Digital Certificates Demystified

Alyson Comer
IBM Corporation
System SSL Development
Endicott, NY
Email: comera@us.ibm.com

February 7th, 2013
Session 12534
Trademarks

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

IBM®, MVS, RACF®, z/OS®

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
UNIX is a registered trademark of The Open Group in the United States and other countries.

*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.

Complete your sessions evaluation online at SHARE.org/SanFranciscoEval
Agenda

- Cryptography
- What are Digital Certificates
- Certificate Types and Contents
- Certificate Formats
- Certificate Validation
- Certificates and SSL
- Certificate Life Cycle
- RACDCERT Command
- Using RACDCERT for SSL certificates
- z/OS PKI Services
Symmetric Encryption

- Provide data confidentiality
- **Same key** used for both encryption and decryption
- **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

This is a plain text

Original Message

Encrypted Message

QWE@56!121ITQM

Encryption Algorithm

Decryption Algorithm

Same Secret Key

Secret Key
Asymmetric Encryption

- **Public / private key pairs** - 2 different keys
  - 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.
- **Slow**, Very expensive computationally
- **Public key is freely distributed** to others, private key is securely kept by the owner
- **Common algorithms**: RSA, DSA, ECC

Diagram:
- Original Message
  - This is a plain text
- Encryption Algorithm
  - QWE@56!121!TQM
- Decryption Algorithm
  - Public Key (related to the private key)
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
- **No keys involved** – Result determined only by the algorithm
- Play a part in data integrity and origin authentication
- **Common algorithms:** SHA1, SHA256, SHA384

### Examples

- **Original Message**
  - *I owe you $10,000*
  - *Digest Algorithm: 4E284BA3E947053267545B507A476B4A6538BAE7*

- **Similar Message**
  - *I owe you $10,000*
  - *Digest Algorithm: 5B0EBF53D5A30220BF68E88CCC04A4ACA3E9470*

- **Totally different digested Message**
  - *I owe you $10.000*
  - *Digest Algorithm: 4E284BA3E947053267545B507A476B4A6538BAE7*
Encryption (for confidentiality)

Encrypting a message:

Sender: Encrypt withRecipient’s Public key

Decrypting a message:

Recipient: Decrypt withRecipient’s Private key

Keys:
- Plain text
- Encrypted text
Signing (for integrity and non repudiation)

Signing a message:

Sender:

1. Hash the message (Msg).
2. Encrypt the hash with the sender's private key.
3. Sign the message (Msg) with the sender's private key to obtain the signature.

Verifying a message:

Recipient:

1. Decrypt the signature with the sender's public key to recover the hash.
2. Hash the received message (Msg).
3. Do they match? If yes, the message is unaltered. Assuming the hashing algorithm is strong.
What is a Digital Certificate?

• A Digital Certificate is a digital document issued by a trusted third party which binds an end entity to a public key.

• Digital document:
  • Contents are organized according to ASN1 rules for X.509 certificates
  • Encoded in binary or base64 format

• Trusted third party aka Certificate Authority (CA):
  • The consumer of the digital certificate trusts that the CA has validated that the end entity is who they say they are before issuing and signing the certificate.

• Binds the end entity to a public key:
  • End entity - Any person or device that needs an electronic identity. Encoded in the certificate as the Subjects Distinguished Name (SDN). Can prove possession of the corresponding private key.
  • Public key - The shared half of the public / private key pair for asymmetric cryptography
  • Digitally signed by the CA
What is a Digital Certificate?

- Best way to think of it is as an **ID card**, like driver licenses or passport
- To **establish your identity** or credential to be used in electronic transactions
- Digital certificate technology has been in existence for over 25 years
- 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.
- Generally a digital certificate provides identity to a person or a server
How is Digital Certificate used?

- **Prove Identity to a peer:**
  - Owner of the certificate can prove possession of the certificate's private key
  - Identity can be validated by checking it is signed by a trusted Certificate Authority

- **Prove authenticity of a digital document:**
  - Programs can be signed by code signing certificates
  - E-mail signatures
  - Certificates are signed by CA certificates

- **Establish a secure connection:**
  - Certificates contain a public key which allows protocols such as SSL and TLS to exchange session keys
What is in a Digital Certificate?

