

Securing and Encrypting Network Traffic with z/OS Communications Server and Policy Agent

Security Workshop

x.509 Certificates for z/OS Communications Server



IBM Washington System Center
IBM Technical Sales Support

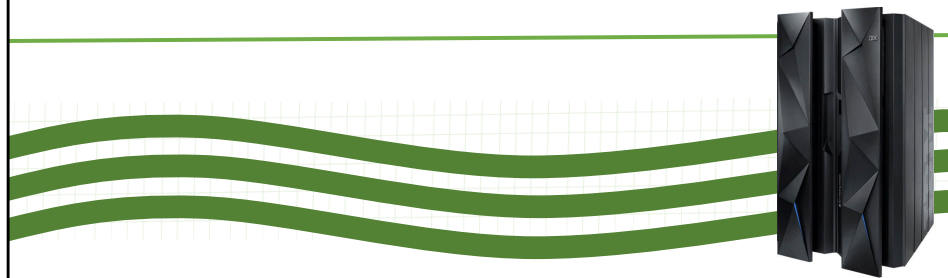
Trademarks

- **The following are Registered Trademarks of the International Business Machines Corporation in the United States and/or other countries.**
 - IBM
 - z/OS
- **The following are trademarks or registered trademarks of other companies.**
 - Microsoft is a registered trademark of Microsoft Corporation in the United States and other countries.
- All other products may be trademarks or registered trademarks of their respective companies.
- Refer to www.ibm.com/legal for further legal information.
- OSA-Express Features
- There are many different types of OSA-Express features. In this document where OSA is mentioned it refers to an OSA port on any of the OSA-Express features unless a specific OSA-Express feature is mentioned.

Agenda

- Encryption
- Encryption Keys and Certificates
- Digital Key Storage
- Certificate Usage and RACF Commands
- Certificate Creation
- Host Name Check

Encryption



006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 4

Confidentiality and Privacy

- We encrypt data to make it:
 - Confidential
 - Private
- Confidentiality and Privacy are required by Security Mandates for certain types of data:
 - Data Payload itself (i.e., Credit Card Data with Payment Card Industry - PCI - Mandates)
 - Passwords and optionally Userids (PCI, NIST, etc.)
 - Encryption Keys for maintaining Data Privacy
- Cryptographic Hardware and Software can be validated according to standards listed in FIPS140-2. See the following URLs:
 - <http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401val2014.htm>
 - <http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexa.pdf>

006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 5

The Cryptographic Module Validation Program (CMVP - www.nist.gov/cmvp) validates cryptographic modules to FIPS PUB 140-2 and other cryptography based standards. The CMVP is a joint effort between NIST and the Communications Security Establishment Canada (CSEC - www.cse-cst.gc.ca). Modules validated as conforming to FIPS PUB 140-2 are accepted by the Federal agencies of both countries for the protection of sensitive information (United States) or Designated information (Canada).

In the CMVP, vendors of cryptographic modules use independent, accredited testing laboratories to have their modules tested. Organizations wishing to have validations performed would contract with the laboratories for the required services.

Please see following excerpt from <http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401val2014.htm>

"It is important to note that the items on this list are cryptographic modules. A module may either be an embedded component of a product or application, or a complete product in-and-of-itself. If the cryptographic module is a component of a larger product or application, one should contact the product or application vendor in order to determine if their product utilizes an embedded validated cryptographic module. There is inevitably a larger number of security products or applications available which use embedded validated cryptographic modules, than the number of modules which are found in this list. In addition, it is possible that other vendors, who are not found in this list, might incorporate a validated cryptographic module from this list embedded into their own products.

When selecting a module from a vendor, verify that the product or application that is being offered is either a validated cryptographic module itself (e.g. VPN, SmartCard, etc) or the product or application uses an embedded validated cryptographic module (toolkit, etc). Ask the vendor to supply a signed letter stating their application, product or module is a validated module or incorporates a validated module, the module provides all the cryptographic services in the solution, and reference the modules validation certificate number from this listing."

Annex A (<http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexa.pdf>) provides a list of the Approved security functions applicable to FIPS PUB 140-2. The categories include symmetric key, asymmetric key, message authentication and hashing.

Approved Symmetric Key Technologies: AES (Advanced Encryption Standard), TDES (Triple Data Encryption Algorithm or Triple Data Encryption Standard), or, EES (Escrowed Encryption Standard (EES).

Approved Asymmetric Key Technologies include:

- DSS – DSA (Digital Signature Standard, RSA and ECDSA)
- Secure Hash Standard (SHS) (SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512)
- Random Number Generators (RNG and DRBG)
- Message Authentication (Triple-DES MAC, CMAC, CCM, GCM, GMAC and HMAC)
- Key Management
 - Recommendation for Key Derivation Using Pseudo random Functions

Encryption Keys and Certificates

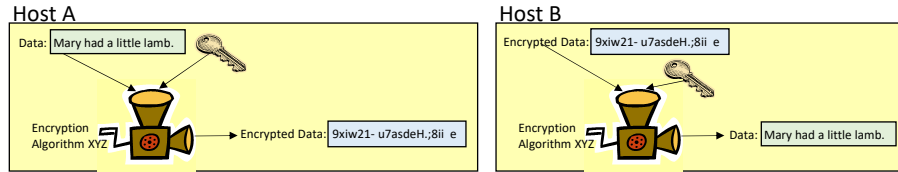


006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 6

Symmetric Keys



- Symmetric Keys are identical on both sides of the connection.
 - Host A has a piece of data and an encryption key.
 - Those are input to an encryption algorithm.
 - An encrypted piece of data is output.
 - Host A sends the encrypted data to Host B.
 - Host B inputs the encrypted data and the same key into the same encryption algorithm.
 - The output data is identical to the original data.
- The algorithms are publicly available.
 - Cryptographic algorithms are mentioned but algorithm description and details are outside the scope of these presentations. Web searches provide information on algorithms.
 - Also referred to as cipher suites.
- The key must be kept secret so that only Host A and Host B know what it is.
 - One partner creates the key.
 - How does that host securely send the key to the partner host?
- Symmetric keys are more efficient than asymmetric keys.
 - Less overhead for encryption.

006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 7

Symmetric key usage is more efficient than asymmetric key usage.

The trouble is getting the key on both sides for usage.

Encryption and Hash algorithms:

3DES Also known as triple DES, this encryption method uses three DES operations on a single data block with three different keys. Provides greater security than single DES.

AES Cipher Block Chaining (CBC) (AES_CBC) mode The AES algorithm using the CBC mode. IP security for z/OS Communications Server supports AES_CBC with a 128-bit or 256-bit key length.

AES Galois Counter Mode (GCM) The AES algorithm using Galois Counter Mode and with a 16-byte integrity check value (ICV). Galois Counter Mode is a combined-mode algorithm that performs both encryption and authentication simultaneously. IP security for z/OS Communications Server supports AES_GCM with a 128-bit or 256-bit key length.

AES Galois Message Authentication Code (GMAC) The AES algorithm using Galois Counter Mode to encode authentication data in either AH or ESP headers. AES_GMAC functions as a combined-mode algorithm; however, it provides authentication without encryption. IP security for z/OS Communications Server supports AES_GMAC with a 128-bit or 256-bit key length.

AES Extended Cipher Block Chaining (XCBC) The AES algorithm using the XCBC mode to encode authentication data in either AH or ESP headers, with 128-bit keys and hash truncation to 96 bits.

