

# DidiSoft OraPGP version 1.4

## Table of contents

---

Introduction .....	4
Setup .....	5
Setup .....	5
Uninstall .....	5
From Trial to Production .....	6
user/pass .....	6
Upgrade .....	7
From version 1.2 .....	7
From version 1.0 .....	7
Constants .....	8
Exception handling .....	8
Signature verification .....	8
Functions .....	9
ENCRYPT .....	9
ENCRYPT (VARCHAR2) .....	9
ENCRYPT_BLOB .....	9
ENCRYPT_CLOB .....	9
ENCRYPT_RAW .....	10
DECRYPT .....	10
DECRYPT (VARCHAR2) .....	10
DECRYPT_BLOB .....	11
DECRYPT_CLOB .....	11
DECRYPT_RAW .....	11
SIGN .....	12
SIGN (VARCHAR2) .....	12
SIGN_BLOB .....	12
SIGN_CLOB .....	12
SIGN_RAW .....	13
VERIFY .....	13
VERIFY (VARCHAR2) .....	13
VERIFY_BLOB .....	13
VERIFY_CLOB .....	14
VERIFY_RAW .....	14
EXTRACT_SIGNED .....	14
EXTRACT_SIGNED .....	14
EXTRACT_SIGNED_BLOB .....	14
EXTRACT_SIGNED_CLOB .....	15
EXTRACT_SIGNED_RAW .....	15
SIGN_AND_ENCRYPT .....	15
SIGN_AND_ENCRYPT (VARCHAR2) .....	15
SIGN_AND_ENCRYPT_BLOB .....	16
SIGN_AND_ENCRYPT_CLOB .....	16
SIGN_AND_ENCRYPT_RAW .....	16
VERIFY_ENCRYPTED .....	17
VERIFY_ENCRYPTED (VARCHAR2) .....	17
VERIFY_ENCRYPTED_BLOB .....	17
VERIFY_ENCRYPTED_CLOB .....	18
VERIFY_ENCRYPTED_RAW .....	18
Analytical .....	18
IS_ENCRYPTED .....	18

SIGNING_KEY_ID .....	19
IS_SIGNED .....	19
ENCRYPTION_KEY_ID .....	19
Misc .....	20
IS_TRIAL_VERSION .....	20
VERSION .....	20
SET_ASCII_VERSION_HEADER .....	20
GET_ASCII_VERSION_HEADER .....	20
SET_INTEGRITY_PROTECT .....	20
Exception handling .....	22
Support and Contact .....	23

## Introduction

---

OraPGP is a PL/SQL package for the Oracle(c) Database product family, version 11 and above, that offers OpenPGP cryptography procedures compatible with the OpenPGP Standard defined in [RFC 1991](#), [RFC 4880](#) and [RFC 6637](#).

The package is implemented as a set of Java Stored Procedures.

## Setup

---

### Setup

**Step 1) Extract** the distribution ZIP archive to a folder on your disk drive (referred below as *[extraction folder]*)

#### **Step 2) 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\11.1.0\db\_1\BIN\loadjava.bat*)

**Note:** check the [notes regarding the user/pass](#) database user name and how to specify a remote machine TNS name or SID.

#### *[Windows environment]*

```
loadjava.bat -resolve -verbose -user user/pass [extraction folder]
\SetupFiles\ora-pgp-1.4.jar
```

#### *[Unix/Linux environment]*

```
loadjava.sh -resolve -verbose -user user/pass [extraction folder]/
SetupFiles/ora-pgp-1.4.jar
```

#### **Step 3) Register the ORA\_PGP PL/SQL package**

using your favorite PL/SQL environment (SQL\*Plus, Oracle(c) SQLDeveloper(c), etc.) execute the script:

```
[extraction folder]/SetupFiles/Ora_Pgp_Package.sql
[extraction folder]/SetupFiles/Ora_Pgp_Package_Body.sql
```

#### **Step 4) Check that everything is working**

Execute in your PL/SQL environment:

```
SELECT ORA_PGP.VERSION FROM dual
```

### Uninstall

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

### Windows

```
dropjava.bat -resolve -verbose -user user/pass [extraction folder]
\SetupFiles\ora-pgp-<version>.jar
```

## Unix/Linux

```
dropjava.sh -r -v -u user/pass [extraction folder]/SetupFiles/ora-pgp-
<version>.jar
```

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

```
SQL> drop package ORA_PGP;
```

**Note:** check the [notes regarding the user/pass](#) database user name.

## From Trial to Production

To switch from an evaluation (trial) version of the software to a licensed production version, please execute the steps in the [Uninstal process](#).and then proceed with the [Setup](#)

### user/pass

The -user parameter of the loadjava and dropjava commands can be simply:

### 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

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

### Host and Port

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:

```
loadjava -resolve -verbose -thin -user username/password@host:port:SID
```

for example

```
loadjava -resolve -verbose -thin -user scott/tiget@127.0.0.1:1521:ORCL
```

# Upgrade

---

## From version 1.2

In version 1.3 and above the ORA\_PGP package no longer needs specific java execution permissions.

