IBM Systems Group

RACF and the Digital Certificate

June 14,2004
Session E3

Christine Marusek
RACF Development and Test
IBM Poughkeepsie
marusek@us.ibm.com

Trademarks

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

Agenda

- An overview of cryptography
  - Secret key
  - Public key
  - Applications of cryptographic technologies
  - Confidentiality
  - Non-repudiation
  - Message integrity
  - Secure sockets layer (SSL)
- RACF Support for Digital Certificates on OS/390
  - RACDCERT command syntax
  - Creating certificates in RACF
  - Key Rings
  - Mapping Certificates with Certificate Name Filtering
- What's Next?
- Implementation Considerations

An Overview of Cryptography

Two Families of Cryptographic Algorithms

- There are two distinct branches of cryptographic algorithms:
  - **Shared key** or symmetric
  - **Public key** asymmetric

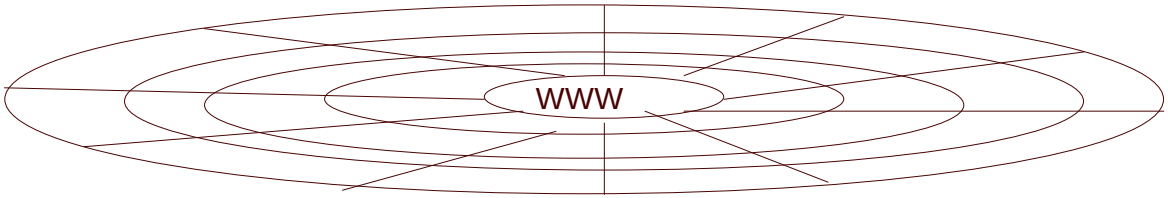
Secret Key Cryptography

- Symmetric or Secret Key Cryptography
  - The sender and receiver share a secret key
  - Transmission of the key occurs “out of band” or encrypted under a different key
  - Usually more efficient than public key algorithms

Public Key Cryptography

- Public Key Cryptography
  - Two keys are involved: One party uses the non-secret, told-to-everyone **public key** and the other uses the private, non-shared **private key.**
  - The public and private keys are complementary: One is used to encrypt data, the other is used to decrypt data.
  - The public and private keys are mathematically related

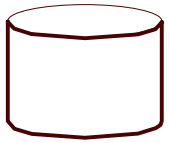
Public Key Cryptography



Mary

Asymmetric Public-keys

Jane

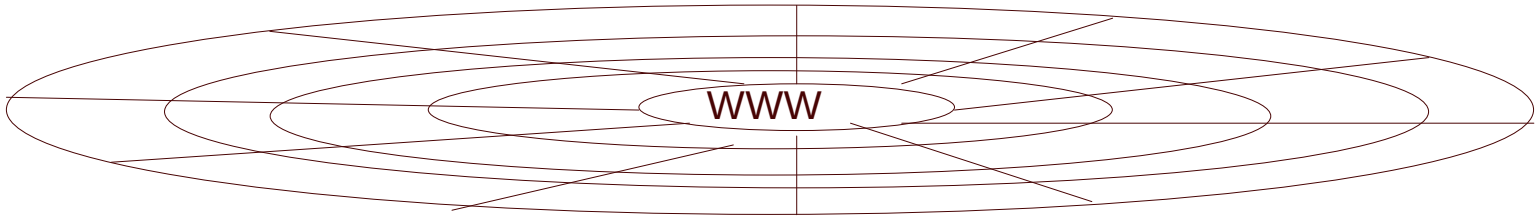


1- Mary encrypts message with Jane's Public-key.

1- Jane decrypts message with her private key.

Non-repudiation

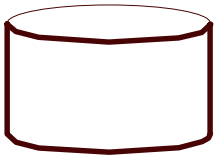
- Public key cryptography can be used to authenticate the origin of data and to prevent a denial of authenticity. This is called **non-repudiation**.
- Non-repudiation could be implemented by having the:
  - Originator of the message first encrypting the message with their private key and then with the public key of the recipient.
  - Receiver of the message first decrypts the message with their private key and then with the public key of the sender
  - Net: Confidentiality and non-repudiation



Mary

Asymmetric Public-keys

Jane



1- Mary encrypts message with her private key.
2- Mary encrypts result with Jane's public key.

1- Jane decrypts message with her private key.
2- Jane decrypts message with Mary's public key obtaining original message.

Public Key vs. Secret Key Cryptography

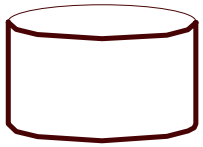
- How do public key and secret key algorithms compare?
  - Key Management
    - Public key crypto requires no shared secret; Secret key requires an initial distribution of a shared secret key
  - Performance
    - Secret key algorithms tend to perform faster
  - Other uses
    - Public key can be used to sigh messages to verify message origin
  - Key strength
    - Bit for bit, secret keys provide more resistance to brute force attacks than public keys

Public Key and Secret Key: Perfect Together

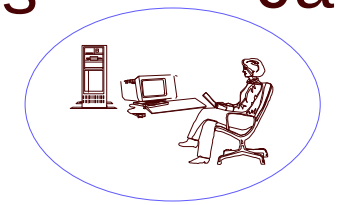
- Many implementations use secret key and public key cryptography together
  - Public key is used to set up an initial session and to protect the transmission of a random secret key
  - The random secret key is used with a secret key algorithm to encrypt the subsequent data flows.
- Advantages
  - Using public key to set up session simplifies key management
  - Using secret key cryptography for the rest of the session has performance advantages

Mary

Asymmetric Public-keys



Jane



1- Mary acquires random DES key.
2- Mary encrypts message with DES key.
3- Mary PK encrypts DES key with her (Mary's) private-key.
4- Mary PK encrypts result with Jane's public-key.