Hashed message authentication code (HMAC) A one-way hash function that combines the contents of a message and a secret key to produce a hash value; used for authentication.

HMAC_MD5 HMAC using the MD5 algorithm.

HMAC_SHA1 HMAC using a SHA1 algorithm that encodes authentication data in AH or ESP headers, using a 160-bit hash value and 96-bit integrity check value (ICV).

HMAC_SHA2 HMAC using a SHA2 algorithm that encodes authentication data in AH or ESP headers and that is qualified by the length of key and hash truncation. The algorithm can have 256-bit keys and hash truncation to 128 bits, 384-bit keys and hash truncation to 192 bits, or 512-bit keys and hash truncation to 256 bits.

Secure hash algorithm 1 (SHA1) A MAC algorithm similar to MD5, but more secure. This algorithm produces a 160-bit hash value.

Secure hash algorithm 2 (SHA2) A MAC algorithm similar to SHA1, but more secure. This algorithm produces a 256-bit, 384-bit or 512-bit hash value.

Asymmetric Keys



- Asymmetric Keys are not identical but are mathematically related.
 - Data encrypted with one of the keys in the key pair can only be unencrypted with the other key.
 - Cannot be unencrypted with the first key!
 - But knowing one key, in no way helps guess what the second key is.
- Public/Private Key Pair
 - One of keys is kept by the owner and never shared. – The Private Key
 - The other key is given to everyone. – The Public Key
- Authentication
 - Host A inputs a piece of data and their private key into an encryption algorithm and sends the encrypted data to Host B.
 - Host B inputs the encrypted data and the public key into the same encryption algorithm and obtains the original data.
 - Everyone can read the message because everyone has the public key.
 - The message had to come from Host A because they are the only one with the private key.
 - Host A has been authenticated as the originator of the message. - Authentication
 - Host A cannot repudiate that they created the message. – Non-Repudiation
- Privacy
 - Host B inputs a piece of data and Host A's public key into an encryption algorithm and sends the encrypted data to Host A.
 - Host A inputs the encrypted data and their private key into the same encryption algorithm and obtains the original data.
 - The message was kept secret when it was sent over the network. Only Host A could read the message. – Data Privacy
- Public/Private Asymmetric Cryptography requires a starting point.
 - Host B has to trust that the public key they have is associated with Host A's private key.
 - How is this trust established?

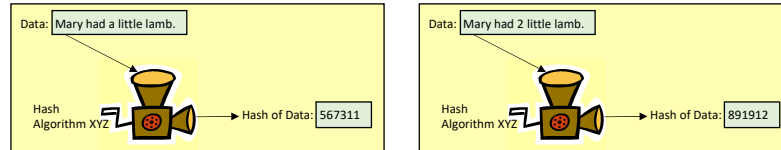
006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 8

Asymmetric key usage has a higher overhead than symmetric key usage.
 Public/Private key cryptography is commonly referred to as Public-key Cryptography.
 Encryption Algorithm is mathematical instructions

Hash Algorithm



- Encryption
 - An encrypted message is the same length as the original message.
- Hash
 - Hash is a one direction operation. A hash value can not be changed back into the original message.
 - A hash is a fix length output value (smaller than the original message) that is not unique to the input but rare enough to be an added verification that the data has not been changed in transit.
 - Data Integrity Verification
- Encryption and Hash Usage Together
 - Host A takes the data message and creates a Hash value from it.
 - Host A appends the Hash value to the data message and then uses the Session key to encrypt the whole thing.
 - Host A sends the encrypted message to Host B.
 - Host B unencrypts the data using the Session key.
 - Host B takes the data message and creates a Hash value from it, using the same algorithm.
 - Host B compares the created Hash value to the one sent with the data message. If the two values are the same, then the message has not been changed in transit.
- Digital Signature
 - Sometimes it doesn't matter if the message is viewed by others. Sometimes it is only important to prove that the message hasn't been changed and to prove it came from the person who claims it came from.
 - Host A takes the data message and creates a Hash value from it.
 - Host A encrypts the Hash value using their Private key.
 - Host A appends the encrypted Hash value to the unencrypted data message.
 - Host A sends the unencrypted data message with the encrypted Hash value to Host B.
 - Host B uses Host A's Public key to unencrypt the Hash value, thereby proving the message came from Host A.
 - Host B takes the data message and creates a Hash value from it, using the same algorithm.
 - Host B compares the created Hash value to the one sent. If the two values are the same, then the message has not been changed in transit.

006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 9

When the hash value is used for providing data integrity for the data phase of a flow, the hash value is protected by the session (symmetric) key.

In the negotiation phase of a secure connection establishment, the hash is generated against the session key to prove that it has not been altered in flight. When the hash value is used in certificates, it also provides data integrity for a subset of the certificate and becomes the signature in the certificate.

A hash is a way of representing the original content in a compressed and unique form. Since it is one-way, it cannot be reversed to reveal the original value or form. But if the original value or form is hashed again, the new hash should match the hash that was received, thus proving that the original data was not changed.

Common hash algorithms include:

MD5 (no longer considered secure or irreversible)

SHA-1

SHA-256

SHA-512

SHA-2/210

x.509 Digital Certificate

- X.509 is a standard format.
- Public/Private Key Pair Creation
 - An X.509 Certificate is created and issued along with the key pair.
 - Certificate establishes the credentials of the entity.
 - The public key is always attached to the certificate.
- Signed
 - Each certificate is signed by a Certificate Authority (CA) private key.
- Certificate Fields (some of them)
 - Serial Number – unique on the creation system
 - Algorithm – the algorithm that was used to sign the certificate
 - Issuer – CA that created the certificate (or CA certificate chain)
 - Does the partner need the immediate CA certificate or the full certificate chain for authentication?
 - Usually just the immediate CA certificate but it depends upon the application.
 - Valid Dates – Date range that certificate is valid
 - Subject Distinguished Name – the full name (ie. International Business Machines Incorporated)
 - Extensions – Optional fields that may be used for authentication
 - Altname (nickname) (ie. IBM)
 - IP Address
 - Domain Name
 - Fully Qualified Host Name



006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 10

The ITU Telecommunication Standardization Sector (ITU-T) is one of the three sectors of the International Telecommunication Union (ITU). It coordinates standards for telecommunications.

In cryptography X.509 is an ITU-T standard for a public key infrastructure (PKI) and Privilege Management Infrastructure (PMI).

X.509 specifies, amongst other things, standard formats for public key certificates, certificate revocation lists, attribute-certificates, and a certification path validation algorithm.

Extensions can be easier to match for IPsec authentication purposes rather than the full distinguished name.

Certificate Hierarchy

CA Certificate and Key Pair



Server Certificate and Key Pair



- Certificate Authority (CA) creates a CA certificate and associates it with a public/private key pair.
- CA signs the CA certificate with the private key.
 - Self-signed certificate.
- CA creates a certificate for an end entity and associates it with a public/private key pair.
 - End entity certificate is often referred to as a server certificate even though it could be used by a server or client.
- CA signs the Server certificate with the CA private key.
- As listed above there may be a single CA certificate, or the first CA certificate could have been used to create a second CA certificate that could have been used to create a server certificate, etc.
 - The hierarchy of the certificates is also referred to as a certificate chain.

006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 11

A certificate chain can contain a single CA certificate or multiple CA certificates.

