Redpaper

Saheem Granados

OpenPGP Key Exchange and Migration

Archived

Introduction

Business exchange processes must define the mechanism for establishing trust among partners. Using cryptography as the foundation for trust, the OpenPGP support on z/OS® and its use of the Java™ Cryptography Extension (JCE) within IBM®'s Java 5 SDK for z/OS, allow for three options:

- Passphrase exchange
- OpenPGP certificate exchange
- X.509 certificate exchange

These three choices each involve the exchange of key material; a passphrase is equivalent to secret key material and must be treated as such. In addition to key exchange among trusted partners, migration of public/private key material from one format to another can be useful. Transfer of public/private key pairs can allow OpenPGP exchanges to leverage existing keys or keys migrated from other OpenPGP tools to the implementation provided with IBM Encryption Facility V1.2.

This paper describes the steps for exchanging OpenPGP certificates, X.509 certificates, and public/private key pairs, using the OpenPGP support and IBM Java 5 SDK tools: *keytool* and *hwkeytool*. Since the main focus is to describe the use of the tools on z/OS, passphrase exchange is not discussed.

Exchanging OpenPGP certificates

OpenPGP certificates encapsulate public key material. In addition to public key material, an OpenPGP certificate can contain additional data, including:

- 0 or more User ID packets – A User ID packet consists of a name, e-mail address, and comment string. The name is the only required information. Its purpose is to identify the entity that owns the certificate. Multiples are allowed since it is very common for a person to have multiple e-mail addresses.

- A primary public key – A primary public key must be of a type that can verify signatures, that is, RSA or DSA.
- 0 or more sub-public keys – A sub-public key can be of any type supported by Encryption Facility: RSA, DSA, ElGamal.
- 0 or more Signature sub-packets – Signature sub-packets encapsulate the results of a signature calculation on different elements of a certificate. They can exist in many different sections of the certificate and can be calculated by anyone's private key or even calculated using the private key pair of the public keys encapsulated within the certificate. Moreover, the sub-packet contains information that facilitates the search for the public key needed to verify the packet. This, in turn, allows for verification of the calculated signature they represent. Depending on its contents, a Signature sub-packet may be:
  - Asserting the validity of the certificate and any user ID packets included in the certificate. This is paramount for establishing trust.
  - Asserting the expiration date of the certificate.
  - Asserting the revocation of the certificate.

Exporting OpenPGP certificates

Encryption facility's OpenPGP support provides three commands that export public key material in OpenPGP certificates: `-eA`, `-eP`, and `-eK`.

The `–eA` command interactively creates a new OpenPGP certificate, using the public key material of the key in the Java keystore whose alias is specified as a command line parameter. (See the JAVA_KEY_STORE_NAME and JAVA_KEY_STORE_TYPE keywords in the configuration file for information on how to specify the location of the Java keystore file and its type). Figure 1 is an example of an –eA invocation. This invocation was executed from a UNIX® System Services (USS) shell.

Archived

```
/home/suimgwi/>java -jar CSDEncryptionFacility.jar -homedir . -a -o openpgp.asc
-eA saheemRSA
CSD0029I Exporting an OpenPGP certificate for saheemRSA...
CSD0026I At least one user ID is required for an OpenPGP certificate.
A user ID consists of three parts: a name, a comment(optional), and an email
address(optional).
CSD0020A Real Name:
redbooks 1
CSD0024A Comment:
redbooks1 example
CSD0022A Email address:
redbooks@us.ibm.com
CSD0025A You specified user ID: "redbooks 1 <redbooks1 example>
<redbooks@us.ibm.com>"
Change (N)ame, (C)omment, (E)mail, (X)Cancel or (O)kay to accept?
o
CSD0027A Add another user ID? (yes/no)
n
CSD0019A For how many days should this OpenPGP certificate be valid (0 for
always valid):[0]
0
CSD0032A Do you want to add the exported OpenPGP Certificate to your OpenPGP
key ring? (yes/no)
Yes
CSD0075A Certificate <4FA96D749D661CE2> already exists in the key ring. Would
you like to replace it? (yes/no)
Yes
CSD1347I 1 OpenPGP certificate(s) were exported successfully to openpgp.asc.
CSD0051I Command processing has completed successfully.
```

Figure 1 Exporting public key example

The file openpgp.asc was created and now can be exchanged with a business partner. This command also results in a copy of the newly created OpenPGP certificate being stored in the OpenPGP keyring file used by the OpenPGP support. (See the KEY_RING_FILENAME keyword in the configuration file for information on how to specify the location of the keyring file.)