1- Jane PK decrypts with her (Jane's) private-key.
2- Jane PK decrypts with DES key with Mary's public-key.
3- Jane decrypts message with DES key.

Message Signing

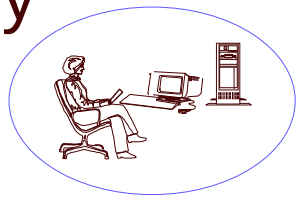
- Message origin can be determined through the use of **message signing.**
- Clear text messages are run through a **message digest** algorithm to produce a message digest or hash.
- Message digests are **one-way**: knowing the message digest reveals nothing about the message.
- If the message is sent with the message digest, then the digest can be calculated and compared at the receiving location. If the locally computed digest matches the one received with the message, then the message has not be altered. This provides **message integrity**.
- Message digests can be **signed**, that is, encrypted using a user's **private** key, providing non-repudiation.

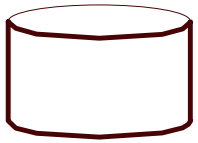
Message Signing

Mary

Asymmetric Public-keys

Jane





1- Mary decomposes message with MD5.
2- Mary PK encrypts MD5 result with her (Mary's) private-key.

Note: Not necessarily encrypted.

1- Jane decomposes message with MD5.
2- Jane PK decrypts Mary's signature with Mary's public-key.
3- Jane compares Mary's MD5 result with her own MD5 result. Equal means message is authentic.

Coke is a refreshing beverage.	Mary's signature

Summary

- Cryptography is the core technology for enabling data integrity, non-repudiation, and data confidentiality across hostile media.
- Understanding cryptography and its applications is essential to properly implement, evaluate, and audit internet-based applications.
- Security for Internet applications must be a key initial design point.

Certificates

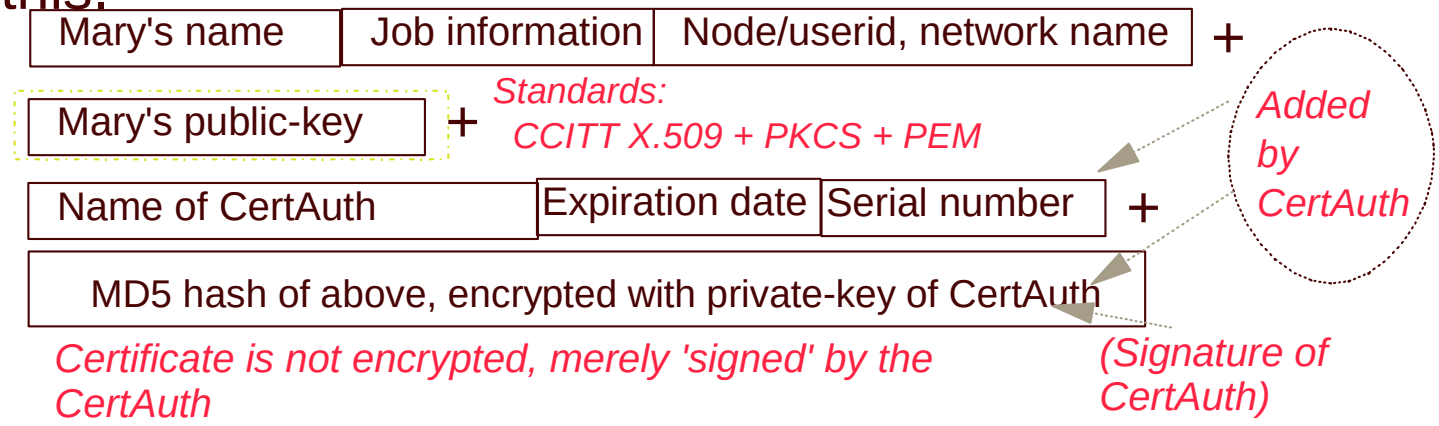
- Where do I go to find someone's public key?
  - Look in a well-known and trusted place.
  - Trusted directory
  - Public publication
- Allow the user to tell you their public key.
  - How do you know it's really theirs?
  - How do you know that it hasn't been replaced in transit?
- Certificate authorities are trusted organizations who vouch for public keys.
  - Public keys are delivered via certificates, which are signed with the private key of the certificate authority.
  - User's manage their certificates.

Certificates…

Mary

Certifying Authority (CertAuth)

Mary's Digital Certificate looks about like this.