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

Certificate Signature

Version 1, 2, 3

This is the hash/encrypt algorithm used in the signature, eg. sha256RSA

The certificate binds a public key to a subject

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

The private key is NOT in the certificate. It is kept in a key store

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+ extensions currently defined
- Top 7 extensions of interest:
  - **Authority Key Identifier** – Unique identifier of the signer
  - **Subject Key Identifier** – Unique identifier of the subject
  - **Key Usage** – Defines how the public key can be used
    - Digital Signature
    - Key Agreement
    - Certificate Signing
    - Key Encipherment
    - Data Encipherment
    - CRL signing
  - **Subject Alternate Name** – Additional identity information
    - Domain name
    - URI
    - E-mail
    - IP address
  - **Basic Constraints** – Certificate Authority Certificate or not
  - **CRL Distribution** – Locating of Revoked certificate information
  - **Extended Key Usage** – Defines what purposes the public key can be used for
    - Server Auth
    - Client Auth
    - OCSP Signing
    - Code Signing
    - E-mail
    - Timestamping
Example of a x.509 Digital Certificate

Complete your sessions evaluation online at SHARE.org/SanFranciscoEval
Digital Certificates and Certificate stores

• Certificate must be placed in a **certificate store** before it can be used by an application, like Communication Server’s AT-TLS or HTTP server, for secure communication.

• On z/OS, many components call System SSL APIs, which in turn call RACF **R_datalib** callable service to access the certificate store.
  - Application → System SSL → R_DataLib

• Different names:
  - Certificate store = key ring = key file = key database
Types of Digital Certificates

- **Self signed**
  - Self-issued
  - Issuer and subject names identical
  - Signed by itself using associated private key
  - No trusted party involved; trusting subject

- **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.
  - Signed certificate can be an end-entity certificate or a Certificate Authority certificate.
Types of Digital Certificates - 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, EV Certificate…

- **Certificate Authority (CA) certificate**
  - Used to sign other certificates
  - Root CA: the top
  - Intermediate CA: signed by root CA or other intermediate CA
Digital Certificate Formats

- X.509 Digital Certificate can exist in many different forms
  - Single certificate
  - **PKCS Package** - (Public-Key Cryptographic Standards)
    - Developed by RSA
      - **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 is used to convert binary data to displayable text for easy cut and paste
Certificate Revocation

• Normally the lifetime of certificate is the defined **validity period**
• Revocation provides a means for a certificate to become **invalid prior to its validity end date**

• **Reasons for revocation:**
  • Private key associated with the certificate has been **compromised**
  • Certificates are being used for purpose other than what they are defined

• **CRL** – Certificate Revocation List:
  • List of certificates that should no longer be trusted
  • CRL Distribution Point extension in the X.509 certificate gives information about where to locate revocation information for the certificate.

• **OCSP** – Online Certificate Status Protocol:
  • Provides a query function for the revocation status of a certificate
Certificate Chain Validation

Start

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

Is the root CA in my key ring?

Finish

Self signed: Issuer=Subject
Certificate Validation

- **Signature chain validation:**
  - End Entity certificate signature is validated by signer's public key
  - Any intermediate CA certificates signatures are validated against their signer's public key
  - Root CA certificate is validated against its own public key
  - Root CA certificate must be trusted
- **Validity period** – Check if the certificate has expired
- **Status** – Check if the certificate has been revoked:
  - **CRL** - Check if it is on a Certificate Revocation List
  - **OCSP** - Check with the CA which issued this certificate through the Online Certificate Status Protocol
Certificates in SSL 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 material with server’s public key and sends it to server
5. Server decrypts the secret key material 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.
Certificate Life Cycle Planning

- Initially getting a certificate for secure traffic is **only the beginning**
- Must plan for the **certificate life cycle**
- Certificate expiration causes **system outage**
- Things to consider:
  - **How many** certificates are actively used in the system?
  - Certs **locally created** VS Certs by **external provider**
  - Renew using **existing public/private keys** or **require new keys**

- **How to keep track of the expiration dates** of all the certificates in the system?
  - Spreadsheets?
  - Utilities?
  - Automation for renew?
  - Use certificate management vendor products?
RACDCERT Overview

