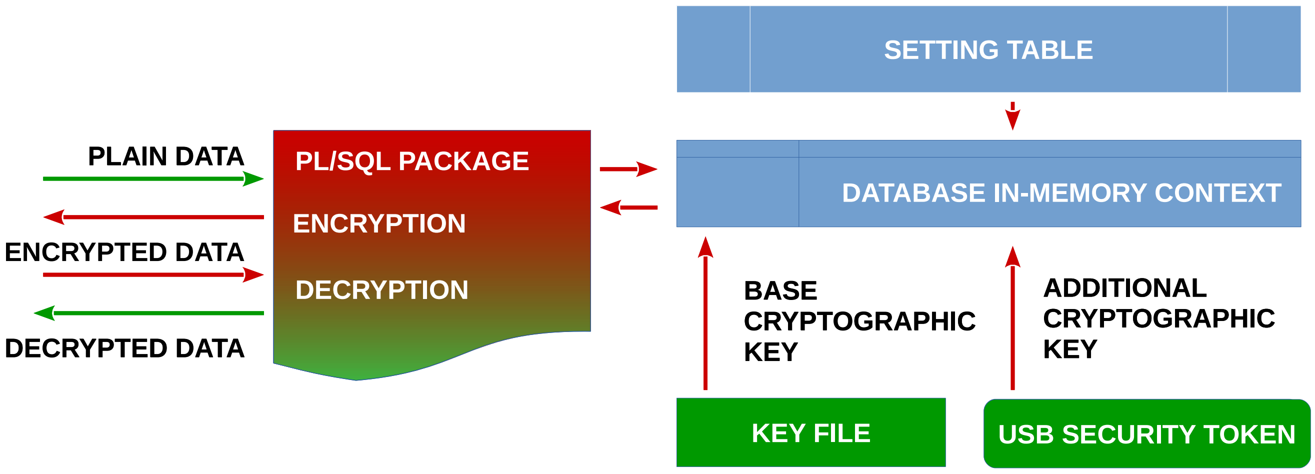
FCBCrypto software Windows installation and user guide

Table of contents

CONCEPT



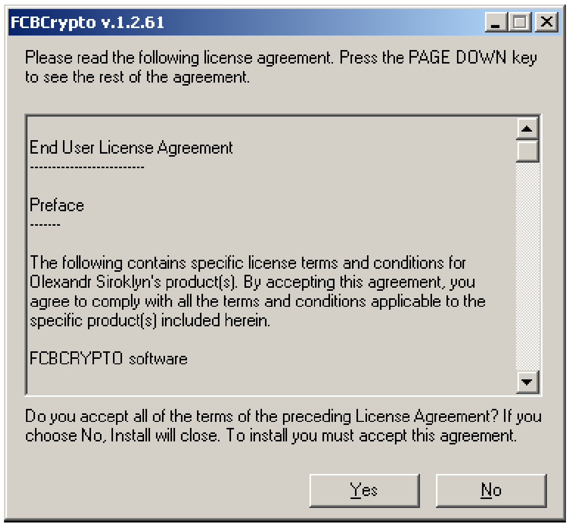
Database in-memory context is filled by cryptographic related information from a pre-defined table, an operating system key file and an USB security token (Linux only). Data go through the PL/SQL package to be encrypted or decrypted. Before encryption or decryption actions the PL/SQL package reads cryptographic information from the database in-memory context. Data are encrypted or decrypted and returned back.

BEGINING

FCBCRYPTO software is provided as a binary self-extracting archive file. Its name looks like **fcbcrypto-1.2.40.exe** where 1 is a version number, 2 is a subversion number and 40 is a build number.

UNPACKING

Please run fcbcrypto-*.exe file. End User License Agreement Window appears:


FCBCrypto v.1.2.61

Please read the following license agreement. Press the PAGE DOWN key to see the rest of the agreement.

End User License Agreement

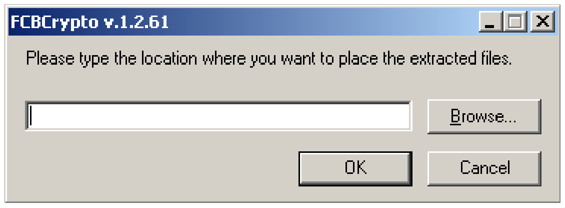
Preface

The following contains specific license terms and conditions for Olexandr Siroklyn's product(s). By accepting this agreement, you agree to comply with all the terms and conditions applicable to the specific product(s) included herein.