The **–eP** command exports certificates that exist in the OpenPGP keyring file. It relies on a case-insensitive, substring match of a User ID packet. Figure 2 is an example of searching the keyring for the OpenPGP certificate created by the result of the **–eA** command.

```
/home/suimgwi/>java -jar CSDEncryptionFacility.jar -homedir . -a -o openpgp.asc
-eP redbooks
CSD0700A File <openpgp.asc> already exists.  Do you want to overwrite? (yes/no)
Yes
CSD0029I Exporting an OpenPGP certificate for "redbooks"...
CSD1347I 1 OpenPGP certificate(s) were exported successfully to openpgp.asc.
CSD0051I Command processing has completed successfully.
```

Figure 2 Exporting certificate example

The **–eK** command relies on the Key ID of the public key (primary or sub in the case of OpenPGP certificates) associated to a X.509 or OpenPGP certificate to locate the public key material to export. Key IDs are an 8 byte number, specified in hexadecimal (using 16

Archived

characters). The OpenPGP support first searches the OpenPGP keyring for a certificate that encapsulates the public key whose calculated key ID matches the specified command line option. If no key is found in the OpenPGP key ring, the keystore is searched. The **–pK**, **-pA**, and **–pP** display commands display key information including the Key ID. Figure 3 is an example of **–pA** that shows the Key ID of the key used for the first export example.

```
/home/suimgwi/>java -jar CSDEncryptionFacility.jar -homedir . -a -o openpgp.asc
-pA saheemRSA
CSD1352I Displaying certificate with alias: saheemRSA
dn: CN=Saheem Granados, OU=zos-security, O=IBM, L=Poughkeepsie, ST=NY, C=US
primary: RSA 1,024  key ID: 4FA96D749D661CE2  fingerprint:
15A586227F82F9F705BFAB414FA96D749D661CE2  created: 1/20/06 [expired: 4/20/06]
```

Figure 3 Display key information example

Figure 4 is an example of **–eK**.

```
/home/suimgwi/>java -jar CSDEncryptionFacility.jar -homedir . -a -o openpgp.asc
-eK 4FA96D749D661CE2
CSD0700A File <openpgp.asc> already exists.  Do you want to overwrite? (yes/no)
Yes
CSD0029I Exporting an OpenPGP certificate for 4FA96D749D661CE2...
CSD1347I 1 OpenPGP certificate(s) were exported successfully to openpgp.asc.
CSD0051I Command processing has completed successfully.
```

Figure 4 Example of -eK command to export certificate

The OpenPGP support located the certificate in the OpenPGP keyring.

Importing OpenPGP certificates

Encryption Facility's OpenPGP support provides the **–i** command to import OpenPGP certificates. The certificates can be in ASCII armor form or binary form.

Note: A certificate file kept in z/OS in the ASCII armor form actually has an EBCDIC-encoded content that displays the same as when the certificate is displayed on an ASCII-based platform.

During import, the OpenPGP support attempts to verify any signature sub-packets it encounters. If it encounters a sub-packet that requires a public key that is not readily available in its OpenPGP keyring or Java keystore, it warns the user and asks for permission to continue the import. In addition, if the certificate contains signature sub-packets that assert that the certificate has been revoked or expired, it warns the user and asks for permission to continue the import. Finally, if key material enclosed within the certificate exists in another OpenPGP certificate within the invocation's OpenPGP key ring, it asks if it should replace the existing certificate with certificate to import. Figure 5 is an example of an import command.

Archived

```
/home/suimgwi/>java-jarCSDEncryptionFacility.jar-homedir.-ieswpluto2.asc
CSD0038I Importing OpenPGP certificate with primary DSA public key ID:
B22653E6B463DC6C...
CSD0075A Certificate <B22653E6B463DC6C> already exists in the key ring. Would
you like to replace it? (yes/no)
Yes
CSD0073I Total number of certificates imported successfully: 1
CSD0051I Command processing has completed successfully.
```

Figure 5 Importing certificate example

Exchanging X.509 certificates

Encryption Facility's OpenPGP support does not provide commands for exchanging X.509 certificates. Fortunately, since it relies on Java and the JCE framework, two tools are readily available to work with X.509 certificates: *keytool* and *hwkeytool*.

Note: Java tools currently do not import certificates into Java keystores that are based on the z/OS Security Server, that is, keystore types JCERACFKS and JCECCARACFKS. These keystores are "read-only" keystores from the Java framework perspective. Their contents can only be modified using commands provided in the Security Server. In the case of RACF®, refer to RACF publications for RACF commands that import X.509 certificates. For other security server products, refer to the appropriate documentation for availability of commands and interfaces for importing X.509 certificates.

