OraRSA

Table of contents

Introduction

OraRSA (ORA_RSA) is a PL/SQL package for the Oracle(c) Database product family, version 11 and above, that offers RSA cryptography procedures.

Internally the package is implemented as a set of Java Stored Procedures.

In order to perform asymmetric RSA cryptography operations the package utilizes existing public and private RSA (X.509) keys.

Install

Setup

Step 0) Extracting the package

Extract the distribution ZIP archive to a folder on your development disk drive (referred below as "***extraction folder***")

Load the JAR files

Step 1) Load the JAR files

Load the JAR files using the *loadjava.sh/.bat* utility located in the Oracle Database instance /BIN/ folder.
(For example on Windows C:\app\<User>\product\<oracle version>\db_1\BIN \loadjava.bat)

Note: Replace below *user/pass* with a database schema/user name and password applicable for your Oracle Database.

Oracle 11, 12, 18c

[Windows environment]
loadjava.bat -r -v – u user/pass [extraction folder]\SetupFiles\Oracle11_18\jce-jdk13-161.jar
loadjava.bat -r -v – u user/pass [extraction folder]\SetupFiles\Oracle11_18\ora-rsa-<version>.jar

[Unix environment]
loadjava.sh -r -v – u user/pass [extraction folder]/SetupFiles/Oracle11_18/jce-jdk13-161.jar
loadjava.sh -r -v – u user/pass [extraction folder]/SetupFiles/Oracle11_18/ora-rsa-<version>.jar

Oracle 19c

[Windows environment]
loadjava.bat -r -v – u user/pass [extraction folder]\SetupFiles\Oracle19\bcprov-jdk15on-161.jar
loadjava.bat -r -v – u user/pass [extraction folder]\SetupFiles\Oracle19\ora-rsa-<version>.jar

[Unix environment]
loadjava.sh -r -v – u user/pass [extraction folder]/SetupFiles/Oracle19/bcprov-jdk15on-161.jar
loadjava.sh -r -v – u user/pass [extraction folder]/SetupFiles/Oracle19/ora-rsa-<version>.jar

ORA_RSA package

Step 2) Registering the ORA_RSA PL/SQL package

Using your favorite PL/SQL execution environment (SQL*Plus, Oracle(c) SQLDeveloper(c), etc.) execute the scripts:

[extraction folder]/SetupFiles/Ora_RSA_Package.sql
[extraction folder]/SetupFiles/Ora_RSA_Package_Body.sql

Step 3) Permissions

Using your favorite PL/SQL execution environment (SQL*Plus, Oracle(c) SQLDeveloper(c), etc.) execute:

```
CALL dbms_java.grant_permission( 'user',
'SYS:java.security.SecurityPermission', 'putProviderProperty.BC', '' );
CALL dbms_java.grant_permission( 'user',
'SYS:java.security.SecurityPermission', 'insertProvider.BC', '' );
commit;
```

Note: Replace above 'user' with the database user/schema name that will execute the code.

ORA_SMIME package

Step 1) Registering the ORA_SMIME PL/SQL package

Using your favorite PL/SQL execution environment (SQL*Plus, Oracle(c) SQLDeveloper(c), etc.) execute the scripts:

[extraction folder]/SetupFiles/Ora_SMIME_Package.sql
[extraction folder]/SetupFiles/Ora_SMIME_Package_Body.sql

Step 2) Permissions

Using your favorite PL/SQL execution environment (SQL*Plus, Oracle(c) SQLDeveloper(c), etc.) execute:

```
CALL dbms_java.grant_permission( 'user',
'SYS:java.security.SecurityPermission', 'putProviderProperty.BC', '' );
CALL dbms_java.grant_permission( 'user',
'SYS:java.security.SecurityPermission', 'insertProvider.BC', '' );
commit;
```

Note: Replace above 'user' with the database user/schema name that will execute the code.

Step 3) Email sending Permissions

In order to use ORA_SMIME with UTL_SMTP, proper SMTP permissions must be granted for UTL_SMTP to each schema (database user with login permissions) that will access UTL_SMTP.