FCBCRYPTO software

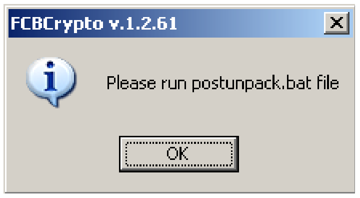
Do you accept all of the terms of the preceding License Agreement? If you choose No, Install will close. To install you must accept this agreement.

You must accept End User License Agreement to continue. Next dialog window proposes to choose a catalog to unpack software:


FCBCrypto v.1.2.61

Please type the location where you want to place the extracted files.

And the last unpacking step is a request to run postunpack.bat file.


FCBCrypto v.1.2.61

Please run postunpack.bat file

postunpack.bat file simply re-arranges unpacked files to sub-folders. After all you should get files and directories are similar to

```
Z:\2>dir

02/13/2018  12:43 PM    <DIR>          dat
02/13/2018  12:43 PM    <DIR>          doc
02/13/2018  12:41 PM               389 postunpack.bat
02/13/2018  12:43 PM    <DIR>          sql
```

```
Z:\2>dir doc

02/13/2018  12:41 PM           178,235 FCBCryptoWindowsInstallationAndUserGuide.pdf
02/13/2018  12:41 PM            58,430 FCBCryptoDataSheet.pdf
02/13/2018  12:41 PM           222,042 FCBCryptoUNIXInstallationAndUserGuide.pdf
02/13/2018  12:41 PM            39,059 FCBCryptoLicense.pdf
```

```
3Z:\2>dir sql

02/13/2018  12:41 PM               260 cre_PrimaryObjects.sql
02/13/2018  12:41 PM             3,500 cre_tbl_fcbcrypto_setting.sql
02/13/2018  12:41 PM               709 cre_fnc_fcbcrypto_hashtype.sql
02/13/2018  12:41 PM               155 cre_lib_fcbcrypto.sql
02/13/2018  12:41 PM               315 cre_jsr_fcbcrypto.sql.win
02/13/2018  12:41 PM               348 cre_jsr_fcbcrypto_feedback.sql.win
02/13/2018  12:41 PM             2,770 cre_pkg_fcbcrypto.pks
02/13/2018  12:41 PM            11,830 cre_pkg_fcbcrypto.pkb
02/13/2018  12:41 PM               276 cre_DemoAndTestsObjects.sql
02/13/2018  12:41 PM             1,201 step_5_test.sql
02/13/2018  12:41 PM            16,719 step_7_test.sql
02/13/2018  12:41 PM             6,543 cre_pkg_fcbcrypto_demo.pck
02/13/2018  12:41 PM             1,467 step_8_test.sql
02/13/2018  12:41 PM             6,849 step_10_test.sql
02/13/2018  12:41 PM             5,680 step_11_test.sql
02/13/2018  12:41 PM               956 step_12_test.sql
02/13/2018  12:41 PM               346 step_14_test.sql
02/13/2018  12:41 PM             1,058 step_15_test.sql
02/13/2018  12:41 PM               734 step_17_test.sql
02/13/2018  12:41 PM             6,405 step_18_test.sql
```

PREREQUISITES

There are prerequisites to start installation

a) Oracle database 11g or 12c any edition excepting Express Edition[1]
b) %ORACLE_HOME% environment variable
c) Oracle listener service
d) %ORACLE_HOME%\bin\sqlplus utility
e) database user privileges
f) Oracle Database Java subsystem[2]

1 Oracle Database 11g Express Edition
2 a standard database cross-version and cross-edition component or How to Add the JVM Component to an Existing

b) **%ORACLE_HOME%** environment variable must be setup

c) **Oracle listener service** must be setup and started.

d) **%ORACLE_HOME%\bin\sqlplus**[3] utility must be available via **%PATH%** environment variable. **%ORACLE_HOME%\bin\sqlplus** is used in FCBCRYPTO software installation process to run SQL scripts.

e) A database user where FCBCRYPTO software is going to be installed must have following grants