Asymmetric Keys Starting Point

- Well Known Certificate Authority (CA)
 - VeriSign
 - Thawte
 - etc.
- Starting Point - Well Known CA certificates are preloaded:
 - Internet Browsers:
 - Internet Explorer
 - Firefox
 - etc.
 - TN3270 Clients:
 - Pcomm
 - Host On-Demand
 - Rocket BlueZone
 - etc.
 - z/OS RACF
 - etc.

006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 12

Most platforms and applications have preloaded well known CA certificates all marked as trusted.

z/OS RACF is preloaded with all the well known CA certificates but they are all marked as "not trusted". For usage they must be changed to "trusted". This is just an additional security feature of the platform.

Encryption Algorithm

- A set of mathematical instructions (a procedure) for encoding data to make the data unrecognizable.
- The data encryption algorithms themselves are publicly known. Therefore, the algorithms by themselves are not secure.
- But the algorithms use "keys" during the cryptographic execution and those keys do not all have to be publicly known.
- The keys and the algorithm together are what make the encrypted output a secret.
- To decode or "decrypt" the encoded data, you need to know the key or keys that were used to encrypt the data to begin with.

Symmetric Keys -- the identical key is used on both sides of a secured connection or relationship

- The symmetric keys are kept secret or kept PRIVATE

Asymmetric Keys -- non-identical keys are used on each end of a secured connection or relationship

- One key is the PRIVATE or SECRET KEY and one key is the PUBLIC KEY.
- Each end of the connection generates its own PUBLIC and PRIVATE KEY.
- Diffie-Hellman Groups:
 - Group 1: Uses a 768-bit Prime Modulus in the DH calculation
 - Group 2: Uses a 1024-bit Prime Modulus in the DH calculation (more secure but also more CPU-intensive than Group 1)
 - Group 5: Uses a 1536-bit Prime Modulus in the DH calculation (more secure but also more CPU-intensive than Group 1 or Group 2)
 - Group 14: Uses a 2048-bit Prime Modulus in the DH calculation (more secure but also more CPU-intensive than Group 1, Group 2, or Group 5)

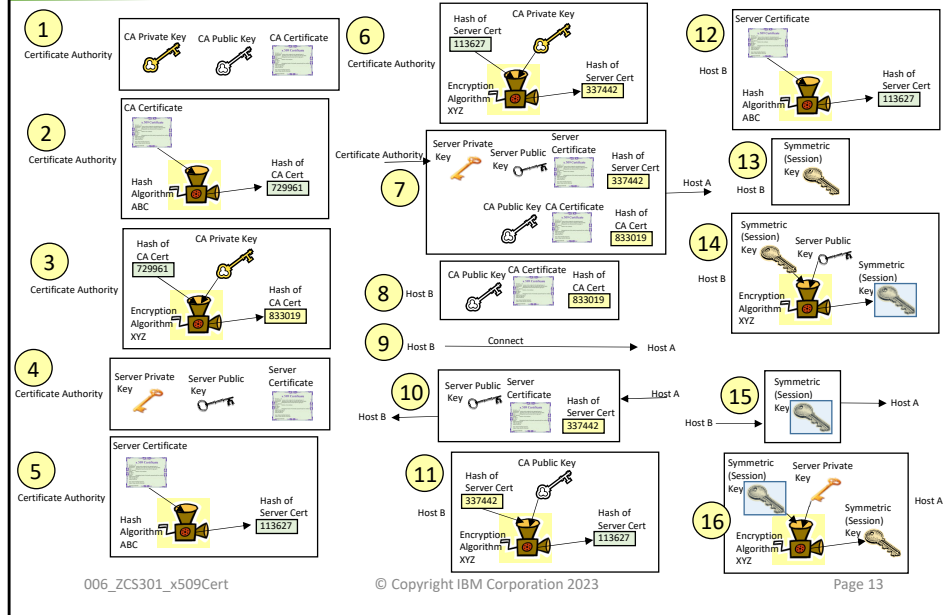
Keys are Essential to

- Authentication -- if both sides in a connection or relationship possess identical or related keys, then this can be used as proof of the identity of the partner.
- Non-Repudiation -- if only one partner (M) possesses a specific key and if it can be proven that that key was used to encrypt data, then the data must be from that partner (M).

Hashes are Essential to

- Data Integrity Verification -- the hash is a mathematical expression of the data; if the data has not changed, that mathematical expression is the same at the origin as at the destination.
- Data Privacy -- the hash cannot be reversed; thus, if someone captures the hash, they cannot reverse it to obtain the actual data. Thus, the data is kept private.
- Creation of Digital Signatures: a digital signature is an encrypted HASH value.

Asymmetric Keys and Certificate Usage



Steps of Asymmetric Public/Private Key Cryptography

1. CA (Well Known or Not) creates a CA private key, matching public key, and certificate.
2. CA creates Hash of CA certificate.
3. CA signs (encrypts) the CA certificate Hash with the CA private key.
4. CA creates a Server private key, matching public key, and certificate. (Not showing is server request.)
5. CA creates a Hash of the Server certificate.
6. CA signs the Server certificate Hash with the CA private key.
7. CA sends the Server private key, matching public key, Server certificate, signed Server certificate Hash, CA public key, CA certificate, and signed CA certificate Hash to Host A.
8. Host B has CA public key, CA certificate, and signed CA certificate Hash preloaded.
9. Host B starts connection to Host A.
10. Host A sends the Server public key, Server certificate, and signed Server certificate Hash to Host B.
11. Host B unencrypts the Server certificate Hash using the CA public key.
12. Host B creates Hash of the Server certificate and checks that it matches the one sent, authenticating Host A.
13. Host B creates Symmetric (Session) key.
14. Host B encrypts the Session key with the Server public key.
15. Host B sends the encrypted Session key to Host A.
16. Host A uses the Server private key to unencrypt the Session key.

At the end of the process described on this page, both sides have the same Symmetric key. The Symmetric key can be used to send data back and forth with less overhead than asymmetric encryption.

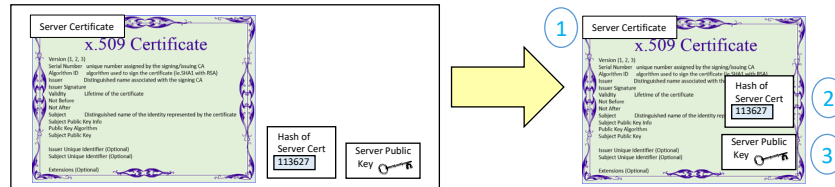
If both sides are authenticated, then server requests client certificate for authentication in step 10.

A key is a specific length. The symmetric key length is the same as the associated asymmetric key length.

The encrypted data length is the same size as the unencrypted data length.

The certificate request can include the data required, a generated certificate, or some other information. The certificate request process is outside the scope of this class.

Certificate



- From this point forward in this presentation “Certificate” will refer to:
 1. The text document that contains the end entity’s information (referred to so far as the certificate)
 2. The Hash of the certificate signed by the CA’s private key
 3. The end entity’s public key. The three are bound together.

006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 14

The Session symmetric key can be derived from a pre-master secret (also known as a shared secret). In this case only the pre-master secret needs to be sent to the partner. The partner uses the pre-master secret to generate the same Session symmetric key on their side.

RFC2246 for SSL/TLS describes how the pre-master secret is conveyed:

- Through direct transmission of the RSA-encrypted secret, or by the transmission of Diffie-Hellman parameters which will allow each side to agree upon the same premaster secret.

