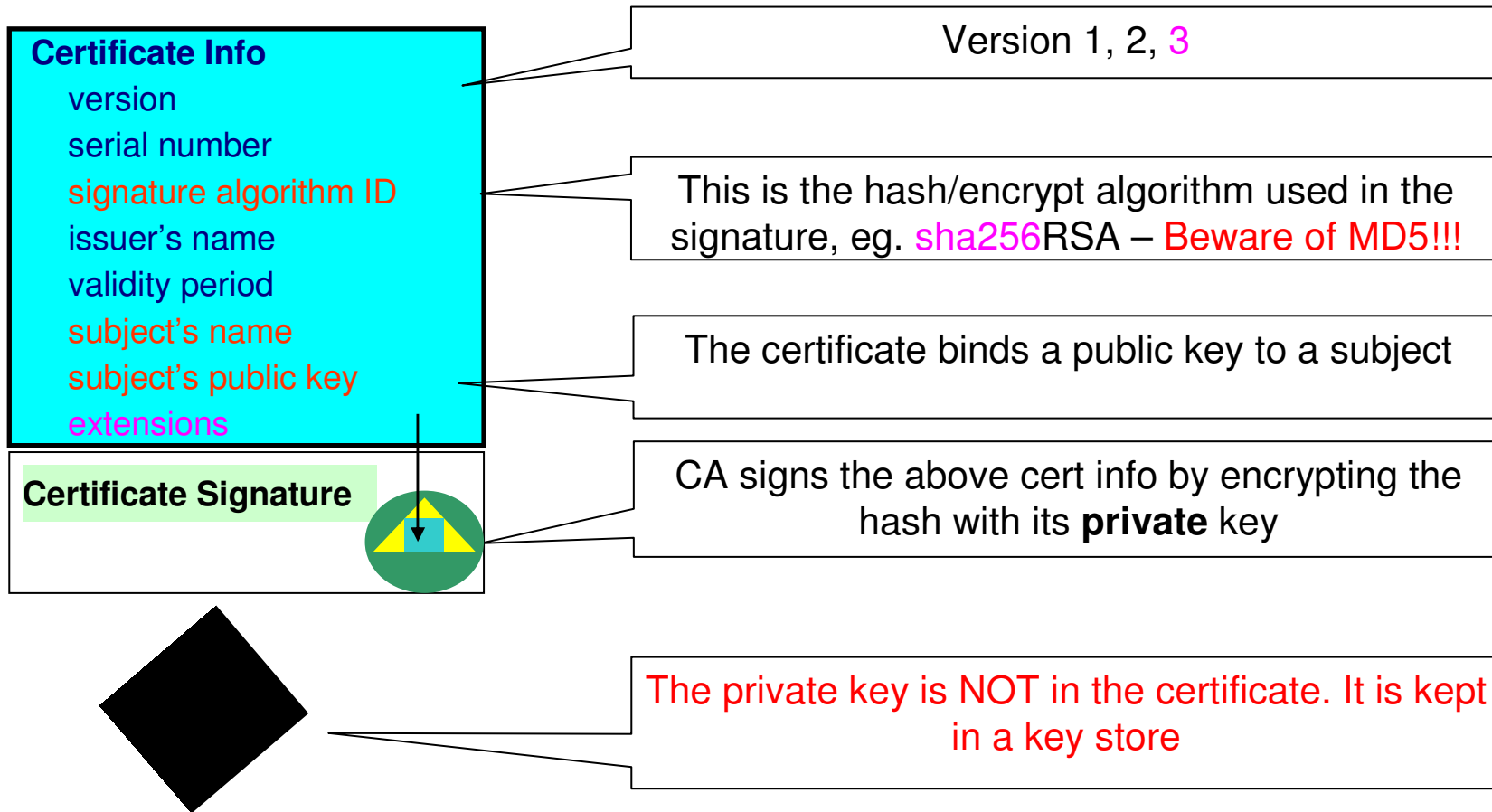
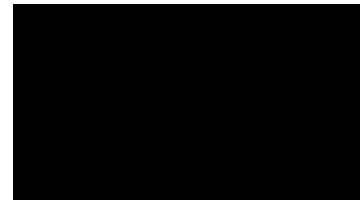
Packaging of the information is commonly known as the x.509 digital certificate. X.509 defines the format and contents of a digital certificate.