Using keytool with X.509 Certificates

Use keytool with the JCEKS and JKS keystore types. Figure 6 shows keytool being used on a distributed platform to export an X.509 certificate.

```
keytool.exe -export -keystore keytool.jceks -storepass TKETKE -storetype jceks
-alias first -v -rfc > first.export.asc
```

Figure 6 Keytool syntax

The file first.export.asc is transferred to the z/OS host. Note that unlike ASCII armor with OpenPGP certificates, the file that contains a DER base64-encoded certificate must be in ASCII on z/OS (that is, it must be kept in its binary form, meaning that transfers implying text translation must not be used). Figure 7 shows the import of a self-signed X.509 certificate.

Archived

```
/home/suimgwi/>keytool -import -rfc -alias firstimportRSA -keystore ibmpgp.jks
-storetype jks -storepass TKETKE -file first.export>
Owner: CN=Saheem Granados, OU=zOSRSA, O=STG, L=Pok, ST=NY, C=US
Issuer: CN=Saheem Granados, OU=zOSRSA, O=STG, L=Pok, ST=NY, C=US
Serial number: 46a6474b
Valid from: 7/24/07 2:39 PM until: 10/22/07 2:39 PM
Certificate fingerprints:
         MD5:  76:5C:DA:34:B1:A2:F6:53:1B:E3:46:58:89:0B:97:6F
         SHA1: AF:01:58:14:F6:17:A5:49:E4:F6:2D:7E:15:44:26:C9:F8:02:7C:D0
Trust this certificate? [no]: yes
Certificate was added to keystore
```

Figure 7 Import of self-signed certificate example

The OpenPGP support can use this certificate to encrypt data. The **–rA** option specifies the imported X.509 certificate and the **–e** command forces the OpenPGP support to use public key based encryption.

```
/home/suimgwi/>java -jar CSDEncryptionFacility.jar -homedir . -rA
firstimportRSA -o enc.out.imported -e PERSON1.SMALL.LDIF
CSD0700A File <enc.out.imported> already exists.  Do you want to overwrite?
(yes/no)
Yes
CSD1333A Alias firstimportRSA refers to an X.509 certificate that is not valid.
Do you want to continue? (yes/no)
Yes
CSD0768I Output data can be exchanged with the owner of the key with key ID
65BB2F961C557DEF.
CSD0051I Command processing has completed successfully.
```

Figure 8 Example of -rA option

Using hwkeytool with X.509 certificates

Use of hwkeytool for importing X.509 certificates is very similar to using keytool, and is shown in Figure 9.

```
hwkeytool -import -rfc -alias firstimportRSA -keystore cca2.jks -storetype
JCECCAKS -storepass TKETKE -file first.export.asc -provider
com.ibm.crypto.hdwrCCA.provider.IBMJCECCA
Owner: CN=Saheem Granados, OU=zOSRSA, O=STG, L=Pok, ST=NY, C=US
Issuer: CN=Saheem Granados, OU=zOSRSA, O=STG, L=Pok, ST=NY, C=US
Serial number: 46a6474b
Valid from: Tue Jul 24 14:39:07 EDT 2007 until: Mon Oct 22 14:39:07 EDT 2007
Certificate fingerprints:
         MD5:  76:5C:DA:34:B1:A2:F6:53:1B:E3:46:58:89:0B:97:6F
         SHA1: AF:01:58:14:F6:17:A5:49:E4:F6:2D:7E:15:44:26:C9:F8:02:7C:D0
Trust this certificate? [no]:  y
Certificate was added to keystore
```

Figure 9 hwkeytool example

The main difference is due to the extra parameter and a different store type value needed by hwkeytool. The **–storetype** is JCECCAKS. This type is required for using ICSF/Hardware keys. The provider parameter is needed to enable the IBM CCA JCE provider; similar to the

OpenPGP support's JCE_PROVIDER_LIST configuration keyword, the **-provider** option enables the JCE provider that leverages ICSF and hardware cryptography.