The client uses a series of cryptographic operations to convert the pre-master secret into a master secret, from which all key material required for encryption and message authentication is derived. Then the client sends a "change cipher spec" message to make the server switch to the newly negotiated cipher suite. The next message sent by the client (the "finished" message) is the first message encrypted with this cipher method and keys.

The server responds with a "change cipher spec" and a "finished" message of its own.

The SSL handshake ends, and encrypted application data can be sent.

An excellent summarization of the SSL process is at:

<http://www.pierobon.org/ssl/ch2/detail.htm>

Another excellent overview of the SSL processes is at:

http://pic.dhe.ibm.com/infocenter/wmqv7/v7r1/index.jsp?topic=%2Fcom.ibm.mq.doc%2Fsy10520_.htm

Digital Key Storage



006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 15

Different Types of Digital Keys

- Key Functions:
 - Authentication and Non-Repudiation
 - Encryption
 - Data Integrity Verification through Hashing
- Key Types:
 - Symmetric and Asymmetric
- Common Networking Protocols that Use Keys
 - TLS
 - IPSec (Virtual Private Networks or "VPN")
 - Secured Shell ("SSH")
- Where Can Keys Be Stored? How Are They Distributed?
 - In a Configuration File (Definition File) - Password
 - OSPF in OMPRoute (Dynamic Routing Protocol)
 - OSPF uses a "password" that is used to build an MD5 Hashing value -- not really a key, but used for authentication of partner.
 - Manual Tunnels for VPNs (Symmetric Keys)
 - Dynamic Tunnel with Pre-Shared Keys (Symmetric Keys)
 - In a Generated Key File and "known hosts file"
 - SSH (Uses Asymmetric, Public/Private Keys to generate Session Key)
 - Bound to an x.509 Certificate: Public Asymmetric Key
 - With x.509 Certificates the implementation is called Public Key Infrastructure (PKI)
 - In a Key Ring or Key Database: Private and Public Asymmetric Key
 - In Cryptographic Hardware (Master Key)
 - Not recommended to store Private key in hardware, but rather in PKCS (Public-Key Cryptography Standard) dataset for ICSF.

006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 16

ICSF refers to the Integrated Cryptographic Service Facility.

Each party, both client and server, has its own certificate, a matching private key, and a list of trusted certificate-authority certificates. When the client needs to authenticate itself to the server to be able to perform a transaction, both the server and client need to verify one another. The protocol for a secure handshake for mutual verification begins with the parties exchanging certificates. Each party then separately validates the other's certificate to make sure that its signature is valid, that the subject name in the certificate is correct, and that the certificate originated from a trusted certificate authority. If successful, each party must prove to the other that it owns the private key that matches its public key certificate. This step establishes proof of possession and can be accomplished by having each party sign a known unique value, such as a hash of the message traffic between the two parties. If each signature can be validated using the associated public key, the proofs are successful. The final step in this handshake is for one of the parties to generate a random symmetric key, encrypt it using the other party's public key, and send it to the other party. This random symmetric key may then be used to encrypt the data for the remainder of the session. Once the secure handshake is complete, secure transactions can be safely handled in the z/OS environment between this client and server.

Key Ring

- Certificates and Keys may be stored in Key Repositories
 - Key Ring, Key Database, on a card, on a chip, or in an area of storage on a computer platform
 - IPsec only supports RACF Key Ring repository
 - AT-TLS supports both RACF Key Ring and Unix gskkyman repository
 - Unix gskkyman is outside the scope of this course.
- Key rings are defined in RACF
 - Provides access to some of the certificates and associated private keys loaded in the RACF database.
 - Certificate must be "marked" (defined) as TRUSTED.
 - Certificates with date ranges outside of the current date are created as UNTRUSTED.
 - In order to use a certificate the current date must be in the date range defined in the certificate.
 - RACF is the preferred certificate storage for z/OS Communications Server applications
 - NOTE: IKE with RSA Signature Mode requires a SAF Keyring; other key repositories are not supported.
- Keep Track of certificate Expiration Dates
 - RACF ICETOOL at www.racf.co.uk
 - Unload the RACF database and then run the icetool to produce a report on certificate expiration dates.
 - Health Check RACF_CERTIFICATE_EXPIRATION
 - Identifies expired and near expired certificates.
 - PKI Services
 - See next page.

006_ZCS301_x509Cert © Copyright IBM Corporation 2023 Page 17

A key ring is a collection of certificates that identify a networking trust relationship (also called a trust policy).

An x.509 certificate contains information about the entity that uses it. It establishes the credentials of the entity that it represents. It also contains a public key that can be used to assist with encryption and signature validations.

When the certificate is generated, a private key is also produced, but this private key is not stored inside the x.509 certificate.

- It is stored on a keyring or key database together with the digital certificate.
- The certificate must be TRUSTED.
- A CA uses its private key to sign a user's public key.

The X.509 digital certificate is a data structure that contains, at minimum, the following fields:

- The distinguished name of the owner of the public key, also called the subject's name
- The distinguished name of the issuer of the certificate, also called the issuer's name
- The public key itself
- The time period during which the certificate is valid, also called the validity period
- The certificate's serial number as designated by the issuer
- The issuer's digital signature.
 - The digital signature is at the end of the message (or, in this case, the certificate). To create the signature, the certificate fields (except for the signature) are hashed and then encrypted using the private key. The certificate contains the signing algorithm of the entity that signed the certificate. For example RSA with SHA1.
 - Digital signature: This is generated using the signing authority's (CA's) private key. To verify the digital signature, we need the signing authority's public key, which is found in the certificate of the signing authority.

In addition to these required fields, an X.509 certificate may contain one or more extensions that hold information about how the key is to be used (a KeyUsage extension) or how the certificate authority conducts its business (a CertificatePolicies extension).

One of the important extensions for IPSecurity is the Alternate Name extension.

Some protocols, like, IPsec, can require more than one way to identify a certificate.

The Alternate Name extension (also called "ALTNAME") might be:

- email address
- IP address
- Fully Qualified Domain Name
- User@domain

Only the Public key resides in the x.509 certificate itself. The private key resides in the repository of the end-entity that is represented with the certificate.

The private key is used for signing certificates. It is also used to decrypt data that has been encrypted with the corresponding public key. It is also used to encrypt data that can be decrypted only by the corresponding public key.

Any certificate in the "chain of trust" must be marked as TRUSTED in the repository where the key ring or key database resides.

z/OS RACF contains a list of Well-known Certificate Authorities, but none of them is marked TRUSTED until you change that setting. Appendix C of the Security Server RACF Security Administrator's Guide contains listings of RACF supplied certificates.