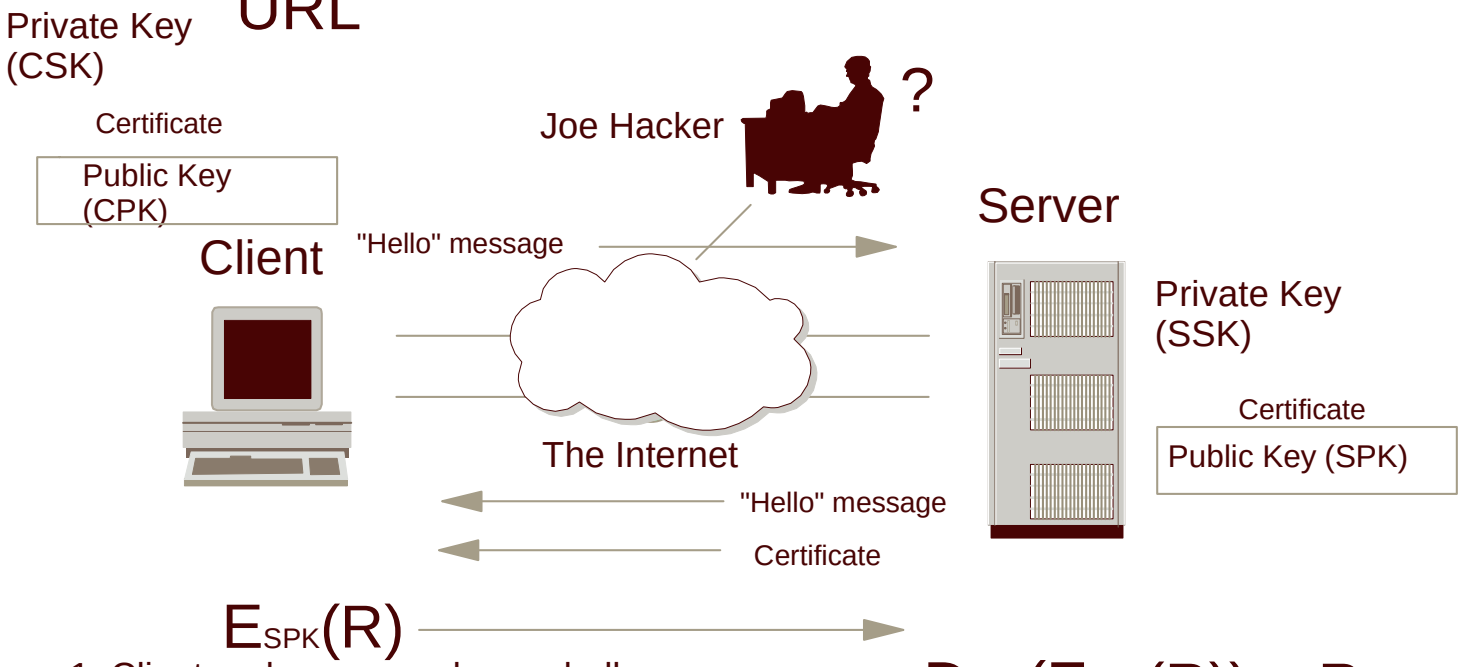


Secure Sockets Layer (SSL)

- Secure Sockets Layer (SSL) is a security protocol developed by Netscape and RSA. SSL is based on public key certificates and provides:
  - Confidentiality
  - Server validation
  - Optionally, client validation

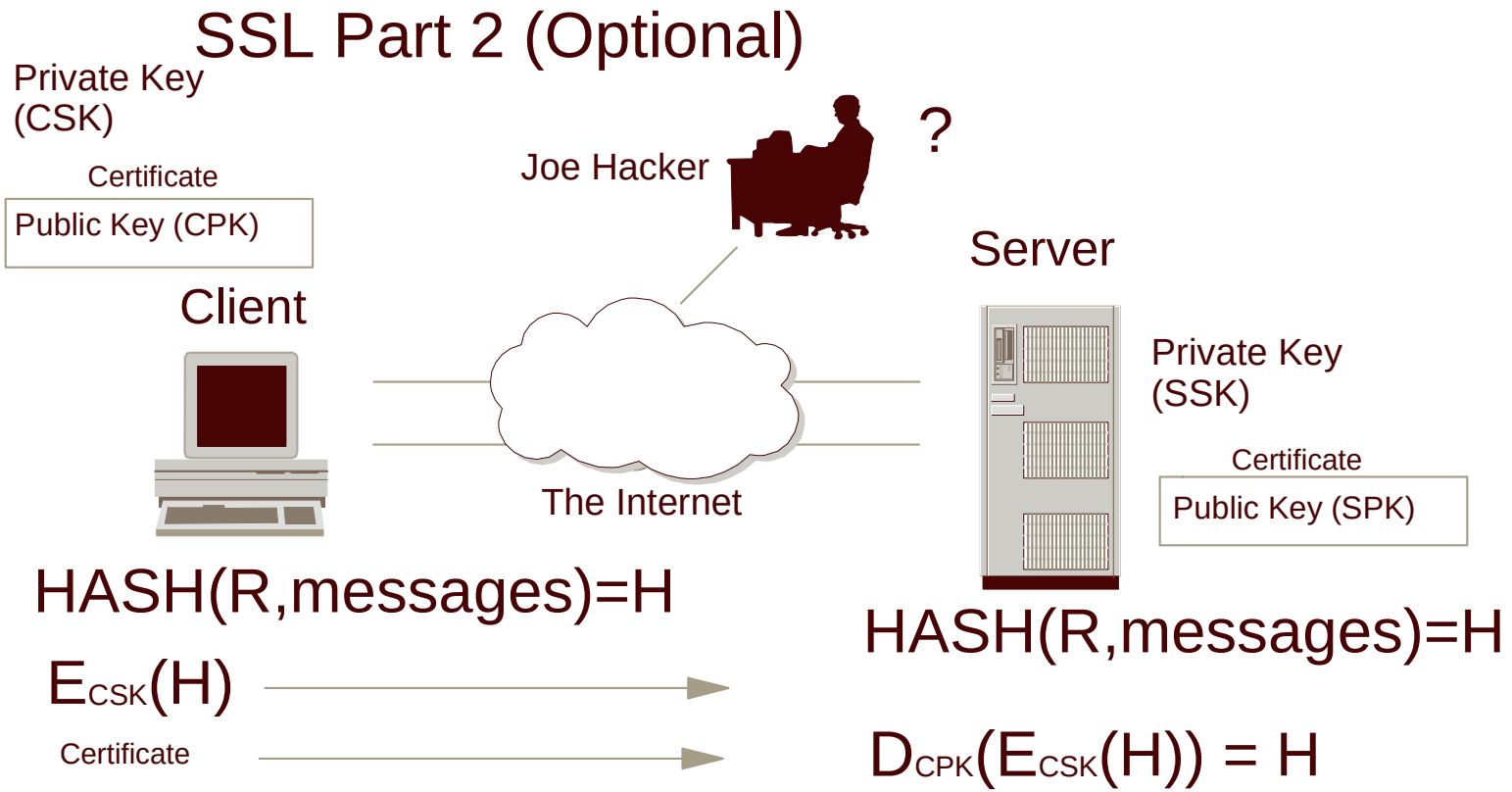
Secure Sockets Layer (SSL)…

SSL Part 1 - Kicked off by https:// URL



1. Client and server exchange hellos
2. Server sends certificate to client. Client "validates" certificate
3. Client encrypts random # with server's public key. Sends to server
4. Server decrypts random # with own private key

Secure Sockets Layer (SSL)…



1. Client and server compute hash of R plus all messages
2. Client encrypts hash with own private key. Sends to server with certificate
3. Server "validates" certificate, decrypts hash with client's public key.