Considerations on certificate exchange

Exchanging certificates is an essential task for establishing trust among trusted business partners. The OpenPGP support and IBM's Java SDK provide the opportunity to finalize the exchange. Many examples were shown in this section, demonstrating the mechanics of exporting and importing certificates.   Aside from these tasks, users must understand that using the tools may not be sufficient for establishing trust. In all the examples shown here, no certificate authority was used for the X.509 certificates. Moreover, OpenPGP certificates use a decentralized approach for trust verification of certificates. Without a trusted certificate authority, users must carefully inspect certificates and establish special procedures for accepting the validity of a certificate. If the public keys are available, the OpenPGP support verifies all signature information in the certificate. This ensures the certificate has not been manipulated. However, users still may need to use Key IDs, fingerprints, visual, and verbal verification with partners to really cement certificate-based trust.

Migrating key pairs

Exchanging public key material is a well-documented endeavor. The public nature of the key material and use of digital certificates and digital signatures reduce the security requirements and thus the complexity needed for transfer. (Note, as previously stated, the benefits of public key exchange do not necessarily mean that establishing trust is a trivial task). Migration of key material that is re-formatting a key so it can still be usable in a different standard, is a completely different matter.

Migration of key pairs implies that the owner of a public/private key pair would like to continue to use the same key pair in their OpenPGP environment. Many times these key pairs were generated for use by other cryptographic tools, or even other OpenPGP tools. The OpenPGP support in Encryption Facility does not provide any direct support for transfer of public/private key pairs. (Note, however, that RFC 2440 describes *Secret Packets* that can serve as a defined syntax for transferring private keys in the OpenPGP space.) Fortunately, the keytool and hwkeytool tools allow for exchange of private keys using the PKCS#12 format.

The use of PKCS#12 format provides many benefits:

- RACF security server allows for importing of password-protected PKCS#12 format files; refer to RACF publications for details about importing password-protected PKCS#12 data.
- IBM Java SDK on z/OS tools allow for imported private keys to be stored in the ICSF PKDS.
- Migration from other OpenPGP tools that provide export of public/private keys in PKCS#12 syntax.

Figure 10 shows exporting a public/private key pair using keytool on a distributed platform.

```
$ keytool.exe -export -keystore keytool.jceks -storepass TKETKE -storetype
jceks -alias firstRSA -rfc -v -pkcs12  > first.export.asc
Enter key password for <firstRSA>TKETKE
```

Figure 10   Exporting public/private key on distributed platform

After the key has been exported, it must be transferred to z/OS. Once again, it must remain in binary format, without any attempt to perform text translation. Here it can be imported into the PKDS, as shown in Figure 11.

```
/home/suimgwi/>hwkeytool -import -rfc -alias firstimportCCARSA -keystore
cca2.jks -storetype JCECCAKS -storepass TKETKE -file first.export.asc -provider
com.ibm.crypto.hdwrCCA.provider.IBMJCECCA -pkcs12 -hardwaretype PKDS
Enter key password for <firstimportCCARSA>TKETKE
Content of pkcs12 file was imported in keystore
```

Figure 11 Importing the keys into PKDS

An additional option was provided on the hwkeytool invocation from this example. The **–hardwaretype** option allows for the user to specify where the key should be stored. The options are RETAINED, CLEAR, or PKDS. RETAINED stores the key material in the Crypto Coprocessor and is not recommended. CLEAR stores the key material in the Java keystore protected by the security semantics of Java keystores. Finally, the PKDS option stores the key in the ICSF PKDS.

Once imported, the key can now be exported as an OpenPGP certificate, as in Figure 12.

```
/home/suimgwi/>java -jar CSDEncryptionFacility.jar -homedir . -a -o pkcs12.as
-eA firstimportCCARSA
CSD0029I Exporting an OpenPGP certificate for firstimportCCARSA...
CSD0026I At least one user ID is required for an OpenPGP certificate.
A user ID consists of three parts: a name, a comment(optional), and an email
address(optional).
CSD0020A Real Name:
redbooks pkcs12
CSD0024A Comment:
pkcs12 ecxample
CSD0022A Email address:
pkcs12@us.ibm.com
CSD0025A You specified user ID: "redbooks pkcs12 <pkcs12 ecxample>
<pkcs12@us.ibm.com>"
Change (N)ame, (C)omment, (E)mail, (X)Cancel or (O)kay to accept?
o
CSD0027A Add another user ID? (yes/no)
n
CSD0019A For how many days should this OpenPGP certificate be valid (0 for
always valid):[0]

CSD0032A Do you want to add the exported OpenPGP Certificate to your OpenPGP
key ring? (yes/no)
Yes
CSD1347I 1 OpenPGP certificate(s) were exported successfully to pkcs12.as.
CSD0051I Command processing has completed successfully.
```

Figure 12 Exporting the key as OpenPGP certificate

Using GNU’s Privacy Guard (gpg) on a distributed platform, the OpenPGP certificate is imported (Figure 13).