You will have to unload the old BouncyCastle jar files and the classes of the package (contained in ora-pgp-<version>.jar):

**Unload the old version JAR files** from the Oracle© database:

```
dropjava.sh/.bat -resolve -verbose -user user/pass [extraction folder]\SetupFiles\old\jce-jdk13-151.jar
dropjava.sh/.bat -resolve -verbose -user user/pass [extraction folder]\SetupFiles\old\bcpjg-jdk13-151.jar
dropjava.sh/.bat -resolve -verbose -user user/pass [extraction folder]\SetupFiles\ora-pgp-<version>.jar
```

Then you can continue with the [fresh Setup](#).

## From version 1.0

**Unload the old version JAR files** from the Oracle© database:

```
dropjava.sh/.bat -resolve -verbose -user user/pass [extraction folder]\SetupFiles\jce-jdk13-151.jar
dropjava.sh/.bat -resolve -verbose -user user/pass [extraction folder]\SetupFiles\bcpjg-jdk13-151.jar
dropjava.sh/.bat -resolve -verbose -user user/pass [extraction folder]\SetupFiles\pgplib-2.7.0.jar
dropjava.sh/.bat -resolve -verbose -user user/pass [extraction folder]\SetupFiles\ora-pgp-1.0.0.jar
```

After the old JAR files are unloaded continue to [Setup](#)

**Note:** Replace above user/password with the database [username/schema and password](#) used in the [initial Setup](#).

**Note:** If you don't have the old JAR files, you can use the current ones and use the [current Uninstall commands](#) instead.

## Constants

---

### Exception handling

Constants for [exception handling](#):

- wrong password specified for a private OpenPGP key  
**RSA\_WRONG\_PASSWORD\_ERR** CONSTANT INTEGER := 801;
- an input OpenPGP private key is invalid or corrupted  
**PGP\_WRONG\_PRIVATE\_KEY\_ERR** CONSTANT INTEGER := 802;
- an input OpenPGP public key is invalid or corrupted  
**PGP\_NO\_PUBLIC\_KEY\_ERR** CONSTANT INTEGER := 803;
- expected OpenPGP signed content passed for verification is a detached OpenPGP signature  
**PGP\_DETACHED\_SIGNATURE\_ERR** CONSTANT INTEGER := 804;
- expected OpenPGP signed content passed for verification is an encrypted OpenPGP message  
**PGP\_DATA\_ENCRYPTED\_ERR** CONSTANT INTEGER := 805;
- expected key encrypted OpenPGP message is a password encrypted OpenPGP message  
**PGP\_PASS\_ENCRYPTED\_ERR** CONSTANT INTEGER := 806;
- OpenPGP message integrity check failed  
**PGP\_INTEGRITY\_CHECK\_ERR** CONSTANT INTEGER := 807;
- expected input OpenPGP message is either corrupted or not an OpenPGP message at all  
**PGP\_NO\_PGP\_DATA\_ERR** CONSTANT INTEGER := 808;
- public key passed for encryption is expired  
**PGP\_KEY\_EXPIRED\_ERR** CONSTANT INTEGER := 809;
- public key passed for encryption is revoked  
**PGP\_KEY\_REVOKED\_ERR** CONSTANT INTEGER := 810;
- general OpenPGP error non among the other known cases  
**PGP\_GENERAL\_PGP\_ERR** CONSTANT INTEGER := 800;
- I/O error  
**PGP\_GENERAL\_IO\_ERR** CONSTANT INTEGER := 700;

### Signature verification

Constants for signature verification

- signature has been verified  
**SIGNATURE\_VERIFIED** CONSTANT INTEGER := 1;
- signature was tempered and cannot be verified  
**SIGNATURE\_BROKEN** CONSTANT INTEGER := 0;
- the public key used to check the signature doesn't match  
**PUBLIC\_KEY\_NOT\_MATCH** CONSTANT INTEGER := -1;
- the checked data has no signature  
**NO\_SIGNATURE** CONSTANT INTEGER := -2;



## Functions

---

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

### ENCRYPT

#### ENCRYPT (VARCHAR2)

```
function ENCRYPT(message varchar2, public_key varchar2) return varchar2
function ENCRYPT(message BLOB, public_key varchar2) return BLOB
function ENCRYPT(message varchar2, public_key BLOB) return varchar2
function ENCRYPT(message BLOB, public_key BLOB) return BLOB
```

OpenPGP encrypts a field with a specified public key.