Note: Please refer to the Oracle documentation for UTL_SMTP for more details.

The example code below creates ACL (Access Control List) for user called 'NASKO' to use the SMTP server at localhost, port 25:

```
begin
  DBMS_NETWORK_ACL_ADMIN.CREATE_ACL('acl_for_mail.xml', 'ACL for Mail
Job','NASKO', TRUE, 'connect');

  DBMS_NETWORK_ACL_ADMIN.ASSIGN_ACL('acl_for_mail.xml','localhost', 25);

COMMIT;
end;
/
```

```
grant execute on UTL_SMTP to NASKO;
```

username/password and connecting to Oracle

Oracle database located on the same machine

username/password - where *username* is a user(schema) name in the Oracle Database and *password* is its password.

Oracle database located on a Remote host machine

TNS Name

username/password@***database*** - in this case *database* can be a TNS name or Net8 name-value list

If we want to install the script into a remote machine specified with @host:port:SID then the additional **-thin** parameter must also be used:

Host and Port

loadjava -r -v **-thin** -u *username/password*@**host:port:SID** <jar file to be loaded>

Upgrade

Upgrading to a new version of ORA_RSA requires **first to unload the old version JAR files** from the Oracle© database:

Oracle 11, 12, 18c

dropjava -r -v – u user/pass [extraction folder]\SetupFiles\Oracle11_18\jce-jdk13-161.jar
dropjava -r -v – u user/pass [extraction folder]\SetupFiles\Oracle11_18\ora-rsa-<version>.jar

Oracle 19c

dropjava -r -v – u user/pass [extraction folder]\SetupFiles\Oracle19\bcprov-jdk15on-161.jar
dropjava -r -v – u user/pass [extraction folder]\SetupFiles\Oracle19\ora-rsa-<version>.jar

Then continue to Setup

Note: Replace above *user/pass* with the database user/schema name and password used in the initial Setup.

Uninstall

The uninstall process is similar to the Upgrade and requires **first to unload the old version JAR files** from the Oracle© database:

Oracle 11, 12, 18c

dropjava -r -v – u user/pass [extraction folder]\SetupFiles\Oracle11_18\jce-jdk13-161.jar
dropjava -r -v – u user/pass [extraction folder]\SetupFiles\Oracle11_18\ora-rsa-<version>.jar

Oracle 19c

dropjava -r -v – u user/pass [extraction folder]\SetupFiles\Oracle19\bcprov-jdk15on-161.jar
dropjava -r -v – u user/pass [extraction folder]\SetupFiles\Oracle19\ora-rsa-<version>.jar

and afterwards to drop the ORA_RSA package using your favorite PL/SQL execution environment (SQL*Plus, Oracle(c) SQLDeveloper(c), etc.

```
SQL> drop package ORA_RSA;
SQL> drop package body ORA_RSA;

SQL> drop package ORA_SMIME;
SQL> drop package body ORA_SMIME;
```

Note: Replace above *user/pass* with the database user/schema name and password that were used in step Setup

Switch from Trial to Production version

To switch from an evaluation (trial) version of the software to a licensed production version, please execute the steps in the Upgrade process.

RSA (X.509) Keys

In order to perform RSA asymmetric cryptography, you will need RSA keys (also known as X.509 keys or certificates). The package assumes that you already have such keys generated by other software tools (like OpenSSL e.g.)

The PL/SQL functions provided by the package accept the keys transparently either as local file system path locations or serialized in text format (PEM format) or binary format (DER format). RSA keys can be met in a wide variety of file formats and the package internally tries to recognize the format of the provided key/certificate and utilize it.