- create any context
- drop any context
- create library
- create table
- create procedure
- create sequence
- create trigger
- create type
- create session
- alter session
- select any dictionary
- read/write on **CUSTOM_KEY_DIR**[5] where key file will be stored
- execute on **SYS.DBMS_CRYPTO**[4]
- execute on **SYS.UTL_COMPRESS**[5]

f) Installed and valid, otherwise [6]

DATA ENCRYPTION

FCBCRYPTO software provides [4]AES(128, 192, 256)[7] encryption technique for char, nchar, varchar2, nvarchar2, string, blob, clob, nclob, raw, long, long raw, data types and modified OTP[8] encryption technique for number, float, date data types. Any timestamp or binary number data types aren't supported. AES(128, 192, 256) means a 16, 24 or 32 symbol base cryptographic key must be used. FCBCRYPTO software uses the base cryptographic key as a common key as for AES so for OTP technique. A base cryptographic you must have to perform data encryption/decryption.

Key file and key file directory must be presented at the **TBL_FCBCRYPTO_SETTING** table (see **SETTINGS** chapter). Those values reach the table via sql\cre_tbl_fcbcrypto_setting.sql file's launch during installation process (see **INSTALLATION** chapter). That means you should define an operating system key file name, an Oracle directory name for the key file directory and a base cryptographic key before installation, i.e.

- define a no white space 16 symbol base cryptographic key
- defile a key file directory for example c:\tmp

3 you aren't limited and you can use any GUI based Oracle SQL aware program to run SQL scripts
4 DBMS_CRYPTO
5 UTL_COMPRESS
6 Oracle Database (Doc ID 1461562.1)
7 Advanced Encryption Standard
8 One-time pad

- create a key file for example c:\tmp\key.txt
- insert the base cryptographic key into the key file
- save the key file
- define under which Oracle database owner you install FCBCRYPTO software
- run SQL*Plus utility %ORACLE_HOME%\bin\sqlplus.exe "/ as sysdba"
- [5]SQL> create directory CUSTOM_KEY_DIR as c:\tmp
- SQL> grant read, write on directory CUSTOM_KEY_DIR to FCBCRYPTO-software-owner
- open to edit sql\cre_PrimaryObjects.sql file
- adjust cre_tbl_fcbcrypto_setting.sql CUSTOMER_KEY_DIR key.txt string

INSTALLATION

It's highly recommended to perform installation under Oracle binary owner operating system account, i.e. if Oracle binaries' installation was performed under Administrator account, Oracle services start under Administrator account, please use Administrator account to install FCBCRYPTO software.
There are installation steps:

```
Z:\cd directory-where-FCBCRYPTO-software-was-unpacked\sql

Z:\%ORACLE_HOME%\bin\sqlplus.exe[9] owner/password @cre_PrimaryObjects.sql
```

And an output in case of success

```
Library created.

Java created.

Java created.

Package created.

No errors.

Package body created.

No errors.

Package body altered.
```

And if you want to install and run demos and tests

```
Z:\xcopy directory-where-FCBCRYPTO-software-was-unpacked\dat\* CUSTOM_KEY_DIR[5]

Z:\%ORACLE_HOME%\bin\sqlplus.exe owner/password @cre_DemoAndTestsObjects.sql
```

9 you aren't limited and you can use any GUI based Oracle SQL aware program to run SQL scripts

SETTINGS

FCBCRYPTO software has settings. All of them get values during FCBCRYPTO software installation process, but after process' finish you can adjust most of them. All settings are placed at **TBL_FCBCRYPTO_SETTING** table. Let's see:

S_ACTIVE	S_NAME	S_VALUE	S_VALUE_2	S_DESCRIPTION
Y	g_encryption_type	4358	16[4]	SYS.DBMS_CRYPTO.ENCRYPT_AES128 + SYS.DBMS_CRYPTO.CHAIN_CBC + SYS.DBMS_CRYPTO.PAD_PKCS5
N	g_encryption_type	4359	24[4]	SYS.DBMS_CRYPTO.ENCRYPT_AES192 + SYS.DBMS_CRYPTO.CHAIN_CBC + SYS.DBMS_CRYPTO.PAD_PKCS5
N	g_encryption_type	4360	32[4]	SYS.DBMS_CRYPTO.ENCRYPT_AES256 + SYS.DBMS_CRYPTO.CHAIN_CBC + SYS.DBMS_CRYPTO.PAD_PKCS5
Y	g_keyfile_dir	DATA_PUMP_DIR		directory (see ALL_DIRECTORIES view) where keyfile is placed
Y	g_keyfile	key.txt		keyfile name
Y	g_os	Windows		local operating system name
Y	g_context	global		global (SGA based) or local (PGA based) context access
Y	g_str_deflation_ratio	6		0 deflation is off, 1-fast ...9-best is on for varchar2 and nvarchar2 data type
Y	g_raw_deflation_ratio	6		0 deflation is off,