Archived

```
$ gpg --import pkcs12.as
gpg: WARNING: using insecure memory!
gpg: please see http://www.gnupg.org/faq.html for more information
gpg: key 1C557DEF: public key "redbooks pkcs12 <pkcs12ecxample>
<pkcs12@us.ibm.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

Figure 13   Importing the OpenPGP certificate

Now, gpg can be used to encrypt data as shown in Figure 14 (note that, as an example, we are encrypting the PKCS#12 file that was initially generated on the distributed platform).

```
$ gpg –t -o encrypted.gpg -r redbooks -e first.export.asc
gpg: WARNING: using insecure memory!
gpg: please see http://www.gnupg.org/faq.html for more information
gpg: 1C557DEF: There is no assurance this key belongs to the named user

pub  1024R/1C557DEF 2007-07-24 redbooks pkcs12 <pkcs12 ecxample> <pkcs12@us.ibm
com>
 Primary key fingerprint: 6802 99C4 7F22 F176 0CD0  9612 65BB 2F96 1C55 7DEF

It is NOT certain that the key belongs to the person named
in the user ID.  If you *really* know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y
```

Figure 14   Encrypting the data[1]

Now that the data has been encrypted using gpg, the OpenPGP support in Encryption Facility must decrypt the data using the imported private key (Figure 15).

[1] Note the gpg output warns for acceptance of trust.

Archived