Here is an example of the RACF raddcert command that can mark a well-known CA as Trusted:

```
RACDCERT CERTAUTH ALTER(LABEL('Verisign Class 3 Primary CA')) TRUST
```

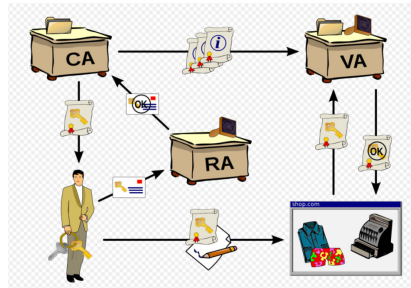
z/OS Cryptographic Services PKI Services

- Public key infrastructure (PKI) provides applications with a framework for performing the following types of security-related activities:

- Authenticate all parties that engage in electronic transactions
- Authorize access to sensitive systems and repositories
- Verify the author of each message through its digital signature
- Encrypt the content of all communications

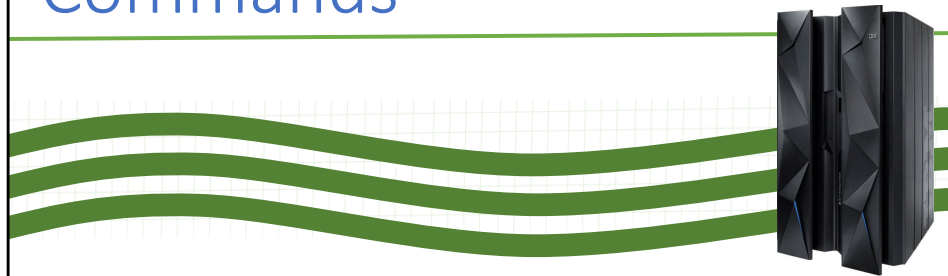
- z/OS Cryptographic Services PKI Services

- Allows you to use z/OS to establish a PKI infrastructure and serve as a certificate authority
- Allows you to use z/OS to issue and administer digital certificates
- PKI Services application may request and obtain certificates through their Web browsers
- Authorized PKI administrators approve, modify, or reject these requests through their Web browsers
- Allows automatic approval for certificate requests from certain users and, to provide additional authentication, add host IDs, such as RACF user IDs, to certificates you issue for certain users
- Allows creation of certificates for browsers, servers, and other purposes
- Allows automatic certificate expiration notification
- Allows certificate renewal via web browser
- Supports certificate revocation lists
- Supports the delivery of certificates through the Secure Sockets Layer (SSL) for use with applications that are accessed from a Web browser or Web server
- Supports the delivery of certificates that support the Internet Protocol Security standard (IPSEC) for use with secure VPN applications or IPSEC-enabled devices



From Wikipedia <http://en.wikipedia.org>

Certificate Usage and RACF Commands

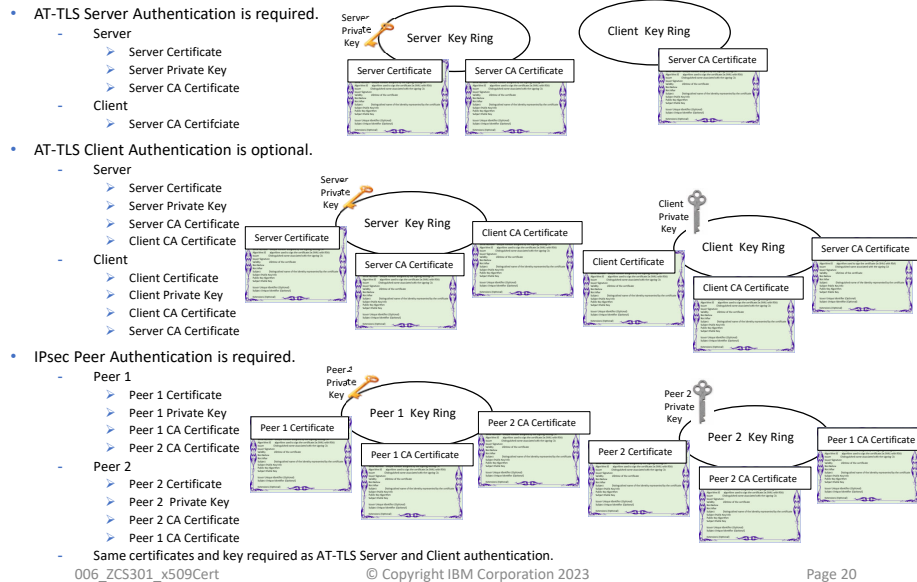


006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 19

Key Ring Contents



The RACDCERT command is used to store and maintain digital certificate information in RACF, and should be used for all maintenance of certificate profiles and related USER profile fields.

The RACF RACDCERT Command can be used to generate a Certificate Request that can be sent to a Certificate Authority that will produce the x.509 certificate and send it back. RACDCERT is used to install and maintain digital certificates, key rings, and digital certificate mappings in RACF. RACDCERT should be used for all maintenance of the DIGTCERT, DIGTRING, and DIGTNMAP class profiles.

- It also produces a key ring.
- It can generate keys.
- It can produce a self-signed certificate.
- Imports/Exports certificates (with and without private keys)
- Can renew and revoke a certificate.

The RACDCERT command is a RACF TSO command used to:

- List information about the certificates for a specified RACF-defined user ID, or your own user ID.
- Add a certificate definition and associate it with a specified RACF-defined user ID, or your own user ID, and set the TRUST flag.
- Alter the TRUST flag or the label name for a definition.
- Delete a definition.
- List a certificate contained in a data set and determine if it is associated with a RACF-defined user ID.
- Add or remove a certificate from a key ring.
- Create, delete, or list a key ring. v Generate a public/private key pair and certificate.
- Write a certificate to a data set. v Create a certificate request. v Create, alter, delete, or list a user ID mapping.
- Add, delete, or list a z/OS PKCS #11 token.
- Bind a certificate to a z/OS PKCS #11 token.
- Remove (unbind) a certificate from a z/OS PKCS #11 token.
- vmport a certificate (with its private key, if present) from a z/OS PKCS #11 token and add it to RACF.

Alternatively, the UNIX System Services Command "gskkyman" can be used to generate the Certificate Request.

- It also produces a keyring. It can generate keys. It can produce a self-signed certificate. Imports/Exports certificates (with and without private keys). Can renew and revoke a certificate.

RACF Key Rings

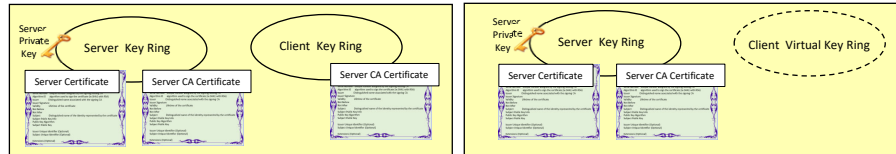
- RACF key rings are protected by resource profiles.
- Users rings need read access to IRR.DIGTCERT.LISTRING to be able to read the contents of their key ring.

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

OTHER PLATFORMS: Other utilities on different platforms are commonly used to create certificates as well. You may run into a utility called "mkkf" or another one called "ikeyman."

Key Ring and Certificate Types



- RACF Key Ring
 - Regular Key Ring
 - Key Ring is owned by a single User ID.
 - Virtual Key Ring
 - Access to all CA Certificates
 - Certificate must be marked as trusted.
 - Useful when ONLY CA Certificates are needed.
 - Client side when Server Only authentication is being used.
- RACF Certificates
 - When a certificate is connected to a key ring the private key is also attached if it is loaded in RACF.
 - Individual User or Personal Certificate (Server Certificate)
 - Only one owner
 - End Entity must "own" the certificate to use it.
 - Site Certificate
 - Supports multiple Owners
 - CA Certificate
 - Well-Known or Local CA

006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 21

A sender's key repository must contain all certificates that represent the full chain of trust for that sender.

A recipient's key repository must contain only the ROOT certificate associated with the sender's full chain of trust.

This chart shows you that you can build a RACF Virtual Keyring if you are not presenting a client certificate for verification.