– IETF RFC 5280

Digital certificates been in existence for over 20 years

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(1 of 2)

- 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(2 of 2)

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

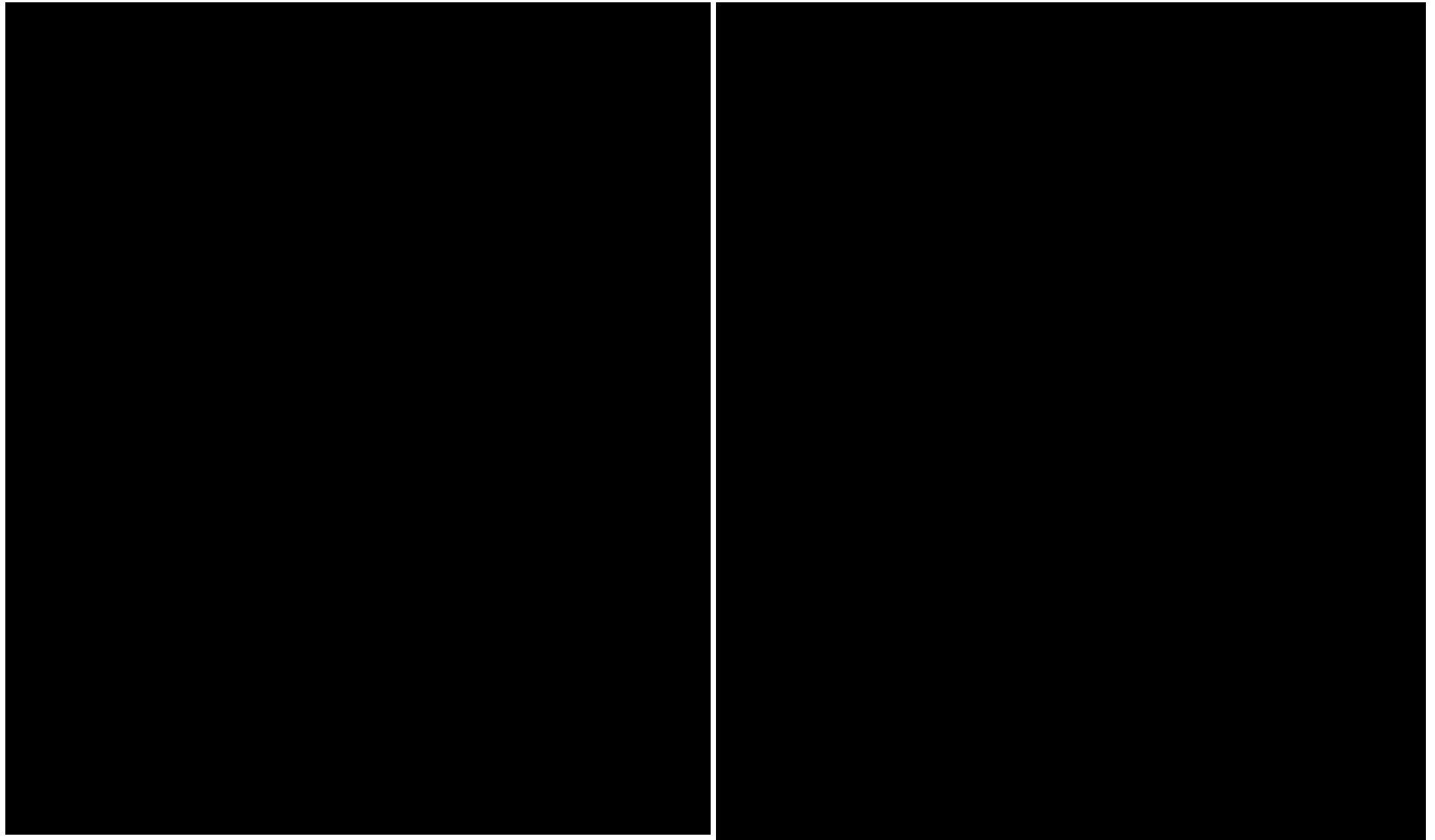
Subject Alternate Name – additional identity information

- 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



Relationship between Certificate and Certificate Store

Certificate must be placed in a certificate store before it can be used by an application to perform identification or validation

The application needs to retrieve the certificate and/or its corresponding private key from the store

On z/OS, many components like Communication Server, HTTP Server call System SSL APIs to access the store

Certificate store = key ring = key file



Types of digital certificates – who issues it

Self signed

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

Signed Certificate

- 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.

Types of digital certificates – what is the usage

Secure Socket Layer (SSL) certificate

- Install on a server that needs to be authenticated, to ensure secure transactions between server and client

Code Signing certificate

- Sign software to assure to the user that it comes from the publisher it claims

Personal certificate

- Identify an individual, enable secure email – to prove that the email really comes from the sender and /or encrypt the email so that only the receiver can read it

More (name it whatever you want)...

- wireless certificate, smart card certificate...

Certificate Authority (CA) certificate

- Used to sign other certificates
- Root CA: the top
- Intermediate CA: signed by root CA or other intermediate CA

Types of digital certificates – what is the usage

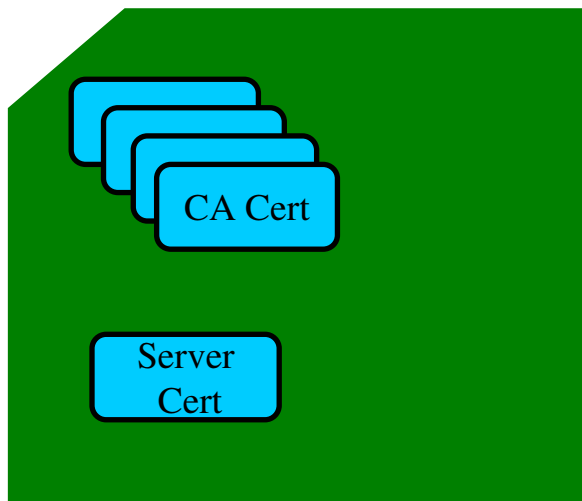
Site certificate (in RACF)