Constants

Hash algorithm

Constants for choosing the hash function for digital signatures:

ORA_RSA.HASH_SHA1 for SHA1withRSA
ORA_RSA.HASH_SHA224 for SHA224withRSA
ORA_RSA.HASH_SHA256 for SHA256withRSA
ORA_RSA.HASH_SHA384 for SHA384withRSA
ORA_RSA.HASH_SHA512 for SHA512withRSA

Error codes

The ORA_RSA package defines the following constants (used for exception handling):

-- wrong password specified for a private RSA key
RSA_WRONG_PASSWORD_ERR CONSTANT INTEGER := 871;
-- error loading RSA key
RSA_KEY_ERR CONSTANT INTEGER := 872;
-- RSA operation error
RSA_ENCRYPTION_ERR CONSTANT INTEGER := 873;
-- I/O error
RSA_GENERAL_IO_ERR CONSTANT INTEGER := 770;

ORA_RSA Functions

The API list is available online at http://www.didisoft.com/ora-rsa/tutorial/functions/

ENCRYPT

function ENCRYPT(message RAW, public_key RAW) return RAW

Description:
RSA encrypts data with an asymmetric public key.

Parameters:
message - message to be encrypted
public_key - the bytes of the RSA (X.509) public key

Result
the encrypted field

Online example:
http://www.didisoft.com/ora-rsa/tutorial/encrypting/

ENCRYPT_WITH_MODULUS

function ENCRYPT_WITH_MODULUS(message RAW, public_key_modulus VARCHAR2, public_key_exponent VARCHAR2) return RAW

Description:
RSA encrypts data with an asymmetric public key supplied as RSA algorithm parameters

Parameters:
message - message to be encrypted
public_key_modulus - the modulus of the RSA public key as hexadecimal string
public_key_exponent - the exponent of the RSA public key as hexadecimal string

Result
the encrypted field

Online example:
http://www.didisoft.com/ora-rsa/tutorial/encrypting/

DECRYPT

function DECRYPT(data RAW, private_key RAW) return RAW
function DECRYPT(data RAW, private_key RAW, key_password varchar2) return RAW

Decrypts an RSA encrypted field with a specified RSA private key.

Parameters:
message - encrypted message to be decrypted
private_key - the RSA private key bytes PEM encoded or DER encoded, or in .pfx/.p12 format
key_password - password that unlocks the private key (optional)

Result
the decrypted data

Online example:
http://www.didisoft.com/ora-rsa/tutorial/decrypting/

DECRYPT_WITH_MODULUS

function DECRYPT_WITH_MODULUS(data RAW, private_key_modulus VARCHAR2, private_key_exponent VARCHAR2) return RAW

Decrypts an RSA encrypted field with a specified RSA private key supplied as raw RSA algorithm parameters

Parameters:
message - encrypted message to be decrypted
private_key_modulus - the modulus of the RSA private key as hexadecimal string
private_key_exponent - the exponent of the RSA private key as hexadecimal string

Result
the decrypted data

Online example:
http://www.didisoft.com/ora-rsa/tutorial/decrypting/

SIGN

function SIGN(message RAW, private_key RAW) return BLOB
function SIGN(message RAW, private_key RAW, private_key_password varchar2) return BLOB

Description:
Produces an RSA digital signature for using a specified RSA private key.

Parameters:
message - data for which a digital signature will be generated
private_key - the RSA private key bytes PEM encoded or DER encoded, or in .pfx/.p12 format
key_password - password that unlocks the private key (optional)

Result
RSA digital signature over SHA-1 hash of the input data

VERIFY

function VERIFY(message RAW, signature RAW, public_key RAW) return pls_integer

verifies an RSA digital signature computed over data field with a specified public key.

Parameters:
message - signed message for which the signature was computed
signature - digital signature of the message
public_key - absolute file path on the server to the public key or the public key PEM or DER encoded

Result
1 - if the signature was validated with the provided public key
0 - if the signature cannot be validated with the provided public key

GETVERSION

function GETVERSION return VARCHAR2

Description:
Returns the build version of the package

Result
the build version number of the package

Example

```
select ORA_RSA.GETVERSION from dual
```

IS_TRIAL_VERSION

function IS_TRIAL_VERSION return pls_integer

Description:
Returns is the installed package a trial (evaluation) or production version