For applications using System TLS, such as z/OS FTP, or other middleware programs that read RACF key rings through the callable service, a virtual key ring can be specified in place of a real key ring, whenever a real key ring is expected.

USER CERTIFICATE:

A certificate that is associated with a RACF user ID and is used to authenticate the user's identity. The RACF user ID can represent a traditional user or be assigned to a server or started procedure.

SHARED SITE CERTIFICATE:

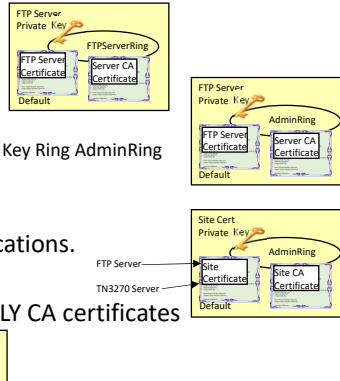
You can share a certificate and the certificate's private key among two or more servers (user IDs) when you add or generate the shared certificate and its private key as a SITE certificate, for example using the RACDCERT SITE GENCERT command. Sharing a certificate can save you the expense of purchasing a new certificate for each server and avoids the overhead of exporting and importing certificate copies. Sharing a private key requires a high degree of authority for each server involved. The key ring containing the shared certificate must be protected and each server must be configured to access the shared key ring and have sufficient access authority to read it. In addition, each server must have CONTROL authority for the IRR.DIGTCERT.GENCERT resource. This resource controls the server's ability to retrieve private keys using the R_datalib callable service and is checked when you issue the RACDCERT GENCERT SIGNWITH command.

CERTIFICATE-AUTHORITY CERTIFICATE:

A certificate that is associated with a certificate authority and is used to verify signatures in other certificates. Also called a "Signing Certificate."

AT-TLS or IPsec Certificate Definition

- By default the DEFAULT certificate on the Key Ring is used.
 - To use a certificate that is not the default, the certificate Label Name must be specified.
 - The certificate Label Name is defined when the certificate is added to the certificate repository. Label Name is unique on the repository.
- Individual User or Personal Certificate (Server Certificate)
 - Certificate Must be Owned by User
 - Key Ring Owned by User
 - KEYRING FTPServerRing
 - Key Ring Not Owned by User
 - KEYRING ADMIN/AdminRing
 - Where userid ADMIN is the owner of Key Ring AdminRing
- Site Certificate
 - KEYRING ADMIN/AdminRing
 - Certificate shared by multiple applications.
- CA Certificate
 - Virtual Key Ring provides access ONLY CA certificates
 - KEYRING *AUTH*/*



006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 22

An individual Server Certificate MUST BE OWNED BY THE PROCESS that invokes it.

If you own the keyring, you just have to identify the simple name of the keyring to open it. If someone else owns the keyring, then you must identify the keyring by prefacing the simple name with the name of the owner.

Each RACF user ID is associated with a virtual key ring. The most common type is the CERTAUTH virtual key ring, which is used when an application validates the certificates of others but has no need for its own certificate and private key.

For applications using System SSL, such as z/OS FTP, or other middleware programs that read RACF key rings through the R_datalib callable service, a virtual key ring can be specified in place of a real key ring, whenever a real key ring is expected. To include virtual key rings, the application user specifies an asterisk (*) for the key ring name along with the ring owner's user ID using the form ring-owner/*.

If using a SAF repository other than RACF, note that ACF2 appends a suffix of ".KEYRING" to the keyring name. Therefore, a keyring that you think is called "MYKEYRING" must be referenced as "MYKEYRING.KEYRING" for an ACF2 repository.

Multiple Rules or Multiple Key Rings

- User ID must own certificate to use it.
 - Server or Client
- When Client Authentication is being used, is a separate key ring and a separate rule required for each client?
- User ID does not have to own the key ring.
 - Multiple Rules, one for each User ID
 - One rule specifies keyring FTPCL/FTPCLRing and label User21Cert
 - One rule specifies keyring FTPCL/FTPCLRing and label User31Cert
 - User ID FTPCL owns FTPCLRing keyring
 - When client User ID USER21 matches traffic then User21Cert is used.
 - When client User ID USER31 matches traffic then User31Cert is used.
 - Are you able to differentiate the traffic/rules?
 - Multiple Key Rings, all the same Key Ring name
 - keyring FTPClientRing
 - When User ID USER21 traffic matches the rule keyring FTPClientRing owned by User ID USER21 is used.
 - When User ID USER31 traffic matches the rule keyring FTPClientRing owned by User ID USER31 is used.

Certificate Packages

- Single Binary Certificate (Format is "CERTDER")
 - Coded as DER (Distinguished Encoding Rules), platform-independent format
 - Use: Used mostly for Certificate Requests, which are always DER-encoded and then base64-encoded, like base64-encoded certificates.
- PKCS#7 (binary package) (Format is "PKCS7DER")
 - One or more Certificates packaged together but not signed or encrypted
 - End Entity Certificate
 - Certificate Chain of Trust
 - Use: When the CA wants to deliver multiple certificates to a destination
- PKCS#12 (binary package) (Format is "PKCS12DER")
 - One or more Certificates packaged together, password-encrypted
 - End Entity Certificate
 - Certificate Chain of Trust (includes CA certificates if found)
 - Can Contain the Private Key if generated by the Certificate Authority
 - Use: When the CA wants to ship a package confidentially that contains the private key.
 - Use: To migrate certificates and keys from one platform to another.
- Base64-encoded Certificates (Format is "CERTB64" if no private key; PKCS12B64 if with private key and then it requires password-encryption)
 - Ascii, Text
 - Use: When making a Certificate Request in an email or if Certificate Management Facility on another platform requires ASCII format.

006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 24

PKCS = Public Key Cryptographic Standards

Some applications accept only certain formats in the packaging of certificates. Before you export a certificate, be sure that you export it in a format acceptable to the application program.

Personal Communications (PCOMM) accepts CERTB64, DER, and P12.

Microsoft Management Console accepts DER and P12.

If you want to create a backup copy of an existing certificate (and its non-ICSF private key) on a different system, use the RACDCERT EXPORT command to create a PKCS #12 format data set on the system where the certificate resides, and send the data set to the other system where you can use it as input with the RACDCERT ADD command to recreate the same certificate.

Restriction: If the private key is stored in ICSF (key type ICSF or PCICC), a PKCS #12 data set cannot be created. (See z/OS Security Server RACF Command Language Reference for details about using the RACDCERT EXPORT command.)

If you want to create a backup copy of an existing certificate (and its non-ICSF private key) on a different system, use the RACDCERT EXPORT command to create a PKCS #12 format data set on the system where the certificate resides, and send the data set to the other system where you can use it as input with the RACDCERT ADD command to recreate the same certificate.

Restriction: If the private key is stored in ICSF (key type ICSF or PCICC), a PKCS #12 data set cannot be created. (See z/OS Security Server RACF Command Language Reference for details about using the RACDCERT EXPORT command.)

You can use any depth CA chain. The key to getting this to work is what is sent in the certificate request payload. A CA chain Root->Sub1->Sub2->Sub3-Sub4-Sub5...SubN-EE will work as long as a Certificate Request sent identifies SubN or is empty.