S_ACTIVE	S_NAME	S_VALUE	S_VALUE_2	S_DESCRIPTION
				1-fast ...9-best is on for raw data type
Y	g_blob_deflation_ratio	6		0 deflation is off, 1-fast ...9-best is on for blob data type
Y	g_clob_deflation_ratio	6		0 deflation is off, 1-fast ...9-best is on for clob data type
Y	g_service_function_user_list	SYSTEM		user list who authorized are to run service routines: prc_init, prc_deinit, fnc_usb_ste_present
Y	g_encrypted_context_base_key			Encrypted base key is stored in context

g_encryption_type is a kind of AES encryption to use. De-/activate any of them via “Y” or “N” value in the S_ACTIVE field. Only one “Y” value must be set. S_VALUE_2 is the size[4] of a base cryptographic key. Please change neither S_VALUE nor S_VALUE_2 by hands.

g_keyfile_dir is a **CUSTOM_KEY_DIRECTORY**[5] directory from DBA_DIRECTORIES view where the base cryptographic key file is placed. S_ACTIVE value is always “Y”.

g_keyfile. S_VALUE is a Windows file name where the base cryptographic key is stored. S_ACTIVE value is always “Y”.

g_os is an operating system common name. Please change in case of migration to the other OS only. S_ACTIVE value is always “Y”.

g_context can have global (SGA based) or local, i.e. session only (PGA based) context access[10] value. Default is global. Change it if you need. S_ACTIVE value is always “Y”.

g_str_deflation_ratio, g_raw_deflation_ratio, g_blob_deflation_ratio, g_clob_deflation_ratio is compression quality (for the correspondent data types) in the range from 0 to 9, i.e. 0 = no compression, 1 = fast compression, 9 = best compression. Default value is 6. Change it if you need. S_ACTIVE value is always “Y”. See **DATA COMPRESSION** chapter.

g_service_function_user_list[10] lists users who authorized are to run key management service routines: **PKG_FCBCRYPTO**.**PRC_INIT** procedure, **PKG_FCBCRYPTO**.**PRC_DEINIT** procedure(see more details in **KEY MANAGEMENT** chapter). Default value is a FCBCRYPTO software owner. You can add anyone or remove all of them. S_ACTIVE value is always “Y”.

g_encrypted_context_base_key defines to perform (S_ACTIVE = “Y”) or not to perform (S_ACTIVE = “N”) base cryptographic key context encryption, i.e. if base cryptographic key is Z3Wx&*&^%$#@CCZF in case of

10 context creation

S_ACTIVE = “N” it will be stored as Z3Wx&*&^%$#@CCZF value in context, in case of S_ACTIVE = “Y” it will be stored as 3445499CC20C7D39E2CDAB00BDEC84792503556AEC4A88E711E2913E40C457DA52DDD96F3E20E383AF54CBA79A985FCE value.

AFTER INSTALLATION

When installation finishes you have following objects in FCBCRYPTO software schema

- **PKG_ FCBCRYPTO** package[11]
- **FNC_ FCBCRYPTO_HASHTYPE**[12] function
- **LIB_ FCBCRYPTO** external library[13]
- **JSR_FCBCRYPTO** java source[14]
- **JSR_FCBCRYPTO_FEEDBACK** java source[15]
- **TBL_FCBCRYPTO_SETTING** table[16]

PKG_ FCBCRYPTO package is a main part of FCBCRYPTO software. **PKG_ FCBCRYPTO** package consists of two parts: a package specification and a wrapped package body.