- The usage assigned to a certificate when it is connected to a RACF key ring indicates its intended purpose
- There may be a few certificate validation applications which treat a certificate that is connected to a key ring with usage site as a valid certificate authority certificate to bypass the normal certificate verification tests during SSL handshake, for example, an expired certificate can be considered trusted
- Having a SITE certificate in RACF does not benefit you if the validation application does not make use of it

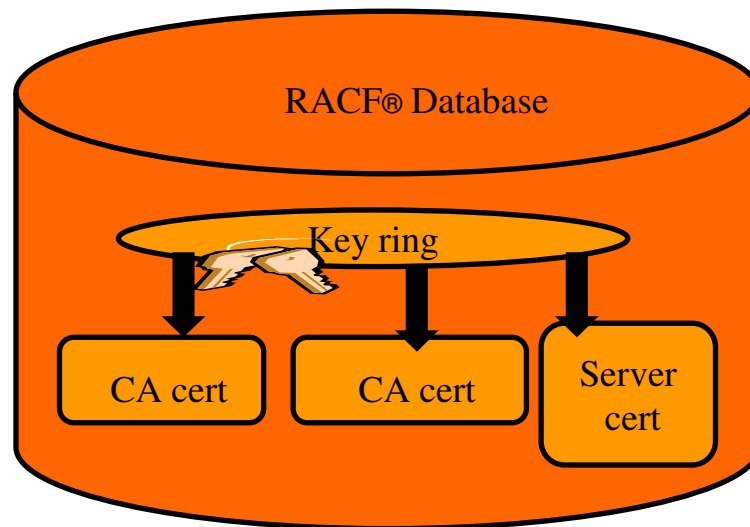
Certificate Stores on z/OS

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

IRR.DIGTCERT.LISTRING – Global control

<ring owner>.<ring name>.LST – Granular control

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
- ▶ Automatic notification or renewal of expiring certificates

Defining a Certificate

How will the certificate be used?

Who will be the certificate authority?

What certificate store is to be used?

What is the size of the public/private keys?

What subject name to use?

Need additional identity information and extensions?

Validity period of the certificate?

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.

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 (1 of 2)

Certificate Key Algorithm

- 1 - Certificate with an RSA key
- 2 - Certificate with a DSA key
- 3 - Certificate with an ECC key

Select certificate key algorithm (press ENTER to return to menu): **1**

RSA Key Size

- 1 - 1024-bit key
- 2 - 2048-bit key
- 3 - 4096-bit key

Select RSA key size (press ENTER to return to menu): **2**

Signature Digest Type

- 1 - SHA-1
- 2 - SHA-224
- 3 - SHA-256
- 4 - SHA-384
- 5 - SHA-512

Select digest type (press ENTER to return to menu): **2**

Fill in the information about the requestor(2 of 2)

File to contain certificate request

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**

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

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

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



File contains cert
returned from CA

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(MyRACFKeyRing)
```

Adding Certificate Authority(CA) Certificate to a key ring

Dataset contains the CA
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

Adding Certificate signed by CA to a key ring

```
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
BhMCMVVMxDTALBgNVBAoTBFRlc3QxGTAXBgNVBAMMEFRlc3Rfc2VsZ19zaWduZWQw
HhcNMDgwMTE3MTMwNjQxWhcNMDkwMTE2MTMwNjQxWjA3MQswCQYDVQQGEwJVUzEN
MA5GA1UEChMEVGVzZdDEZMBcGA1UEAwwQVGVzZdF9zZWxmX3NpZ25lZDCBnzANBgkq
hkiG9w0BAQEFAAOBjQAwGyKCGYEA9tK0v5gLaceozMfMeVd891fCjBVoR+dpzhwK
R2B/QcQYBGLfqS4YM/wGSh6YrmVyg00VxocriySbcxRuBayw3pE4/3JI2myINmLp
bFIdPCnqk/qvFK+1N+nrEnBK9yls7NmxDIuQQfFsX/o/DpoxwzXf+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(1 of 2)

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')

Exporting Certificates through RACDCERT(2 of 2)

Precaution needed for CERTAUTH certificate when you plan to preserve the certificate and the private key by exporting them in a pkcs12 package

- If the original CERTAUTH certificate got deleted and you re-add this package, the field that used for recording serial numbers that it has issued is not reserved
- For example, if this CA certificate has issued 100 certificates, the next certificate to be issued should have serial number 101; but after re-adding it, the certificate to be issued will have serial number 1, which is already used – all the certificates issued by the same CA should have a unique serial number!

Before deleting CERTAUTH certificate, find out the last certificate's serial number it issued

After re-adding, use r_datalib to bump up the serial number field to the appropriate number

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

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](#)

Questions ?

More on Digital
Certificates with
z/OS PKI Services