```
/home/suimgwi/>java -jar CSDEncryptionFacility.jar -homedir .  -o decrypto.out
-d encrypted.gpg
CSD0051I Command processing has completed successfully.
/home/suimgwi/>cat decrypto.out
-----BEGIN CERTIFICATE-----
MIIGpwIBAzCCBmEGCSqGSIb3DQEHAaCCBlIEggZOMIIGSjCCAycGCSqGSIb3DQEHAaCCAxgEggMU
MIIDEDCCAwwGCyqGSIb3DQEMCgECoIICsTCCAqOwJwYKKoZIhvcNAQwBAzAZBBT43vBoroGJi8V7
3f7vNj8L9uhmiAIBCgSCAoAYF4gkEqvWVdDZbAgirzyFXiJRF5+Sws4hZ4OslIeMwFULRJLsI2zY
/TWA8GtzF5WhKD38+ccXD3xCLen54MgGm+IJrYVhM2ovmGIm6SIVVue4mkgOBuzp2+BumIOeo/qS
J6YmL+8scT/kjn3gV1S+QqXfUeGOWzFHvHvqa7uKTdMeJ2SCcwr6SWFOaE3smZCVUkaTd1DWmr96
8WO4L5JTz6P84bfaz+QugHGSgtwmdOSmqqO9oXt5f89jng7+gK2OWrOVKxbxo37vNA2z1bNLJdDk
Ujsflk5MBjkqTVvd79IR4QWs1iJeFJTYkWiuuWl9MNkOOxh6f3+k2ak+V13upctevnCUh9SjCylr
ny+gUc7mJNKqvJ17xLEf3yqxOB5phxrrQMVxYHCmzTytAjOOfd8omT6VPoRwEpQCbU96zvoNPvmE
1ZmIfuMsGOIFKPc+trZRcLzOeWUfV2Hkw3tGoDqofdWQjLmxyL78ARNXBU5adN/hn11XegmZ8OcA
ZWtjj8sugw/p127hvZsrX9wHQDYFHdle1F7vf5/ODGsiCOkkpXeuMADaXo/2u7zSYwAnBZ597Tso
7cdeCXH6bndO6CqAsO5HuWMc59iy6D5CYFVNS377HrVpyneCgkZCtd2FKU6XcxC1Oamw5uMYl8zR
HdAXvpFitCawJIjsORd1Jr32BO4EqreMFwj/kU5biUNz2hfbPmZe5GY4myBaMB6YkhFnIITq1bmJ
LOXA8TY7XynGKVUbckHErb3eZj6ckH++5JROBYmZfYBAbrK/ELkcEa3mIwG8QVwaZY2RB6XwWpuW
VRg23/RvOCSOrFJXd+O4fCyaYv4hBhNmISstfRV1MUgwIQYJKoZIhvcNAQkUMRQeEgBmAGkAcgBz
AHQAUgBTAEEAADAjBgkqhkiG9wOBCRUxFgQUH8LM4QXPP9M9U7CTRj5OS4hSljgwggMbBgkqhkiG
9wOBBwagggMMMIIDCAIBADCCAwEGCSqGSIb3DQEHATAoBgoqhkiG9wOBDAEDMBoEFAmlNfckUNZ1
fkjMwL4+cmjwS+HaAgIHOICCAsh1/UAK76sobtEXUseXYAkO8DcaeFicUy/tUD/NWr4mmKl8dgn8
kQ7HCyzMsXEOXVQUKDFzfyprRXxmAdIZiIkXkHAJkFs8s98vOZAIhfRn79Z5hBwIuN7kwDUNW4Sn
QxKWHtNDSUQjL5uu5liosNz45lVBft75RHahOzzDoZVE+8wyJ+26tlWAHPOdvqY1dUZpn5KSHrBw
I8S41bztDnBex5jQqY3Jd6h+GkqnZ31KZDbAWWSgvA5W8KgqY3FG7rfe+2PMh2LLIsN6DymO6Ke8
w1rpgzAWVO8Kur121GrYbgGOqqy+pWUZjDC5TQjFaRJVoif1a48N15xBr1kDqRr9uQQcWI6cyDQ8
EcBipkbXPyAVTLQDGNthRnBYRO6lOW3X7GaODyIVJUZszgR/t6LOApjxMjp9RH7O+F/mSRcvNLVA
GEdxL/9rhDlOLZLQhacExp9B9LLrcJU/y+RAVloUdHGGWqRfO9nOafnA5iad7rRqiwu6npplccpg
3p8xjigxSR9/AhpHXpPlA9HHcYh4hv3qmlLaMlTHR8yIUNDQFFXgG43NFx8Eiex72wMf5OeEonsa
upPle+Lr+txg2AXLy6dtrlQPg38RQEzLfYrQQ5yhaEhBivPgkiOMa/aM/H5JzP1997hGl137ASsr
5/bKFtvWJz8gBi2FH2+O6V9QB4MIVJjMFvQyuwOvVf2FOADAuUia2mpvF3EjaoogJKKcFBnlgxmP
GCI51LqX/LqqfR3QzUVVQBxDSuyNKUUA6lFpNiRlF1ZS1/z/JlSaiWdjb6OUzJC7WudMhp1Eiej8
NI47oebNgLgVBZ5z2Io7gdHJX7hRlnIZcr2DXPDAawNvnB44JvqQv3GbxaN3rDBNjSZrhCidW56+
g1HPMF/TJejowOW1QPdGNAd7gJf/YGDHWpFCP/k1yZBBHue7ZoXLWsDmvq6ByvyCMDOwITAJBgUr
DgMCGgUABBRbQ2u2yqfGrOqOXrwCZOYe72ZdbwQUTP622khWDXz527doycMwFHYiTm8CAgfQ
-----END CERTIFICATE-----
```

Figure 15   Decrypting the data using the imported private key

The output includes the contents of the decrypted data for completeness.

Considerations on key pair migration

Key migration may be useful in certain environments. Although IBM’s current set of tools do not provide a direct OpenPGP exchange of private key material, the hwkeytool and keytool tools provide a means for migrating key pairs using X.509 and PKCS#12 formats. hwkeytool provides a means for storing the migrated key material in the ICSF PKDS. The RACF security server also provides a means for importing password protected PKCS#12 files. Both serve as very secure repositories for storing cryptographic keys and certificates. One essential point that must be stressed is that transport of private key material must be done with extreme caution.

About the author

This paper was produced at the International Technical Support Organization, Poughkeepsie Center.

Saheem Granados is an Advisory software engineer who has worked for IBM for over 9 years. He is a Certified Information Systems Security Professional who throughout his career has been a designer and developer for critical z/OS security software, including: Net.Commerce™ for OS/390®, Security Server LDAP Server, Tivoli® Federate Identity Manager, and Trusted Key Entry. As the development product owner of the Encryption Facility V1.2 OpenPGP support, Saheem designed the solution and led a team of developers to successful completion of all required development activities.

Thanks to the following people for their contributions to this project:

Paola Bari
International Technical Support Organization, Poughkeepsie Center

Patrick Kappeler
IBM France

Archived

Archived

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Archived

© Copyright International Business Machines Corporation 2007. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This document REDP-4341-00 was created or updated on August 6, 2007.

Send us your comments in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Redpaper™

Archived

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks (logo) ®	Net.Commerce™	Tivoli®
z/OS®	OS/390®	
IBM®	RACF®	

The following terms are trademarks of other companies:

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.