RACF's Support For Digital Certificates

RACF's Support for Digital Certificates

- With OS/390 Security Server R4, RACF can be used to associate certificates to a RACF user ID
  - General resource class DIGTCERT
    - Segment CERTDATA contains the certificate
    - APPLDATA contains the user associated with the certificate
    - UACC contains the TRUST status of the certificate
    - User profile repeat group points to the certificate
  - RACF command RACDCERT to manage the certificates

RACF's Support for Digital Certificates

- Certificates are uniquely identified by the issuer's distinguished name and serial number
- The RACDCERT Command
  - ADD, ALTER, LIST and DELETE a certificate in the RACF database associated with an OS/390 user ID
  - Generate certificates (GENCERT) and certificate requests (GENREQ) and EXPORT certificates in the RACF database associated with an OS/390 user ID
  - Create and administer certificate rings (ADDRING, LISTRING, DELRING, CONNECT and REMOVE) associated with an OS/390 user ID
  - Map certificates (MAP, ALTMAP, LISTMAP, DELMAP) associated with an OS/390 user ID

The RACDCERT Command

```
RACDCERT
      [ID(UserID) | CERTAUTH | SITE | MULTIID ]
      [  LIST [LABEL('label-name') | ALL]
       | ADD('Dataset-Name')
              [ TRUST | NOTRUST ]
       | ALTER  [ (SERIALNUMBER(Serial-Number)
            [ ISSUERSDN('Issuer's Distinguished Name')])]
           [ TRUST | NOTRUST | HIGHTRUST]
           [ NEWLABEL('label-name')]
       | DELETE [ (SERIALNUMBER(Serial-Number)
           [ ISSUERSDN('Issuer's Distinguished Name')])]
       | CHECKCERT (data-set-name)
           [PASSWORD('pkcs12-password')]
```

The RACDCERT Command Continued

```
RACDCERT
    |  GENCERT [(request-data-set-name) ]
        [ SUBJECTSDN(
            [CN('common-name')]
            [ T ('title') ]
            [OU('organizational-unit-name1','organizational-unit-
              name2,…)]
                ..
            [C('country') ])
    [ SIZE(keysize)]
    [ NOTBEFORE(DATE(yyyymmdd)]TIME(hh:mm:ss)])]
    [ SIGNWITH([CERTAUTH |SITE] LABEL('labelname'))]
    [ ICSF | PCICC ]
    [ KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN CERTSIGN)]
    [ ALTNAME (IP(numeric IP addr)
                DOMAIN ('internet-domain-name')
                EMAIL('email addr')
                URI('universal-resource-identifier'))]
```

The RACDCERT Command Continued….

```
RACDCERT
      | EXPORT (LABEL('label-name'))
            DSN(output-data-set-name)
            FORMAT(CERTDER | CERTB64 |PKCS12DER|PKCS12B64)]
            [PASSWORD('pkcs12-password')]

      | GENREQ (LABEL('label-name'))
            DSN(output-data-set-name)


       | DEBUG
```

The RACDCERT Command Continued….

RACDCERT

| ADDRING (ring-name)

| DELRING (ring-name)

| LISTRING (ring-name | *)

| CONNECT [ID(USER-ID) | SITE | CERTAUTH]

LABEL('label-name')
RING(ring-name)
[DEFAULT]
[USAGE(PERSONAL | SITE | CERTAUTH)])

| REMOVE [ID(USER-ID) | SITE | CERTAUTH]

LABEL('label-name')
RING(ring-name)

The RACDCERT Command Continued….

RACDCERT

| MAP [(data-set-name)]

[SDNFILTER('subject's-distinguished-name-filter')]
[IDNFILTER('issuer's-distinguished-name-filter')]
[CRITERIA(criteria-profile-name-template)]
[WITHLABEL('mapping-label-name')]
[TRUST | NOTRUST]