#### Parameters:

*message* - message to be encrypted

*public\_key (varchar2)* - absolute file path on the server to the public key or the public key as ASCII armored text

*public\_key (BLOB)* - public key encryption key as BLOB

#### Result

the encrypted field

#### Online example:

<http://www.didisoft.com/ora-pgp/tutorial/encrypting/>

### ENCRYPT\_BLOB

```
function ENCRYPT_BLOB(message BLOB, public_key varchar2, ascii_armor BOOLEAN) return BLOB
function ENCRYPT_BLOB(message BLOB, public_key BLOB, ascii_armor BOOLEAN) return BLOB
```

OpenPGP encrypts a BLOB field with a specified public key.

#### Parameters:

*message* - BLOB to be encrypted

*public\_key (varchar2)* - absolute file path on the server to the public key or the public key as ASCII armored text

*public\_key (BLOB)* - OpenPGP public key as BLOB

*ascii\_armor* - if TRUE the output will be in ASCII armored format, if FALSE then in binary format

#### Result

the encrypted field

#### Online example:

<http://www.didisoft.com/ora-pgp/tutorial/encrypting/>

### ENCRYPT\_CLOB

```
function ENCRYPT_CLOB(message CLOB, public_key varchar2) return CLOB
function ENCRYPT_CLOB(message CLOB, public_key BLOB) return CLOB
```

OpenPGP encrypts a CLOB field with a specified public key.

**Parameters:**

*message* - CLOB to be encrypted

*public\_key (varchar2)* - absolute file path on the server to the public key or the public key as ASCII armored text

*public\_key (BLOB)* - OpenPGP public key as BLOB

**Result**

the encrypted field

**Online example:**

<http://www.didisoft.com/ora-pgp/tutorial/encrypting/>

## ENCRYPT\_RAW

**function ENCRYPT\_RAW(message RAW, public\_key varchar2, ascii\_armor BOOLEAN) return RAW**

**function ENCRYPT\_RAW(message RAW, public\_key BLOB, ascii\_armor BOOLEAN) return RAW**

OpenPGP encrypts a BLOB field with a specified public key.

**Parameters:**

*message* - RAW to be encrypted

*public\_key (varchar2)* - absolute file path on the server to the public key or the public key as ASCII armored text

*public\_key (BLOB)* - OpenPGP public key as BLOB

*ascii\_armor* - if TRUE the output will be in ASCII armored format, if FALSE then in binary format

**Result**

the encrypted field

**Online example:**

<http://www.didisoft.com/ora-pgp/tutorial/encrypting/>

## DECRYPT

### DECRYPT (VARCHAR2)

**function DECRYPT(message varchar2, private\_key varchar2, key\_password varchar2) return varchar2**

**function DECRYPT(data BLOB, private\_key varchar2, key\_password varchar2) return BLOB**

OpenPGP decrypts a field with a specified private key.

**Parameters:**

*message* - [encrypted message](#) to be decrypted

*private\_key* - absolute file path on the server to the private key or the key as ASCII armored text

*key\_password* - password that unlocks the private key

**Result**

the decrypted field

**Online example:**

<http://www.didisoft.com/ora-pgp/tutorial/decrypting/>

## DECRYPT\_BLOB

```
function DECRYPT_BLOB(message BLOB, private_key varchar2, key_password varchar2) return BLOB
```

```
function DECRYPT_BLOB(message BLOB, private_key BLOB, key_password varchar2) return BLOB
```

OpenPGP decrypts a field with a specified private key.

### Parameters:

*message* - [encrypted message](#) to be decrypted

*private\_key* - private key

*key\_password* - password that unlocks the private key

### Result

the decrypted field

### Online example:

<http://www.didisoft.com/ora-pgp/tutorial/decrypting/>

## DECRYPT\_CLOB

```
function DECRYPT_CLOB(message CLOB, private_key varchar2, key_password varchar2) return CLOB
```

```
function DECRYPT_CLOB(message CLOB, private_key BLOB, key_password varchar2) return CLOB
```

OpenPGP decrypts a field with a specified private key.