According to Wikipedia as of 2011: Distinguished Encoding Rules (DER), is a message transfer syntax specified by the ITU in X.690. The Distinguished Encoding Rules of ASN.1 is an International Standard drawn from the constraints placed on basic encoding rules (BER) encodings by X.509. DER encodings are valid BER encodings. DER is the same thing as BER with all but one sender's options removed. DER is a subset of Basic Encoding Rules (BER) providing for exactly one way to encode an ASN.1 value. DER is intended for situations when a unique encoding is needed, such as in cryptography and ensures that a data structure that needs to be digitally signed produces a unique serialized representation. DER can be seen as a canonical form of BER (see also Canonical Encoding Rules).

RACDCERT Command

- Define RACDCERT as an authorized TSO command in IKJTSOxx
- Define IRR.DIGTCERT.function resources using RDEFINE RACF command
 - function = ADD, ADDRING, ALTER, ALTPMAP, BIND, CONNECT, DELETE, DELMAP, DELRING, EXPORT, EXORTKEY, GENCERT, GENREQ, LIST, LISTMAP, LISTRING, MAP, REKEY, REMOVE, ROLLOVER
- To issue RACDCERT, user must have one of the following authorities
 - SPECIAL attribute
 - Sufficient authority to Facility class IRR.DIGTCERT.function resources

- Permit users to IRR.DIGTCERT.function

Access Permission	Description
READ	control certificates for this user only
UPDATE	control certificates for other users too
CONTROL	control special certificates like CERTAUTH (Certificate Authority) certificates

- A SETROPTS command "refreshes" the controls for the certificate functions
- User executing this command requires the SPECIAL attribute

006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 25

Some RACDCERT Commands

- DIGTCERT ADD - create a certificate using a data set
- DIGTCERT ADDRING - create a key ring
- DIGTCERT CONNECT - add certificate to key ring
- DIGTCERT DELETE – delete certificate
- DIGTCERT DELRING – delete key ring
- DIGTCERT EXPORT – write certificate to data set
- DIGTCERT GENCERT – create a certificate and optionally a public/private key pair
- DIGTCERT GENREQ – create a certificate request
- DIGTCERT LIST – list certificates
- DIGTCERT LISTRING – list key rings
- DIGTCERT REKEY - replicate (rekey) a digital certificate with a new public/private key pair
- DIGTCERT REMOVE – remove certificate from key ring
- DIGTCERT ROLLOVER - supersede one certificate (the source certificate) with another certificate (the target certificate)

006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 26

Documented in Security Server RACF Command Language Reference
REKEY and ROLLOVER are usually issued in that order to replace a certificate.

RACF DIGTCERT Protection

• ADD and GENCERT	SIGNWITH	One's own certificate	Someone else's certificate	SITE or CERTAUTH certificate
	SIGNWITH one's own certificate	READ authority to IRR.DIGTCERT.ADD and READ authority to IRR.DIGTCERT.GENCERT	UPDATE authority to IRR.DIGTCERT.ADD and READ authority to IRR.DIGTCERT.GENCERT	CONTROL authority to IRR.DIGTCERT.ADD and READ authority to IRR.DIGTCERT.GENCERT
	SIGNWITH a SITE or CERTAUTH certificate	READ authority to IRR.DIGTCERT.ADD and CONTROL authority to IRR.DIGTCERT.GENCERT	UPDATE authority to IRR.DIGTCERT.ADD and CONTROL authority to IRR.DIGTCERT.GENCERT	CONTROL authority to IRR.DIGTCERT.ADD and CONTROL authority to IRR.DIGTCERT.GENCERT
	SIGNWITH not specified	READ authority to IRR.DIGTCERT.ADD and READ authority to IRR.DIGTCERT.GENCERT	UPDATE authority to IRR.DIGTCERT.ADD and UPDATE authority to IRR.DIGTCERT.GENCERT	CONTROL authority to IRR.DIGTCERT.ADD and CONTROL authority to IRR.DIGTCERT.GENCERT
• ADD and ADDRING	Function	READ	UPDATE	CONTROL
	ADD	Add a certificate to one's own user ID.	Add a certificate for someone else.	Add a CERTAUTH or SITE certificate.
	ADDRING	Create a key ring for one's own user ID.	Create a key ring for another user ID.	
	LIST	List one's own certificate.	List the certificate of someone else.	List CERTAUTH or SITE certificates.
• LIST and LISTRING	LISTRING	See one's own key ring.	See the key ring of someone else.	

006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 27

You need different types of authority to be able to execute the necessary commands. The authorities required (READ, UPDATE, CONTROL) are described in the RACF Command Language Reference.

You will also find several Facility Classes for Certificates and Keyrings. The authorities required (READ, UPDATE, CONTROL) to access these classes or functions are described in the RACF Command Language Reference.

Effective use of RACDCERT requires that its privileges be carefully controlled. However, end-users and application administrators should be allowed some flexibility in defining their security characteristics. These guidelines might prove useful.

- The ability to add certificate authorities and site certificates should be allowed to only a small set of trusted people.
- End users should be permitted to add, delete, and modify the contents of their own key rings and add, delete, and alter their own certificates.
- Help desk personnel should be allowed the ability to list certificates and rings.

Key rings are associated with specific RACF user IDs. A RACF user ID can have more than one key ring. Key rings are managed using the RACDCERT command, and are maintained in the general resource class called DIGTRING.

Authority to the IRR.DIGTCERT.function resource in the FACILITY class allows a user to issue the RACDCERT command. To issue the RACDCERT command, users must have one of the following authorities:

- SPECIAL attribute
- Sufficient authority to resource IRR.DIGTCERT.function in the FACILITY class.
 - READ access to IRR.DIGTCERT.function to issue the RACDCERT command for themselves.
 - UPDATE access to IRR.DIGTCERT.function to issue the RACDCERT command for others.
 - CONTROL access to IRR.DIGTCERT.function to issue the RACDCERT command for SITE and CERTAUTH certificates.

(This authority also has other uses.)

Authority required for the GENCERT function: The GENCERT keyword allows a certificate to be generated and signed. Effective controls on the user ID that is being associated with the certificate and what certificate is being used to sign the generated certificate are essential.

RACF Granularity

- Access to all Key Rings
 - PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ACCESS(UPDATE) ID(USER99)
- Access to a particular Key Ring
 - PERMIT USER99.USER99RING.LST CLASS(RDATALIB) ACCESS(READ) ID(USER99)
- Access to the Private Key
 - PERMIT USER99.USER99RING.LST CLASS(RDATALIB) ACCESS(UPDATE) ID(USER99)

Two z/OS Security Services manuals are critical to RACF functions and syntax:
z/OS Security Server RACF Security Administrator's Guide
z/OS Security Server RACF Command Language Reference

Access is based on a profile of a specific key ring in a class called RDATALIB

The class RDATALIB must be RACLISTed

A resource with the format <ringOwner>.<ringName>.LST is used to provide access control to a specific key ring on R_datalib READ functions.

This support also allows the retrieval of another person's private key

Instead of giving access to all the rings, just give access to that particular ring.