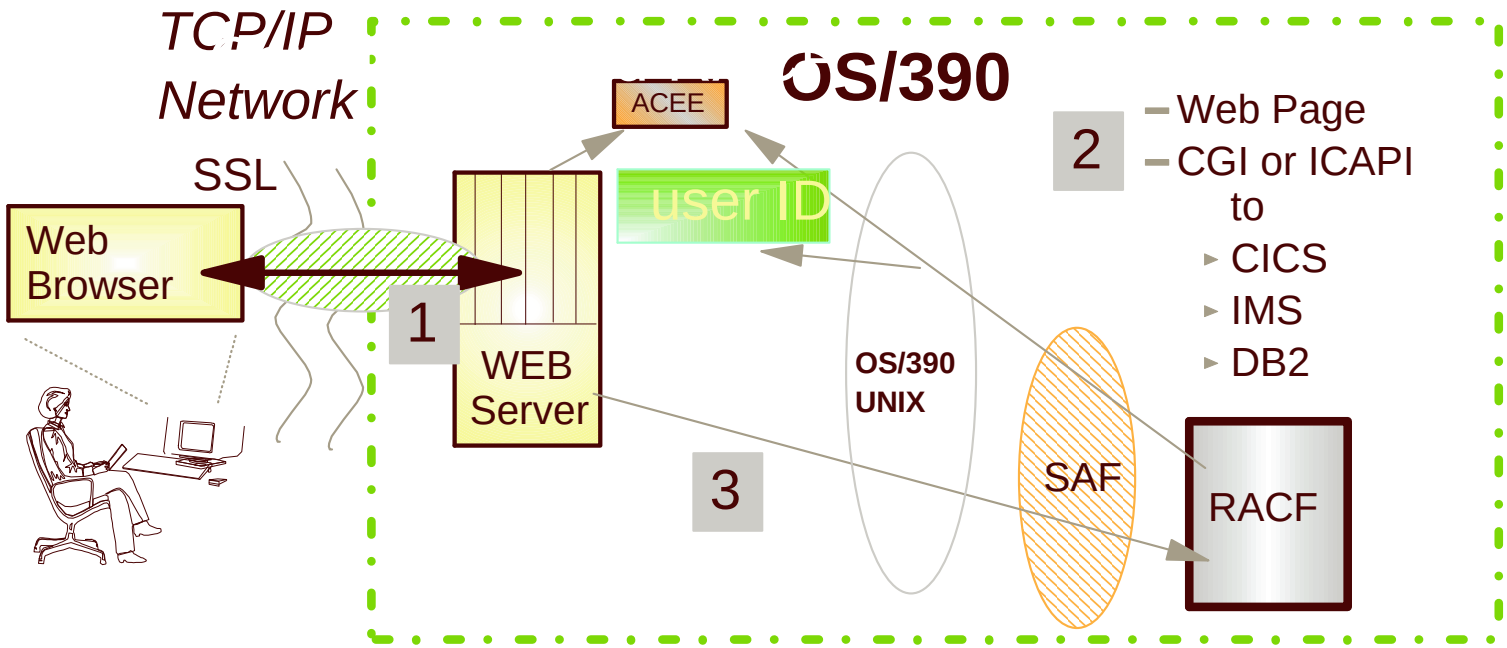
| ALTMAP [LABEL('mapping-label-name'))]

[NEWCRITERIA(criteria-profile-name-template)]
[NEWLABEL('new-mapping-label-name')]
[TRUST | NOTRUST]

| DELMAP (LABEL('mapping-label-name'))]

| LISTMAP (LABEL('mapping-label-name'))]

The Big Picture



1- User authenticates to Secured Sockets Layer (SSL)
2- User requests OS/390 secured resource via browser
3- Web Server invokes RACF via UNIX System Services to build local security context (ACEE),
passing SSL validated certificate instead of prompting for user ID & password

Creating Certificates Using RACF

Creating Certificates Using RACF

- **RACDCERT has functions that allow :**
  - The generation of certificates and certificate requests
  - The definition of certificate authority (CERTAUTH) and site (SITE) certificates.
  - The aggregation of certificates into key rings
  - The importation of PKCS-12 certificates
  - The renaming of the LABEL that is associated with a certificate
- **RACF callable service to retrieve certificate information**
- **RACF database unload (IRRDBU00) and modified RACF SMF unload (IRRADU00) records**
- **BLKUPD allows the specification of mixed case ENTRY**

GENCERT: Creating a Certificate

- **The RACDCERT GENCERT function creates a public/private key pair and a digital certificate.**
  - X.509-style keywords are used to specify certificate information, such as:
  - Subject's distinguished name
    - Default: User's name
  - Certificate validity dates (start date/time & end date/time)
    - Default: Current date as start, one year from start date as the end date
  - Size of key
    - Range: 353-1024; Default: 1024
  - Signature
    - Default: Self-signed

RACDCERT Example 1

- **RACDCERT Example #1- Create a certificate for the user ID (SRVR01) which is the user ID associated with the inventory server:**

```
RACDCERT ID(SRVR01)
         GENCERT
         SUBJECTSDN(CN('co-name.com')
                    OU('Inventory')
                     C('US'))
         WITHLABEL('Inventory Server')
```

- Notes: SSL convention is that the "common name" (CN) is the same as the domain name

RACDCERT Example 2: Part 1

- **RACDCERT Example #2 (Part 1)- Creating the CERTAUTH certificate**

```
RACDCERT CERTAUTH
         GENCERT
         SUBJECTSDN(CN('Local CertAuth')
                    OU('My Company')
                     C('US'))
         WITHLABEL('XYZZY CertAuth')
```

  - Note: This certificate is a certificate authority certificate (CERTAUTH), which can be used to sign other certificates.

RACDCERT Example 2: Part 2

- **RACDCERT Example #2 (Part 2)- Create a certificate for the user ID (SRVR01) which is the user ID associated with the inventory server, this time signed by a certificate authority certificate managed by RACF:**

```
RACDCERT ID(SRVR01)
         GENCERT
         SUBJECTSDN(CN('co-name.com')
                    OU('Inventory')
                     C('US'))
         WITHLABEL('Inventory Server')
         SIGNWITH(CERTAUTH
             LABEL('XYZZY CertAuth'))
```

A Word About Distinguished Names

- **X.509 certificates are identified by distinguished names, which are multi-part hierarchical names**
- **Distinguished names consist of these parts:**
  - Common name (CN), e.g. "Lamont Cranston"
  - Title (T), e.g. "RACF Developer"
  - One or more organizational units (OU), e.g. "RACF Development", "zOS Development", "Server Group"
  - Organization (O), e.g. "IBM Corporation"
  - Locality (L),e.g. "Poughkeepsie"
  - State or Province (SP), e.g. "New York"
  - Country (C), e.g. "US“
- **Think of the distinguished name as a hierarchical name**
  - Lamont Cranston\RACF Developer\RACF Development\zOS Development\Server Group\IBM Corporation\Poughkeepsie\New York\US