### Parameters:

*message* - [encrypted message](#) to be decrypted

*private\_key* - private key

*key\_password* - password that unlocks the private key

### Result

the decrypted field

### Online example:

<http://www.didisoft.com/ora-pgp/tutorial/decrypting/>

## DECRYPT\_RAW

```
function DECRYPT_RAW(message RAW, private_key varchar2, key_password varchar2) return RAW
```

```
function DECRYPT_RAW(message RAW, private_key BLOB, key_password varchar2) return RAW
```

OpenPGP decrypts a field with a specified private key.

### Parameters:

*message* - [encrypted message](#) to be decrypted

*private\_key* - private key

*key\_password* - password that unlocks the private key

### Result

the decrypted field

### Online example:

<http://www.didisoft.com/ora-pgp/tutorial/decrypting/>

## SIGN

### SIGN (VARCHAR2)

```
function SIGN(message varchar2, private_key varchar2, key_password varchar2) return varchar2  
function SIGN(data BLOB, private_key varchar2, key_password varchar2) return BLOB
```

OpenPGP signs a field with a specified private key.

#### Parameters:

*message* - message to be signed

*private\_key* - absolute file path on the server to the private key or the key as ASCII armored text

*key\_password* - password that unlocks the private key

#### Result

the signed field

#### Online example

<https://www.didisoft.com/ora-pgp/tutorial/signing/>

### SIGN\_BLOB

```
function SIGN_BLOB(message BLOB, private_key varchar2, key_password varchar2, ascii_armor  
BOOLEAN) return BLOB  
function SIGN_BLOB(message BLOB, private_key BLOB, key_password varchar2, ascii_armor  
BOOLEAN) return BLOB
```

OpenPGP signs a field with a specified private key.

#### Parameters:

*message* - message to be signed

*private\_key* - absolute file path on the server to the private key or the key as ASCII armored text

*key\_password* - password that unlocks the private key

*ascii\_armor* - if TRUE the output will be in ASCII armored format, if FALSE then in binary format

#### Result

the signed field

#### Online example

<https://www.didisoft.com/ora-pgp/tutorial/signing/>

### SIGN\_CLOB

```
function SIGN_CLOB(message CLOB, private_key varchar2, key_password varchar2) return CLOB  
function SIGN_CLOB(message CLOB, private_key BLOB, key_password varchar2) return CLOB
```

OpenPGP signs a field with a specified private key.

#### Parameters:

*message* - message to be signed

*private\_key* - absolute file path on the server to the private key or the key as ASCII armored text

*key\_password* - password that unlocks the private key

#### Result

the signed field in ASCII armored format

#### Online example

<https://www.didisoft.com/ora-pgp/tutorial/signing/>

## SIGN\_RAW

function SIGN\_RAW(message RAW, private\_key varchar2, key\_password varchar2, ascii\_armor BOOLEAN) return RAW

function SIGN\_RAW(message RAW, private\_key BLOB, key\_password varchar2, ascii\_armor BOOLEAN) return RAW

OpenPGP signs a field with a specified private key.

### Parameters:

*message* - message to be signed

*private\_key* - absolute file path on the server to the private key or the key as ASCII armored text

*key\_password* - password that unlocks the private key

*ascii\_armor* - if TRUE the output will be in ASCII armored format, if FALSE then in binary format

### Result

the signed field

### Online example

<https://www.didisoft.com/ora-pgp/tutorial/signing/>

## VERIFY

### VERIFY (VARCHAR2)

function VERIFY(message varchar2, public\_key varchar2, decrypted\_message OUT varchar2) return pls\_integer

function VERIFY(message BLOB, public\_key varchar2) return pls\_integer

extracts and verifies the signature of an OpenPGP [signed only](#) field with a specified public key.

### Parameters:

*message* - signed message to be verified and extracted

*public\_key* - absolute file path on the server to the public key or the public key as ASCII armored text

OUT *decrypted\_message* - the extracted message

### Result

See: [Signature verification constants](#)

### Online example

<https://www.didisoft.com/ora-pgp/tutorial/verifying/>

### VERIFY\_BLOB

function VERIFY\_BLOB(message BLOB, public\_key varchar2) return pls\_integer

function VERIFY\_BLOB(message BLOB, public\_key BLOB) return pls\_integer

Verifies the signature of an OpenPGP [signed only](#) field with a specified public key.