Certificate Creation



006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 29

Generate Self Signed CA Cert

```
//GBGCAS21 JOB MSGCLASS=X,NOTIFY=&SYSUID
//GBGCAS21 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RACDCERT CERTAUTH GENCERT
    SUBJECTSDN (O('GBG')
                CN('GBGCAS21.LABS.IBM.COM')
                C('US'))
    ALTNAME (IP(192.168.20.102)
             DOMAIN('GBG.LABS.IBM.COM')
             EMAIL('ZOS@GBG.LABS.IBM.COM'))
    NOTBEFORE (DATE (2009-11-27))
    NOTAFTER (DATE (2010-06-27))
    KEYUSAGE (CERTSIGN)
    SIZE (1024)
    WITHLABEL ('GBGCAS21 LABS Server CA')
```

Generate Server Certificate

```
//GBGSRV21 JOB MSGCLASS=X,NOTIFY=&SYSUID
//GBGSRV21 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RACDCERT ID(TN3270) GENCERT
    SUBJECTSDN (O('IBM')
               CN('TN3270.GBG.LABS.IBM.COM')
               C('US'))
    ALTNAME (EMAIL('TN3270@GBG.LABS.IBM.COM'))
    NOTBEFORE (DATE (2009-11-27))
    NOTAFTER (DATE (2010-06-09))
    WITHLABEL ('TN3270 on MVS2')
    SIZE (1024)
    SIGNWITH (CERTAUTH
              LABEL ('GBGCAS21 LABS Server CA'))
```

Export Certificate

```
//GBGX1221 JOB MSGCLASS=X,NOTIFY=&SYSUID  
//GBGX1221 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K  
//SYSTSPRT DD SYSOUT=*  
//SYSTSIN DD *  
RACDCERT ID(USER21) EXPORT(LABEL('USER21 on MVS2')) -  
FORMAT(PKCS12DER) DSN('USER.CS.TEAM21.P12') PASSWORD('USERLABS')
```

Create Certificate with Request

1. Create Self Signed certificate as a place holder.

```
//GBGUPE21 JOB MSGCLASS=X,NOTIFY=&SYSUID
//GBGUPE21 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RACDCERT ID(USER11) GENCERT -
          SUBJECTSDN (O('IBM') -
                     CN('USER11.GBG.LABS.IBM.COM') -
                     C('US')) -
          ALTNAME (EMAIL('USER11@GBG.LABS.IBM.COM')) -
          NOTBEFORE (DATE(2009-11-27)) -
          NOTAFTER (DATE(2010-06-09)) -
          WITHLABEL ('USER11 on MVS1') -
          SIZE (1024) -
          SIGNWITH (LABEL('USER11 on MVS1'))
```



Create Cert with Request (cont.)

2. Create Certificate Request.

```
//GBGURQ21 JOB MSGCLASS=X, NOTIFY=&SYSUID
//GBGURQ21 EXEC PGM=IKJEFT01, DYNAMNBR=30, REGION=4096K
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RACDCERT ID(USER11) GENREQ(LABEL('USER11 on MVS1')) -
DSN('USER11.CERT.REQ')
```

3. FTP certificate request to Certificate Authority.

4. Sign certificate request.

```
//GBGUGN21 JOB MSGCLASS=X, NOTIFY=&SYSUID
//GBGUGN21 EXEC PGM=IKJEFT01, DYNAMNBR=30, REGION=4096K
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RACDCERT ID(USER11) GENCERT('USER11.CERT.REQ') -
NOTBEFORE(DATE(2011-01-07)) -
NOTAFTER(DATE(2012-12-30)) -
SIGNWITH(CERTAUTH LABEL('MVS1 LABS Certificate Authority'))
```

Add Certificate

5. Signed certificate request.

```
//GBGUAD21 JOB MSGCLASS=X,NOTIFY=&SYSUID
//GBGUAD21 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RACDCERT ID(USER11) ADD('USER11.CERT.B64') -
```

Certificate with keys.

```
//GBGUAD21 JOB MSGCLASS=X,NOTIFY=&SYSUID
//GBGUAD21 EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RACDCERT ID(USER11) ADD('USER11.CERT.P12') -
TRUST WITHLABEL('USER11 on MVS1') -
PASSWORD('USERLABS')
```

When you receive a certificate with or without Public/Private key pair you can load it into RACF with the ADD command.

Rekey and Rollover Certificate

Rekey

```
//GBGURK21 JOB MSGCLASS=X, NOTIFY=&SYSUID
//GBGURK21 EXEC PGM=IKJEFT01, DYNAMNBR=30, REGION=4096K
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RACDCERT ID(USER11) REKEY(LABEL('USER11 on MVS1')) -
      SIZE(1024) -
      NOTBEFORE(DATE(2009-11-27)) -
      NOTAFTER(DATE(2010-06-09)) -
      WITHLABEL('USER11 on ANY MVS'))
```

Rollover

```
//GBGURO21 JOB MSGCLASS=X, NOTIFY=&SYSUID
//GBGURO21 EXEC PGM=IKJEFT01, DYNAMNBR=30, REGION=4096K
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RACDCERT ID(USER11) ROLLOVER(LABEL('USER11 on MVS1')) -
      NEWLABEL('USER11 on ANY MVS'))
```

Rekey will create a new Public/Private key pair for the certificate. Rollover to replace the certificate in RACF on all key rings.

Host Name Check

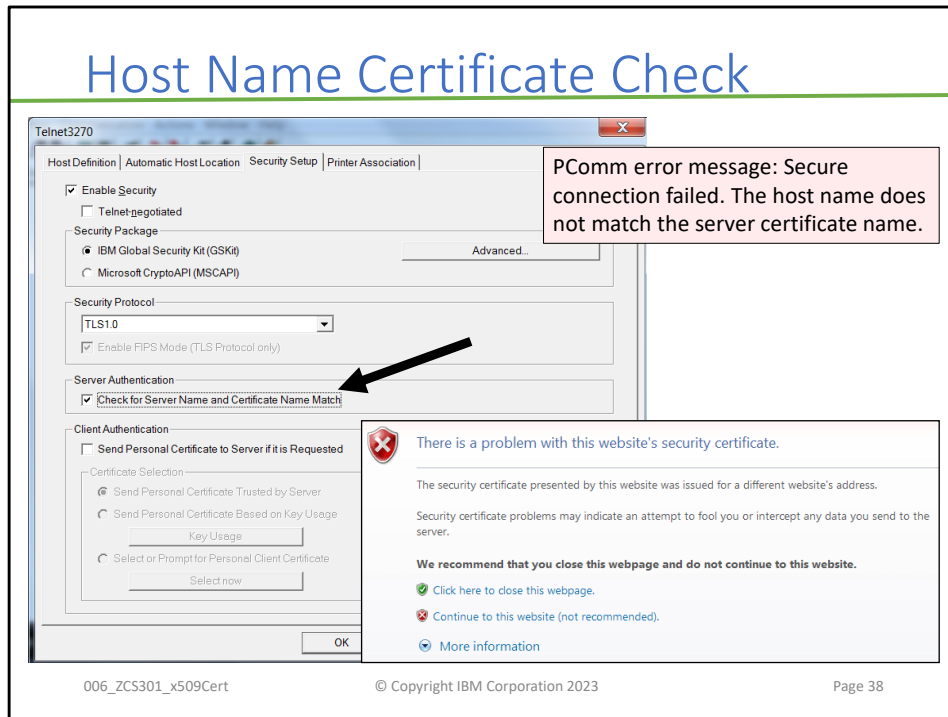


006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 37

Host Name Certificate Check



You can also have the session authenticate the server by matching the server name to the host or server certificate name. The server and certificate names must match exactly, or the session will not connect.

The PComm message appears in the log.

There is a like check that is done by browsers. The similar browser error message is show above.

End of Topic



006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 39

End of Topic



006_ZCS301_x509Cert

© Copyright IBM Corporation 2023

Page 40