Correspondent files[3]:

- sql\cre_pkg_fcbcrypto.pks
- sql\cre_pkg_fcbcrypto.pkb

PKG_ FCBCRYPTO package provides following encryption, decryption and service routines:

CHAR, VARCHAR2, STRING

- function fnc_encvchr (p_data in varchar2, l_custom_deflation in pls_integer default -1) return varchar2
- function fnc_decvchr (p_data in varchar2, l_custom_deflation in pls_integer default -1) return varchar2

NCHAR, NVARCHAR2

- function fnc_encnvch (p_data in nvarchar2, l_custom_deflation in pls_integer default -1) return nvarchar2
- function fnc_decnvch (p_data in nvarchar2, l_custom_deflation in pls_integer default -1) return nvarchar2

NUMBER, FLOAT (in -/+9.9*10^36 range)

- function fnc_encnum (l_value in number) return number
- function fnc_decnum (l_value in number) return number

DATE

11 **PKG_ FCBCRYPTO** package validity depends on all objects below
12 provides maximum available cryptographic hash algorithm for the current version of Oracle database
13 it exists but it is not used in current software release
14 it exists but it is not used in current software release
15 it exists but it is not used in current software release
16 see **SETTINGS** chapter for more details

- function fnc_encdate (l_value in date) return date
- function fnc_decdate (l_value in date) return date

BLOB

- function fnc_encblob (p_blob in blob, l_custom_deflation in pls_integer default -1) return blob
- function fnc_decblob (p_blob in blob, l_custom_deflation in pls_integer default -1) return blob

CLOB

- function fnc_encclob (p_clob in clob, l_custom_deflation in pls_integer default -1) return clob
- function fnc_decclob (p_clob in clob, l_custom_deflation in pls_integer default -1) return clob

NCLOB

- function fnc_encnclob (p_nclob in nclob, l_custom_deflation in pls_integer default -1) return nclob
- function fnc_decnclob (p_nclob in nclob, l_custom_deflation in pls_integer default -1) return nclob

RAW

- function fnc_encraw (p_raw in raw, l_custom_deflation in pls_integer default -1) return raw
- function fnc_decraw (p_raw in raw, l_custom_deflation in pls_integer default -1) return raw

LONG RAW

- function fnc_enclraw (p_lraw in long raw, l_custom_deflation in pls_integer default -1) return long raw
- function fnc_declraw (p_lraw in long raw, l_custom_deflation in pls_integer default -1) return long raw

LONG

- function fnc_enclong (p_long in long, l_custom_deflation in pls_integer default -1) return long
- function fnc_declong (p_long in long, l_custom_deflation in pls_integer default -1) return long

Key management routines. See KEY MANAGEMENT chapter

- procedure prc_init (l_in_key in varchar2 default null)
- procedure prc_deinit

“**enc**” in routine names stands for encryption. “**dec**” in routine names stands for decryption

l_custom_deflation input parameter means compression ratio in the range from 0=no compression, 1=fastest compression to 9=best compression. Omitting, i.e. using default -1 value, means correspondent g_*_deflation_ratio value from **TBL_FCBCRYPTO_SETTING** table (see **SETTINGS** chapter) will be used.

KEY MANAGEMENT

FCBCRYPTO software provides centralized in-memory key management process. Key management initialization is a mandatory first step to start data encryption/decryption[17]. The base cryptographic key or base cryptographic key traces aren't stored or presented in the constant database objects like tables. The base cryptographic key is distributed to the FCBCRYPTO decryption/encryption functions via Oracle database in-memory object only. That object is context (a set of application-defined attributes that validates and secures an application). Encryption/decryption can't be performed without in-memory loaded base cryptographic key. How it works in details.

Initialization

Database user having privileges to execute **PKG_FCBCRYPTO** package runs **PKG_FCBCRYPTO**.**PRC_INIT** procedure. **PKG_FCBCRYPTO**.**PRC_INIT** procedure accepts either

- a null input parameter and then procedure starts to read base cryptographic key from the key file (see g_keyfile in **TBL_FCBCRYPTO_SETTING** table) from the key file directory[5] (see g_keyfile_dir in **TBL_FCBCRYPTO_SETTING** table) or
- a 16, 24 or 32 symbol base cryptographic key as an input parameter[18]

```
SQL> exec pkg_fcbcrypto.prc_init;
or
SQL> exec pkg_fcbcrypto.prc_init('cHBmNz7cfTqH5t82VlXvjd8LdL45XccL');
```

If user is not in allowed-to-run-key-magement-routine-user list, i.e.

```
SQL> select S_VALUE as USERS_ALLOWED_TO_RUN_KEY_MANAGEMENT_ROUTINES
       from TBL_FCBCRYPTO_SETTING
      where S_NAME = ‘g_service_function_user_list’ and S_ACTIVE=’Y’
```

then **PKG_FCBCRYPTO**.**PRC_INIT** stops initialization[19] and a message

```
User is not authorized to run prc_init routine.
```

appears.