### Parameters:

*message* - signed message to be verified and extracted

*public\_key* - absolute file path on the server to the public key or the public key as ASCII armored text

### Result

See: [Signature verification constants](#)

#### Online example

<https://www.didisoft.com/ora-pgp/tutorial/verifying/>

### VERIFY\_CLOB

```
function VERIFY_CLOB(message CLOB, public_key varchar2) return pls_integer  
function VERIFY_CLOB(message CLOB, public_key BLOB) return pls_integer
```

Verifies the signature of an OpenPGP [signed only](#) field with a specified public key.

#### Parameters:

*message* - signed message to be verified and extracted

*public\_key* - absolute file path on the server to the public key or the public key as ASCII armored text

#### Result

See: [Signature verification constants](#)

#### Online example

<https://www.didisoft.com/ora-pgp/tutorial/verifying/>

### VERIFY\_RAW

```
function VERIFY_RAW(message RAW, public_key varchar2) return pls_integer  
function VERIFY_RAW(message RAW, public_key BLOB) return pls_integer
```

Verifies the signature of an OpenPGP [signed only](#) field with a specified public key.

#### Parameters:

*message* - signed message to be verified and extracted

*public\_key* - absolute file path on the server to the public key or the public key as ASCII armored text

#### Result

See: [Signature verification constants](#)

#### Online example

<https://www.didisoft.com/ora-pgp/tutorial/verifying/>

### EXTRACT\_SIGNED

### EXTRACT\_SIGNED

```
function EXTRACT_SIGNED(message varchar2) return varchar2
```

extracts data contained in OpenPGP [signed only](#) field

#### Parameters:

*message* - signed message to be extracted

#### Result

contents of the signed message

#### Online example

<https://www.didisoft.com/ora-pgp/tutorial/verifying/>

### EXTRACT\_SIGNED\_BLOB

```
function EXTRACT_SIGNED_BLOB(message BLOB) return BLOB
```

extracts data contained in OpenPGP [signed only](#) field

**Parameters:**

*message* - signed message to be extracted

**Result**

contents of the signed message

**Online example**

<https://www.didisoft.com/ora-pgp/tutorial/verifying/>

## EXTRACT\_SIGNED\_CLOB

**function EXTRACT\_SIGNED\_CLOB(message CLOB) return CLOB**

extracts data contained in OpenPGP [signed only](#) field

**Parameters:**

*message* - signed message to be extracted

**Result**

contents of the signed message

**Online example**

<https://www.didisoft.com/ora-pgp/tutorial/verifying/>

## EXTRACT\_SIGNED\_RAW

**function EXTRACT\_SIGNED\_RAW(message RAW) return RAW**

extracts data contained in OpenPGP [signed only](#) field

**Parameters:**

*message* - signed message to be extracted

**Result**

contents of the signed message

**Online example**

<https://www.didisoft.com/ora-pgp/tutorial/verifying/>

## SIGN\_AND\_ENCRYPT

### SIGN\_AND\_ENCRYPT (VARCHAR2)

**function SIGN\_AND\_ENCRYPT(message varchar2, private\_key varchar2, key\_password varchar2, public\_key varchar2) return varchar2**

**function SIGN\_AND\_ENCRYPT(message BLOB, private\_key varchar2, key\_password varchar2, public\_key varchar2) return BLOB**

OpenPGP signs and encrypts a field in one pass

**Parameters:**

*message* - message to be signed and encrypted

*private\_key* - absolute file path on the server to the private key or the key as ASCII armored text

*key\_password* - password that unlocks the private key

*public\_key* - absolute file path on the server to the public key or the key as ASCII armored text

#### **Result**

the signed and encrypted field

#### **Online example**

<https://www.didisoft.com/ora-pgp/tutorial/sign-and-encrypt-in-plsql/>

### **SIGN\_AND\_ENCRYPT\_BLOB**

```
function SIGN_AND_ENCRYPT_BLOB(message varchar2, private_key varchar2, key_password
varchar2, public_key varchar2, ascii_armor BOOLEAN) return BLOB
function SIGN_AND_ENCRYPT_BLOB(message BLOB, private_key varchar2, key_password
varchar2, public_key varchar2, ascii_armor BOOLEAN) return BLOB
```

OpenPGP signs and encrypts a field in one pass

#### **Parameters:**

*message* - message to be signed and encrypted

*private\_key* - absolute file path on the server to the private key or the key as ASCII armored text

*key\_password* - password that unlocks the private key

*public\_key* - absolute file path on the server to the public key or the key as ASCII armored text

*ascii\_armor* - if TRUE the output will be in ASCII armored format, if FALSE then in binary format