Result
1 - if the package is a 30 day trial (evaluation) version
0 - if the package is a production version

Example

```
select ORA_RSA.IS_TRIAL_VERSION from dual
```

GET_RSA_ERROR

function GET_RSA_ERROR return pls_integer

Description:
Returns a value used for error identification

Result

Value	Meaning
ORA_RSA.RSA_WRONG_PASSWORD_ERR	The provided password for an RSA key is wrong
ORA_RSA.RSA_KEY_ERR	Broken or illegal RSA key provided
ORA_RSA.RSA_ENCRYPTION_ERR	RSA operation error
ORA_RSA.RSA_GENERAL_IO_ERR	Data I/O error

Note: The values for the constants above can be seen in section Constants.

Example
Check section exception handling

ORA_SMIME Functions

WRITE_DATA

```
procedure WRITE_DATA(message IN OUT CLOB, data VARCHAR2 CHARACTER SET ANY_CS)
```

Description:
Appends data into message in similar way to UTL_SMTP.WRITE_DATA.

Parameters:
message - destination message
data - data to be appended

Result
None

Online example:
http://www.didisoft.com/ora-rsa/tutorial/encrypting/

WRITE_RAW_DATA

```
procedure WRITE_RAW_DATA(message IN OUT CLOB, data RAW)
```

Description:
Appends data into message in similar way to UTL_SMTP.WRITE_RAW_DATA.

Parameters:
message - destination message
data - data to be appended

Result
None

Online example:
http://www.didisoft.com/ora-rsa/tutorial/encrypting/

SIGN_EMAIL

```
function SIGN_EMAIL(message CLOB, private_key BLOB, private_key_password
varchar2) return CLOB

Produces S/MIME signed email

PARAMETERS
   message      email body
   private_key   BLOB obtained from a .pfx private key file
   private_key_password password for the .pfx file
 RETURN
   S/MIME signed message as CLOB. Must be passed afterwards to
UTL_SMTP.WRITE_DATA
 EXCEPTIONS
   See Exception handling
```

Exception Handling

Upon error the ORA_RSA package functions throw exception of type **RSA_EXCEPTION**.

In order to identify the exact cause of the error the helper function **GET_RSA_ERROR** can be used as shown below:

```
BEGIN

  ORA_RSA.<method>( ...

EXCEPTION

   WHEN ORA_RSA.RSA_EXCEPTION THEN
     BEGIN
       IF ORA_RSA.GET_RSA_ERROR() = ORA_RSA.RSA_WRONG_PASSWORD_ERR THEN
         DBMS_OUTPUT.PUT_LINE('The password for the private key is not
matching: ' || SQLERRM);
       ELSIF ORA_RSA.GET_RSA_ERROR() = ORA_RSA.RSA_KEY_ERR THEN
         DBMS_OUTPUT.PUT_LINE('The provided key is not a valid RSA key.');
       ELSIF ORA_RSA.GET_RSA_ERROR() = ORA_RSA.RSA_ENCRYPTION_ERR THEN
         DBMS_OUTPUT.PUT_LINE('Error when performing RSA operation: ' ||
SQLERRM);
       ELSIF ORA_RSA.GET_RSA_ERROR() = ORA_RSA.RSA_GENERAL_IO_ERR THEN
         DBMS_OUTPUT.PUT_LINE('I/O error: ' || SQLERRM);
       END IF;
     END;

   WHEN OTHERS THEN
     DBMS_OUTPUT.PUT_LINE('General error : ' || SQLERRM );
END;
/
```

Support and Contact

Technical support

To receive general information or **technical support**, please contact us at support@didisoft.com.

Sales

For questions related to sales, volume licensing, or OEM licensing, please contact us at sales@didisoft.com.

Product Updates

If you have purchased the library you can access our Customers' Portal where you can download new versions.

Newsletter

To receive product update news, you can subscribe to our Newsletter

For further information, visit us at www.didisoft.com
If you have any ideas, wishes, questions or criticism, don't hesitate to contact us. We will be glad to hear from you.