GENREQ: Creating a Certificate Request

- **RACDCERT GENREQ can be used to create a certificate request which can be submitted to an off-platform certificate authority to sign the certificate.**

```
RACDCERT ID(SRVR01)
         GENREQ(LABEL('Inventory
  Server'))
         DSN('MARKN.INVSERV.GENREQ')
```

EXPORT: Writing a Certificate to a Data Set

- **RACDCERT EXPORT can be used to extract a certificate from the RACF data base and place it into a data set**

```
RACDCERT ID(SRVR01)
         EXPORT(LABEL('Inventory Server'))
         DSN('MARKN.INVSERV.GENREQ')
```

- **One in a data set, the certificate can be moved to the OS/390 HFS, where it can be loaded by a web browser for use on a client system.**

RACDCERT certificate generation

- Support for KeyUsage extension: handshake, dataencrypt, docsign, certsign
- Support for subjectAltName extension: IP address, Domain name, EMAIL address, URI
- Certificate export format, PKCS12. Packages certificate chain and user's private key to allow generation of client certificates and standard usage by web browsers.
- Ability to mark Certifying Authority (CA) certificate as "highly trusted"

Key Rings

- **Certificates are collected into sets called *key rings*. Key facts about key rings:**
  - Each key ring is associated with an OS/390 user ID.
  - User IDs may have more than key ring.
  - CertAuth and Site certificates are associated with the "reserved" user IDs "irrcerta" and "irrsitec"
  - Key rings are identified by a 1 to 237 character ring name
  - Authority to manage key rings may be distributed. However, the system security administrator has the ability to define the superset of key ring contents.

Advantages of RACF Key Rings

- **Why are RACF key rings superior to other platforms key ring implementations?**
  - Application/server owners may implement a trust hierarchy that is a subset of the installation policy. That is, they may not allow non-approved certificate authorities.

ADDRING: Creating a Key Ring

- **RACDCERT ADDRING is used to create a key ring. To create a key ring called "RING01" for the user ID SRVR01, the command is:**

```
RACDCERT ID(SRVR01)
         ADDRING(RING01)
```

CONNECT: Connecting a Certificate to a Key Ring

- **RACDCERT CONNECT is used to add a certificate to a key ring. To connect the certificate labeled "Inventory Server" to a key ring called RING01 that is associated with the user ID SRVR01, the command is:**

```
RACDCERT ID(SRVR01)
         CONNECT(
             LABEL('Inventory Server')
             RING(RING01)
```

Connect: Example 2

- **To connect a certificate authority certificate to a key ring, the CERTAUTH keyword is added to the RACDCERT CONNECT command:**

```
RACDCERT ID(SRVR01)
         CONNECT(CERTAUTH
             LABEL('CertAuth 1')
             RING(RING01)
             DEFAULT)
```

REMOVE: Taking a Certificate out of a Key Ring

- **RACDCERT REMOVE is used to take a certificate out of a key ring. To remote the certificate labeled "Inventory Server" from key ring RING01 for the user ID SRVR01, the command is:**

```
RACDCERT ID(SRVR01)
                  REMOVE(
               LABEL('Inventory Server')
               RING(RING01))
```

DELRING: Deleting a Key Ring

- **RACDCERT DELRING is used to delete a key ring. To delete the key ring called "RING01" for the user ID SRVR01, the command is:**

```
RACDCERT ID(SRVR01)
         DELRING(RING01)
```

- **DELRING does not delete the certificates themselves. It merely deletes the relationship between the certificate and the key ring.**

LISTRING: Show the Contents of a Key Ring

- **RACDCERT LISTRING is used to lis the contents of a key ring. If a listing of all rings associated with a user ID is desired, then LISTRING(*) is specified.**
- **LISTRING displays:**
  - The ring name
  - The label of the certificate
  - The DEFAULT status of the certificate within the ring
  - The usage within the ring

Authority Checking

- **Users with SPECIAL authority can do practically anything to anybody.**
- **For everyone else, authority checks are performed against the resource IRR.DIGTCERT.<function>. The authority required to this resource is:**
  - READ to perform the function on their own certificate or key ring,
  - UPDATE to perform the function on the certificate or key ring of another, and
  - CONTROL to perform the function on a certificate authority or site certificate.

A Word About “irrcerta”, “irrsitec”, and “irrmulti”

- **The three special user IDs "irrcerta“, "irrsitec“, and “irrmulti” are anchor points for certificate information. Key points about these IDs:**
  - They are marked as REVOKED in the RACF data base
  - RACROUTE REQUEST=VERIFY requests fail for these user IDs as they have no default group
  - ADDUSER, DELUSER, and LISTUSER may not be used against these specific IDs. Note that a "LISTUSER *" will list the information about these IDs
  - The SEARCH command for CLASS(USER) will return these user IDs if they fall within the SEARCH criteria
  - ICHEINTY NEXT and RACROUTE EXTRACTN processing will return these IDs if they match the selection criteria

Certificate Name Filtering

An Issue With Certificate Management

- **To enable e-business:**
  - Every user must be identified
  - Every user's certificate must be installed into RACF
  - Each user can have many certificates
  - ... which means lots of certificates and certificate management work!
- **The RACF Solution: Certificate Name Filtering**
  - Allows the definition of a set of rules ("filters") which are used to associate certificates with user IDs
  - Certificates are not stored by RACF
  - Eliminates expiration problems
  - Individual accountability is maintained
  - Access by "shared" user IDs can be restricted