#### **Result**

the signed and encrypted field

#### **Online example**

<https://www.didisoft.com/ora-pgp/tutorial/sign-and-encrypt-in-plsql/>

### **SIGN\_AND\_ENCRYPT\_CLOB**

```
function SIGN_AND_ENCRYPT_CLOB(message varchar2, private_key varchar2, key_password
varchar2, public_key varchar2) return CLOB
function SIGN_AND_ENCRYPT_CLOB(message CLOB, private_key varchar2, key_password
varchar2, public_key varchar2) return CLOB
```

OpenPGP signs and encrypts a field in one pass

#### **Parameters:**

*message* - message to be signed and encrypted

*private\_key* - absolute file path on the server to the private key or the key as ASCII armored text

*key\_password* - password that unlocks the private key

*public\_key* - absolute file path on the server to the public key or the key as ASCII armored text

*ascii\_armor* - if TRUE the output will be in ASCII armored format, if FALSE then in binary format

#### **Result**

the signed and encrypted field in ASCII armored format

#### **Online example**

<https://www.didisoft.com/ora-pgp/tutorial/sign-and-encrypt-in-plsql/>

### **SIGN\_AND\_ENCRYPT\_RAW**

```
function SIGN_AND_ENCRYPT_RAW(message varchar2, private_key varchar2, key_password
varchar2, public_key varchar2, ascii_armor BOOLEAN) return RAW
function SIGN_AND_ENCRYPT_RAW(message RAW, private_key varchar2, key_password varchar2,
public_key varchar2, ascii_armor BOOLEAN) return RAW
```



OpenPGP signs and encrypts a field in one pass

**Parameters:**

*message* - message to be signed and encrypted  
*private\_key* - absolute file path on the server to the private key or the key as ASCII armored text  
*key\_password* - password that unlocks the private key  
*public\_key* - absolute file path on the server to the public key or the key as ASCII armored text  
*ascii\_armor* - if TRUE the output will be in ASCII armored format, if FALSE then in binary format

**Result**

the signed and encrypted field

**Online example**

<https://www.didisoft.com/ora-pgp/tutorial/sign-and-encrypt-in-plsql/>

## VERIFY\_ENCRYPTED

### VERIFY\_ENCRYPTED (VARCHAR2)

**VARCHAR2**

*function VERIFY\_ENCRYPTED(message varchar2, private\_key varchar2, privatekey\_password varchar2, public\_key varchar2, decrypted\_message out varchar2) return pls\_integer*

**BLOB**

*function VERIFY\_ENCRYPTED(message BLOB, private\_key varchar2, privatekey\_password varchar2, public\_key varchar2) return pls\_integer*

extracts and verifies the signature of an OpenPGP [signed and encrypted](#) field

**Parameters:**

*message* - signed and encrypted message to be verified and extracted  
*private\_key* - absolute file path on the server to the private key or the key as ASCII armored text  
*key\_password* - password that unlocks the private key  
*public\_key* - absolute file path on the server to the public key or the key as ASCII armored text  
OUT *decrypted\_message* - the extracted message

**Result**

**See:** [Signature verification constants](#)

**Online example**

<https://www.didisoft.com/ora-pgp/tutorial/decrypt-and-verify-in-plsql/>

### VERIFY\_ENCRYPTED\_BLOB

*function VERIFY\_ENCRYPTED\_BLOB(message BLOB, private\_key varchar2, privatekey\_password varchar2, public\_key varchar2) return pls\_integer*

*function VERIFY\_ENCRYPTED\_BLOB(message BLOB, private\_key BLOB, privatekey\_password varchar2, public\_key BLOB) return pls\_integer*

Verifies the signature of an OpenPGP [signed and encrypted](#) field

**Parameters:**

*message* - signed and encrypted message to be verified and extracted  
*private\_key* - absolute file path on the server to the private key or the key as ASCII armored text  
*key\_password* - password that unlocks the private key  
*public\_key* - absolute file path on the server to the public key or the key as ASCII armored text

## Result

See: [Signature verification constants](#)

## Online example

<https://www.didisoft.com/ora-pgp/tutorial/decrypt-and-verify-in-plsql/>

## VERIFY\_ENCRYPTED\_CLOB

```
function VERIFY_ENCRYPTED_CLOB(message CLOB, private_key varchar2, privatekey_password  
varchar2, public_key varchar2) return pls_integer
```

```
function VERIFY_ENCRYPTED_CLOB(message CLOB, private_key CLOB, privatekey_password varchar2,  
public_key CLOB) return pls_integer
```