- **RACDCERT** is the primary administrative tool for managing digital certificates using RACF.
- **TSO command** shipped as part of RACF
- Command line interface with ISPF panels
- Certificates and Rings are protected by RACF profiles
- Learn more:
  - RACF Command Language Reference

```
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')
```
RACDCERT Commands

- **Certificate Generation:**
  - RACDCERT GENCERT – Generate key pair and certificate
  - RACDCERT GENREQ – Generate a certificate request

- **Certificate Installation:**
  - RACDCERT ADD – Install a certificate and public/private key

- **Certificate Administration:**
  - RACDCERT LIST – Display certificate information from an installed certificate
  - RACDCERT ALTER – Change certificate installation information
  - RACDCERT DELETE – Delete certificate and key pair
  - RACDCERT CHECKCERT – Display certificate information from a dataset
  - RACDCERT EXPORT – Export a certificate or a certificate and private key
  - RACDCERT REKEY – Renew certificate with new key pair
  - RACDCERT ROLLOVER – Finalize the REKEY process
RACDCERT Commands

- Certificate Ring Administration:
  - RACDCERT **ADDRING** – Create a key ring
  - RACDCERT **CONNECT** – Place a certificate in a key ring
  - RACDCERT **REMOVE** – Remove a certificate from a key ring
  - RACDCERT **LISTRING** – Display key ring information
  - RACDCERT **DELRING** – Delete a key ring
A key ring is a collection of certificates that identify a networking trust relationship.

A certificate must be placed in a key ring before it can be used by middleware applications.

Key Ring Syntax for applications: `<user-id>/<ring-name>`

Types of Certificates in RACF:
- **User** – Directly Associated with one z/OS user ID (end entity)
- **CERTAUTH** – Trusted CA certificate used to verify the peer entity's certificate.
- **SITE** – Certificates associated with an off-platform server or other network identity. SITE certificates bypass the normal certificate chain validation. Private keys can be shared.

Key Rings contain Certificate Usage – The usage assigned to a certificate when it is connected to a key ring indicates its intended purpose.
- **PERSONAL** – Used to identify a local user or server application. Personal usage must be used to get access to the private key.
- **CERTAUTH** – Used to verify the peer entity's certificate. Used to identify the local server's CA certificate.
- **SITE** – Certificate associated with an off-platform server or other network identity. SITE certificates bypass the normal certificate chain validation.
Setup a certificate for SSL handshake requiring server authentication

1. Create a **key ring** (aka key file, certificate store) for server and client

Client Key Ring (Client_A)

- CA Certificate

Server Key Ring (Server_A)

- CA Certificate
- Intermediate CA Certificate
- Server Certificate

**RACDCERT ID(SERVER) ADDRING(‘Server_A’)**

**RACDCERT ID(CLIENT) ADDRING(‘Client_A’)**
Setup a certificate for SSL handshake

2. Generate a certificate signing request

- a certificate signing request (also CSR or certification request) is a message sent from the certificate requestor to a certificate authority to obtain a signed digital certificate
- Contains info about the requestor
  - Identifying information, like subject name
  - Public key (may be generated before the request or generated at the same time as the request)
  - Other credentials or proofs of identity required by the certificate authority
- 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

RACDCERT ID(SERVER) 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(SERVER) GENREQ(LABEL('Server Certificate')) DSN('SUAPC8.HIGHRISK.CERTREQ')
3. Provide certificate request to Certificate Authority for signing.

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIB/TCCAWCAQAwczELMAkGA1UEBhMCVVMxETAPBgNVBAgTCE5ldyBZb3JrMREw
DwYDVQQHEwhFbmRpY290dGUgbG9srG91dGkgYzUwDAYDVQQDEwZnZXJuZXRl
MjElMCQxKjA8DQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMiMS+wcxWogUANwFSZo
0rnztynih3xyCpem54k57iTyVJTCwNmOhiNuCB7CZySoLZG0EAIM3Z1+1s4f93A
KAnzP71JhP4sFCbNvRA96dPfRlx6/dRbAml4IxnMbmB1LJBmqzuebaYTA8+vWzAgMB
AAGgSjBiBkgkghkiG9w0BCQ4x0zA5MBgGA1UdEQQQRMA+CDW15Y29tcGFueS5jb2
HQYDVR0OBBYEFIATTW6P61pujfpaaR4NrdtWczOuMA0GCSqGSIb3DQEBBQUA4GB
AJv6GSrF7Ah51Gg2GnNj7OnizIyNGw2tKvhcOPInzFOBjK8JwE7y913/YJ+px/Yc
ESGB3azSb12deC3XsYh2vqBffMG6j3YVeGhagiAwiLBhziPvtx04LDqd4J9ibQ/GT
+1WWV+/Lm97WjAAbtfZnNS3104XeAHN/RoZ6T9yqxgal
-----END NEW CERTIFICATE REQUEST-----
4. 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.