Otherwise **PKG_FCBCRYPTO** context is re-/created with a global or local access. Context access comes from

```
SQL> select S_VALUE as CONTEXT_ACCESS
       from TBL_FCBCRYPTO_SETTING
      where S_NAME = ‘g_context’ and S_ACTIVE=’Y’
```

After context creation context is filled out by the cryptographic related variables and values.

Here **PKG_FCBCRYPTO**.**PRC_INIT** procedure finishes key management initialization. This means all encryption/decryption **PKG_FCBCRYPTO** package routines (see **AFTER INSTALLATION** chapter) are ready to be used.

An example of successful key management initialization SQL*Plus output by **PKG_FCBCRYPTO**.**PRC_INIT** procedure

17 Encryption/decryption **PKG_FCBCRYPTO** package routines don't work without key management initialization
18 This means you don't need to worry key file (see g_keyfile in **TBL_FCBCRYPTO_SETTING** table) can be stolen
19 Why does that user list exist? Because you may have no intention to provide key management routine executable rights to the user already having executable grant on **PKG_FCBCRYPTO** package.

```
PKG_FCBCRYPTO context is being re|-created...
reading setting table and filling context...
reading key file...
filling context...
init done.

PL/SQL procedure successfully completed.
```

What if initialization was not performed or performed unsuccessfully and data encryption/decryption functions are called? An exception rises

```
ORA-06502: PL/SQL: numeric or value error
```

Please have a note **FCBCRYPTO software doesn’t monitor a base cryptographic key presence**, i.e. a presence of the key file (see g_keyfile in **TBL_FCBCRYPTO_SETTING** table) at the key file directory (see g_keyfile_dir in **TBL_FCBCRYPTO_SETTING** table), **in a real-time mode**. Also that means the key file is present or it is not, the key file directory is present or it is not FCBCRYPTO software, if it was initialized successfully once, knows nothing about missed key, missed catalog, i.e. in-memory context still can contain cryptographic information. Only launched **PKG_FCBCRYPTO.PRC_INIT** procedure can check missed stuff.

Deinitialization

PKG_FCBCRYPTO.PRC_DEINIT procedure simply erases previously filled cryptographic in-memory context information and delete in-memory context. After that any attempt to call **PKG_FCBCRYPTO** package encryption/decryption functions to encrypt/decrypt data gives only an error like

```
ORA-06502: PL/SQL: numeric or value error
```

An example of successful key management deinitialization SQL*Plus output by **PKG_FCBCRYPTO.PRC_DEINIT** procedure

```
PKG_FCBCRYPTO context de-initialized

PL/SQL procedure successfully completed.
```

If user is not in a allowed-to-run-key-magement-routine-user list, i.e.

```
SQL> select S_VALUE as USERS_ALLOWED_TO_RUN_KEY_MANAGEMENT_ROUTINES
       from TBL_FCBCRYPTO_SETTING
      where S_NAME = ‘g_service_function_user_list’ and S_ACTIVE=’Y’
```

then **PKG_FCBCRYPTO.PRC_DEINIT** procedure cancels deinitialization[20] and a message appears.

```
User is not authorized to run prc_deinit routine.
```

20 Why does that user list exist? Because you may have no intention to provide key management routine executable rights to the user already having executable grant on **PKG_FCBCRYPTO** package.