Verifies the signature of an OpenPGP [signed and encrypted](#) field

### Parameters:

*message* - signed and encrypted message to be verified and extracted

*private\_key* - absolute file path on the server to the private key or the key as ASCII armored text

*key\_password* - password that unlocks the private key

*public\_key* - absolute file path on the server to the public key or the key as ASCII armored text

## Result

See: [Signature verification constants](#)

## Online example

<https://www.didisoft.com/ora-pgp/tutorial/decrypt-and-verify-in-plsql/>

## VERIFY\_ENCRYPTED\_RAW

```
function VERIFY_ENCRYPTED_RAW(message RAW, private_key varchar2, privatekey_password  
varchar2, public_key varchar2) return pls_integer
```

```
function VERIFY_ENCRYPTED_RAW(message RAW, private_key RAW, privatekey_password varchar2,  
public_key RAW) return pls_integer
```

Verifies the signature of an OpenPGP [signed and encrypted](#) field

### Parameters:

*message* - signed and encrypted message to be verified and extracted

*private\_key* - absolute file path on the server to the private key or the key as ASCII armored text

*key\_password* - password that unlocks the private key

*public\_key* - absolute file path on the server to the public key or the key as ASCII armored text

## Result

See: [Signature verification constants](#)

## Online example

<https://www.didisoft.com/ora-pgp/tutorial/decrypt-and-verify-in-plsql/>

## Analytical

### IS\_ENCRYPTED

```
function IS_ENCRYPTED(message varchar2) return pls_integer
```

```
function IS_ENCRYPTED(message BLOB) return pls_integer
```

checks is given field OpenPGP encrypted.

### Parameters:

*message* - field to be checked

#### Result

- 1 - if the data is encrypted with a public key
- 0 - if the data is NOT encrypted with a public key

### SIGNING\_KEY\_ID

**function SIGNING\_KEY\_ID(message varchar2) return varchar2**  
**function SIGNING\_KEY\_ID(message BLOB) return BLOB**

gets the hexadecimal signing Key ID of an OpenPGP signed data, or the signing Key ID of an OpenPGP key (public or private)

#### Parameters:

*message* - signed data or public key or private key

#### Result

the first hexadecimal signing Key ID

Other possible results are:

'[PUBKEY](#)' - if the data is encrypted with a key

'[SYMKEY](#)' - if the data is symmetrically encrypted with a password

'[NOKEY](#)' - unknown data

### IS\_SIGNED

**function IS\_SIGNED(message varchar2) return pls\_integer**  
**function IS\_SIGNED(message BLOB) return pls\_integer**

checks is given field OpenPGP [signed only](#)

#### Parameters:

*message* - field to be checked

#### Result

- 1 - if the data is signed only
- 0 - if the data is NOT signed only (probably [encrypted](#))

### ENCRYPTION\_KEY\_ID

**function ENCRYPTION\_KEY\_ID(message varchar2) return varchar2**  
**function ENCRYPTION\_KEY\_ID(message BLOB) return BLOB**

gets the hexadecimal encryption Key ID of an OpenPGP encrypted data, or the encryption Key ID of an OpenPGP key (public or private)

#### Parameters:

*message* - encrypted data or signed and encrypted data or public key or private key

#### Result

the first hexadecimal encryption Key ID

Other possible results are:

'[ANYKEY](#)' - if the data is encrypted with a wildcard (hidden) key, in this case all possible keys must be tried for decryption

'[SYMKEY](#)' - if the data is symmetrically encrypted with a password

'[NOKEY](#)' - if the data is signed only

## Misc

### IS\_TRIAL\_VERSION

**function IS\_TRIAL\_VERSION return pls\_integer**

#### Returns

1 - if this is a Trial (evaluation) copy of ORA\_PGP  
0 - if this is a licensed production copy of ORA\_PGP

### VERSION

**function VERSION() return VARCHAR2**

#### Returns

current version of ORA\_PGP

#### Example:

```
select ORA_PGP.VERSION from dual
```

### SET\_ASCII\_VERSION\_HEADER

**procedure SET\_ASCII\_VERSION\_HEADER(version Varchar2)**

Allows customization of the OpenPGP Version header field in the ASCII armored output format

#### Example:

```
ORA_PGP.SET_ASCII_VERSION_HEADER('My Oracle encryption stored procedure 1.0');
```

### GET\_ASCII\_VERSION\_HEADER

**function GET\_ASCII\_VERSION\_HEADER() return Varchar2**

Returns the current OpenPGP Version header field in the ASCII armored output format

#### Example:

```
DBMS_OUTPUT.PUT_LINE (ORA_PGP.GET_ASCII_VERSION_HEADER());
```

### SET\_INTEGRITY\_PROTECT

**procedure SET\_INTEGRITY\_PROTECT(protect BOOLEAN)**

Turns On/Off inclusion of the Modification Detection Code packet (integrity protection) in encrypted and signed and encrypted messages.