Certificate Name Filtering

- **RACDCERT is used to create a filter and map it to a RACF user ID**
- **Filtering is based on the subject's name and the issuer's name from a certificate (the X500 names)**
  - subject's name || issuer's name
- **RACDCERT command or ISPF panels can be used**
  - DIGTNMAP class contains the mapping
- **Each filter must be unique**
- **Other criteria such as application ID or system name can be used in determining the user ID**
- **DIGTCRIT class is used for additional criteria**

Certificate Name Filtering…

- **Old-style certificate lookup is done first (DIGTCERT)**
- **If there is no matching certificate, then RACF searches for a filter, starting from the most specific to the least specific:**
  - Full subject-name with full isser-name
  - Shrinking subject-name with full issuer-name
  - Shrinking subject-name alone
  - Shrinking issuer-name alone
- **The user ID is taken from the first matching filter**
  - If the user ID is MULTIID, additional criteria is user (DIGTCRIT)

Example: Certificates and FIlters

- **A customer's certificate**
  - Subject: CN=Sid Shopper.OU=Customer.O=Ohio.C=US
  - Issuer: OU=BobsMart Subscriber.O=Verisign,Inc.L=Internet
- **A sales clerk's certificate**
  - Subject: CN=Cal Clerk.OU=Reg.OU=Emp.O=Ohio.C=US
  - Issuer: OU=BobsMart Subscriber.O=Verisign,Inc.L=Internet
- **A store manager's certificate**
  - Subject: CN=Mary Manager.OU=Mang.OU=Emp.O=Ohio.C=US
  - Issuer: OU=BobsMart Subscriber.O=Verisign,Inc.L=Internet

Example: Certificates and Filters…

- **Map all customers in Ohio to a state user ID OHIOUSER**
  - OU=Customer.O=Ohio.C=US||OU=BobsMart Subscriber.O=Verisign,Inc.L=Internet
- **Map all BobsMart certificates to user ID ALLB**
  - ||OU=BobsMart Subscriber.O=Verisign,Inc.L=Internet
- **Map Mary's certificate to her user ID MARYM**
  - CN=Mary Manager.OU=Mang. OU=Emp.O=Ohio.C=US|| OU=BobsMart Subscriber. O=Verisign,Inc.L=Internet

Examples: Certificates and Filters…

- **Subject: CN=Sid Shopper.OU=Customer.O=Ohio.C=US**
- **Issuer: OU=BobsMart Subscriber.O=Verisign,Inc.L=Internet**
- **InitACEE would check (after the DIGTCERT class):**
  - DIGTNMAP class profiles for these values:
    - CN=Sid Shopper.OU=Customer.O=Ohio.C=US||OU=BobsMart Subscriber.O=Verisign,Inc.L=Internet
    - OU=Customer.O=Ohio.C=US||OU=BobsMart Subscriber.O=Verisign,Inc.L=Internet
- **This finds a match with OHIOUSER. Otherwise, RACF would have continued to look by:**
  - Continuing to shrink the subject-name with full issuer-name
  - Shrinking subject-name alone
  - Shrinking issuer-name alone

Auditing Considerations

- **Issuer's name and subject's name (X.500 name) are kept for users identified through mappings**
  - Passed by initACEE to RACROUTE REQUEST=VERIFY (X500NAME=)
  - Passed to ICHRIX01/02 exits in RIXP (RIXX5PRP)
  - Pointed to by ACEE (ACEEX5PR)
  - Added to SMF TYPE80 records written for user and supported by SMF unload
  - Part of ENVR objects
  - Support indicated by bit in RCVT (RCVTX500)
- **Enhanced auditing of initACEE failures (TYPE80s and ICH408Is)**
  - Unknown certificate
  - Certificate known, but not trusted

Restricted User IDs

- **Restricted User IDs are user IDs which do cannot gain access to resources based on:**
  - Global access table
  - Universal access (UACC)
  - ID(*)
- **Enabled by specifying:**
  - ADDUSER user-ID RESTRICTED
  - ALTUSER user-ID [RESTRICTED | NORESTRICTED]
- **LISTUSER shows "RESTRICTED" attribute**

WHAT'S NEXT?

Enhancements starting in OS/390 R10 - PKI

- **A web based Certificate Authority application that utilizes RACF to generate and deliver certificates to end users.**
- **Provided as an SPE on OS/390 V2R10**
  - RACF APAR OW45211, PTF UW74164
  - SAF APAR OW45212, PTF UW74113
- **Roughly Equivalent to the IBM HTTP Server's CaServlet**
- **Sample materials**
  - Downloadable from RACF webpage: http://www.ibm.com/servers/eserver/zseries/zos/racf/webca.html
  - HTML pages and certificate templates contained in a conf file
  - Connector REXX execs (CGIs) to process the above

Starting in z/OS V1R3 and beyond: PKI Services

- **A new component of the z/OS Security Server**
  - Full certificate life-cycle management
  - Web-based user and administration process
  - Automated approval mechanism
  - Customizable
  - Support for end user and/or administrator revocation
  - Always enabled

Implementation Considerations

Remote Sharing Considerations

- **RRSF does not propagate private key information**
  - CERTPRVK (private key)
  - CERTPRVS (private key size)
  - CERTPRVT (private key type)
- **Customers must ensure that RRSF has identical propagation rules for the DIGTCERT and DIGTRING class**
  - Recommendation: Define the propagation with the profile AUTODIRECT.node.DIGT*.APPL to have a single set of rules for both classes.