-----BEGIN CERTIFICATE-----
MIICkTCCAfqAwIBAgIUIUQgG7AAG4hMwDQYJKoZIhvcNAQEFBQAwNTELMAkGA1UE
BhMCVVMxDTALBgNVBAoTBHRlclc3QxBzAVBgNVBAMTDkBhMjQWMB0wNTEAFmFyWjA+DQY
JKoZIhvcNAQEBGzIDBQMuAEggG1oG9xg==
-----END CERTIFICATE-----
Setup a certificate for SSL handshake

5. Add the certificates (CA, Intermediate CA and Server) to the RACF database

   RACDCERT CERTAUTH ADD('suapc8.highrisk.cacert')
   TRUST WITHLABEL('CA Certificate')

   RACDCERT CERTAUTH ADD('suapc8.highrisk.cacert2')
   TRUST WITHLABEL('Intermediate CA Certificate')

   RACDCERT ID(SERVER) ADD('suapc8.highrisk.signed')
   WITHLABEL('Server Certificate')
Setup a certificate for SSL handshake

6. Connect certificates to the **key rings**

RACDCERT ID(SERVER) CONNECT (CERTAUTH LABEL('CA Certificate') RING(Server_A) USAGE(CERTAUTH))

RACDCERT ID(SERVER) CONNECT(CERTAUTH LABEL('Intermediate CA Certificate') RING (Server_A) USAGE(CERTAUTH))

RACDCERT ID(SERVER) CONNECT(ID(SERVER) LABEL('Server Certificate') RING(Server_A) USAGE(PERSONAL) DEFAULT)

Server Authentication requires the CA Certificate to be connected to the client’s key ring

RACDCERT ID(CLIENT) CONNECT (CERTAUTH LABEL('CA Certificate') RING(Client_A) USAGE(CERTAUTH))
Setup a certificate for SSL handshake

7. Permit the client and server application access to their key ring, the certificates

RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)

PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(SERVER) ACCESS(READ)

PERMIT IRR.DIGTCERT.LISTRING CLASS (FACILITY) ID(CLIENT) ACCESS(READ)
Certificate Authority on z/OS: PKI Services

- **PKI Services** provides full certificate life cycle management
  - *Request, create, renew, revoke* certificates
  - Provides certificate status:
    - Certificate Revocation List (CRL)
    - Online Certificate Status Protocol (OCSP)
  - Generation and administration of certificates via customizable web pages
  - **Automatic notifications** or renewal of expiring certificates
Review

- Cryptography
- What are Digital Certificates
- Certificate Types and Contents
- Certificate Formats
- Certificate Validation
- Certificates and SSL
- Certificate Life Cycle
- RACDCERT command
- Using RACDCERT for SSL certificates
- z/OS PKI Services
References

- IBM Education Assistant web site:
  http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp
- RACF web site:
- PKI Services web site:
  http://www.ibm.com/servers/eserver/zseries/zos/pki
- IBM Redbooks
  z/OS V1 R8 RACF Implementation
- Security Server Manuals:
  RACF Command Language Reference
  RACF Security Administrator's Guide
- Cryptographic Server Manual
  Cryptographic Services System Secure Sockets Layer Programming
- RFCs
  RFC2459 - Internet X.509 Public Key Infrastructure Certificate and CRL Profile
  RFC5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
Questions?

Questions or Time for Coffee?

Alyson Comer
Session 12534

Complete your sessions evaluation online at SHARE.org/SanFranciscoEval