This is in relation to the EFAIL attack and recent changes in GnuPG/gpg which as of version 2.2.8 rejects messages without Integrity protection.

#### Example:

```
ORA_PGP.SET_INTEGRITY_PROTECT(True);  
ORA_PGP.ENCRYPT... -- the output will have modification protection embedded
```



## Exception handling

---

### Purpose

As of version 1.1 the package provides custom mechanism for exception cause identification. It is useful for proper identification of unexpected results and gives options for taking recovery actions.

### Exception handling details

Upon error the `ORA_PGP` package functions throw exception of type `ORA_PGP.PGP_EXCEPTION`.

In order to identify the exact cause of the error the helper function `ORA_PGP.GET_PGP_ERROR` can be used as shown below:

```
BEGIN
    ORA_PGP.<method>( ...
EXCEPTION
    WHEN ORA_PGP.PGP_EXCEPTION THEN
        BEGIN
            IF ORA_PGP.GET_PGP_ERROR() = ORA_PGP.PGP_WRONG_PASSWORD_ERR THEN
                DBMS_OUTPUT.PUT_LINE('The password for the private key is not
matching: ' || SQLERRM);
            ELSIF ORA_PGP.GET_PGP_ERROR() = ORA_PGP.PGP_WRONG_PRIVATE_KEY_ERR THEN
                DBMS_OUTPUT.PUT_LINE('The provided key is not a valid OpenPGP private
key. ');
            ELSIF ORA_PGP.GET_PGP_ERROR() = ORA_PGP.PGP_NO_PUBLIC_KEY_ERR THEN
                DBMS_OUTPUT.PUT_LINE('The provided key is not a valid OpenPGP public
key. ');
            ELSIF ORA_PGP.GET_PGP_ERROR() = ORA_PGP.PGP_DETACHED_SIGNATURE_ERR THEN
                DBMS_OUTPUT.PUT_LINE('expected OpenPGP signed content passed for
verification is a detached OpenPGP signature. ');
            ELSIF ORA_PGP.GET_PGP_ERROR() = ORA_PGP.PGP_DATA_ENCRYPTED_ERR THEN
                DBMS_OUTPUT.PUT_LINE('expected OpenPGP signed content passed for
verification is an encrypted OpenPGP message ');
            ELSIF ORA_PGP.GET_PGP_ERROR() = ORA_PGP.PGP_PASS_ENCRYPTED_ERR THEN
                DBMS_OUTPUT.PUT_LINE('expected key encrypted OpenPGP message is a
password encrypted OpenPGP message ');
            ELSIF ORA_PGP.GET_PGP_ERROR() = ORA_PGP.PGP_INTEGRITY_CHECK_ERR THEN
                DBMS_OUTPUT.PUT_LINE('OpenPGP message integrity check failed ');
            ELSIF ORA_PGP.GET_PGP_ERROR() = ORA_PGP.PGP_NO_PGP_DATA_ERR THEN
                DBMS_OUTPUT.PUT_LINE('expected input OpenPGP message is either
corrupted or not an OpenPGP message at all ');
            ELSIF ORA_PGP.GET_PGP_ERROR() = ORA_PGP.PGP_KEY_EXPIRED_ERR THEN
                DBMS_OUTPUT.PUT_LINE('public key passed for encryption has expired ');
            ELSIF ORA_PGP.GET_PGP_ERROR() = ORA_PGP.PGP_KEY_REVOKED_ERR THEN
                DBMS_OUTPUT.PUT_LINE('public key passed for encryption has been
revoked ');
            ELSIF ORA_PGP.GET_PGP_ERROR() = ORA_PGP.PGP_GENERAL_PGP_ERR THEN
                DBMS_OUTPUT.PUT_LINE('General OpenPGP error: ' || SQLERRM);
            ELSIF ORA_PGP.GET_PGP_ERROR() = ORA_PGP.PGP_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](mailto:support@didisoft.com).

### Sales

For questions related to sales, volume licensing, or OEM licensing, please contact us at [sales@didisoft.com](mailto: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](http://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.