ICSF Considerations

- **IBM recommends the use of the S/390 Integrated Cryptographic Support Facility (ICSF) for the storage of private keys.**
- **ICSF ensures that the user's private key is stored within ICSF, encrypted under the ICSF master key for the installation**
- **If the RACF database is shared among systems, then all of the ICSFs must have the same master key**
  - Master keys may be managed using the Trusted Key Entry (TKE) workstation
- **ICSF is not required; it is used if available (and configured)**
  - If ICSF is not being used, BSafe (software encryption) is used

ICSF Considerations

- **ICSF-stored private keys are requested by using the "ICSF" keyword on RACDCERT GENCERT and RACDCERT ADD.**
- **Non-ICSF-managed private keys may be moved into ICSF storage by:**
  - RACDCERT EXPORTing the certificate to a data set and then
  - Re-ADDing the certificate specifying the "ICSF" keyword.
    - Since the subject's distinguished name, public key, and issuer's distinguished name are the same, RACF replaces the certificate, and migrates the private key to ICSF
    - Note that the reverse process is not possible.

Summary

- **Digital certificates are the backbone of the security for e-business applications**
  - RACF's support for server-side digital certificates is a natural evolution of the existing RACDCERT support, providing:
  - Confidentiality of private keys
  - An implementation of key rings that allows central control of the security policy
  - Callable service support which enables CDSA-support (OCSF) for the RACF-managed certificate data

Reference Material

LISTRING Output

```
                                  Ring:
                              >GEORGEMsNewRing01<
Certificate Label Name                 Cert Owner     USAGE      DEFAULT
 ------------------------------       ------------   --------   -------
 New Cert Type - Ser # 00               ID(GEORGEM)    PERSONAL     YES
 New Type Cert - VsignC1                ID(GEORGEM)    CERTAUTH     NO
  New Type Cert - VsignC2               ID(GEORGEM)    SITE          NO
  65                                    ID(JOHNP)      PERSONAL      NO

                                  Ring:
                               >GEORGEMsRing<
Certificate Label Name                 Cert Owner     USAGE      DEFAULT
--------------------------------      ------------   --------   -------
  GEORGEM's Cert # 48                   ID(GEORGEM)    PERSONAL     NO
  GEORGEM's Cert # 84                   ID(GEORGEM)    PERSONAL     NO
 New Cert Type - Ser # 00               ID(GEORGEM)    PERSONAL     YES

                                  Ring:
                              >GEORGEMsRing#2<
Certificate Label Name                 Cert Owner     USAGE      DEFAULT
--------------------------------      ------------   --------   -------
  GEORGEM's Cert # 84                   ID(GEORGEM)    PERSONAL     NO

                                  Ring:
                              >GEORGEMsRing#3<
                      *** No certificates connected ***
```

Authority Checking Summary

Function	READ	UPDATE	**CONTROL**
ADD	Add a cert to one's own user ID	Add a cert to someone else's ID	**Add a SITE or CERTAUTH cert**
ALTER	Change the trust status or label of one's own cert	Change the trust status or label of someone else's cert	**Change the trust status or label of a SITE or CERTAUTH cert**
DELETE	Delete one's own cert	Delete someone else's cert	**Delete a SITE or CERTAUTH cert**
EXPORT	**Export one's own cert**	**Export someone else's cert**	**Export a SITE or CERTAUTH cert**
GENREQ	**Generate a request based on one's own cert**	**Generate a request based on someone else's cert**	**Generate a request based on a SITE or CERTAUTH cert**
LIST	List one's own cert	List the someone else's cert	**List a SITE or CERTAUTH cert**

Authority Checking Summary

Function	READ	UPDATE	CONTROL
ADDRING	Create a key ring for one's own ID	Create a key ring for someone else's ID	N/A
CONNECT	Place ones own certificate in one's own ring	Place a CERTAUTH or SITE cert in one's own ring	Place a certificate into someone else's ring
DELRING	Delete one's own key ring	Delete someone else's key ring	N/A
LISTRING	List one's own key ring	List someone else's key ring	N/A
REMOVE	Remove a certificate from one's own key ring	Remove a SITE or CERTAUTH certificate from one's own key ring	Delete a certificate from the ring of another

References

- ***Network Security, Private Communication in a Public World***, Charlie Kaufman, Raida Perlman, and Mike Speciner, Prentice Hall, 1995
- ***The Codebreakers: The Story of Secret Writing***, David Kahn, Macmillan, 1997
- ***Crypto***, Steven Levy, 2001
- ***Cryptography Frequently Asked Questions*** (FAQ), Internet

References

- References
- ***Network Security, Private Communication in a Public World***, Charlie Kaufman, Raida Perlman, and Mike Speciner, Prentice Hall, 1995
- ***The Codebreakers: The Story of Secret Writing***, David Kahn, Macmillan, 1997
- ***Crypto***, Steven Levy, 2001
- ***Cryptography Frequently Asked Questions*** (FAQ), Internet
- ***RACF and the Digital Certificate***, Jim Sweeny and Mark Nelson, Technical Support Magazine, October, 1999, available at http://www.naspa.com/PDF/99/T9911002.pdf

References…

- **RACF Web Page**
  - http://www.ibm.com/servers/eserver/zseries/racf
  - Latest release information on RACF
  - Presentations from this and other RACF meetings and conferences
  - Links to announcement letters
  - RACF user group information
  - Useful RACF tools

References…

- **Sample code on the web page:**
  - CUTPWHIS: removes non-usable passwords from the password history
  - DBSYNC: compares two RACF data bases
  - IRRHFSU: unloads the UNIX HFS security info
  - KMIGRATE: migrates existing DCE and MVS users to a Kerberos registry
  - RAC: an OS/390 UNIX utility which allows RACF commands to be executed from the OS/390 UNIX environment
  - RACFDB2: migrates DB2 access control to RACF
  - RACFICE: creates reports
  - OS390ART: a web-based reporting tool
  - RACTRACE: a tracing facility
  - PKISERV: create client certificates
  - PWDCOPY: copies passwords from one RACF data base to another RACF data base

Disclaimer

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