Key management summary

- base cryptographic key comes from the key file (see g_keyfile_dir in **TBL_FCBCRYPTO_SETTING** table) and the key file directory[5]
  (see g_keyfile_dir in **TBL_FCBCRYPTO_SETTING** table)
- base cryptographic key can come as an input parameter of **PKG_FCBCRYPTO.PRC_INIT** procedure call also
- base cryptographic key is stored in in-memory context after initialization
- base cryptographic keys get to the in-memory context via **PKG_FCBCRYPTO.PRC_INIT** procedure call
- base cryptographic key can be guaranteed erased from the memory by **PKG_FCBCRYPTO.PRC_DEINIT** procedure call, the in-memory context's deletion by database administrator or by the database reboot
- real cryptographic key is a derivative of the base cryptographic keys
- real cryptographic doesn't store in the in-memory context where the base cryptographic keys do
- real cryptographic key is calculated every time when data encryption/decryption **PKG_FCBCRYPTO** package routines are called
- data are encrypted and decrypted by the real cryptographic key only
- any keys aren't stored in the constant database objects like tables
- **PKG_FCBCRYPTO.PRC_INIT** procedure must be call only once if context was defined as global[21]
- **PKG_FCBCRYPTO.PRC_INIT** procedure must be call every time when session is created if context was defined as local
- Encryption/decryption is impossible without initial **PKG_FCBCRYPTO.PRC_INIT** procedure call

DATA COMPRESSION

FCBCRYPTO software provides a data compression option for char, nchar, varchar2, nvarchar2, string, blob, clob, nclob, raw, long, long raw SQL and PL/SQL data types. Compression availability is not a goal of FCBCRYPTO software, but a side effect. This effect got a life because of

- impossibility to forecast the size of encrypted data especially for *char* and string data types.
- obligatory conversion for the text contained data to ALT32UTF8 format before encryption

FCBCRYPTO software data compression bases on Oracle **UTL_COMPRESS** package. FCBCRYPTO software user may change compression quality in the range from 0 to 9, i.e. 0 = no compression, 1 = fast compression, 9 = best compression quality. Default value is 6 and it comes from

```
SQL> select S_NAME, S_VALUE
       from TBL_FCBCRYPTO_SETTING
      where S_NAME like 'g_%_deflation_ratio' and S_ACTIVE='Y';

S_NAME                   S_VALUE
----------------------  --------------
g_str_deflation_ratio   6
g_raw_deflation_ratio   6
```

21 see g_context in **TBL_FCBCRYPTO_SETTING** table

```
g_blob_deflation_ratio  6
g_clob_deflation_ratio  6
```

(see **SETTINGS** chapter). Please have a note compression could be ineffective in case of small size data. Also you can use either default ratio or customized ratio via **l_custom_deflation** input parameter of encryption/decryption **PKG_FCBCRYPTO** package routines (see **AFTER INSTALLATION** chapter). Customized ratio has higher priority over default **TBL_FCBCRYPTO_SETTING** table ratio. It's highly not recommended to use **l_custom_deflation** input parameter less than 5 when string field has length less than maximum, i.e. 4000 or 32767 characters, because it can lead to impossibility to store encrypted data

DEMOS, EXAMPLES AND TESTS

If you performed

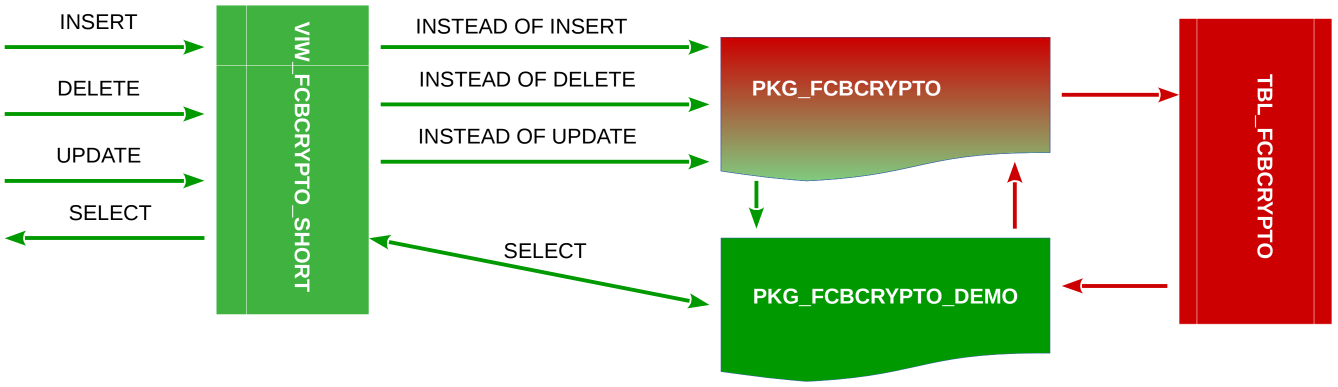
```
Z:\xcopy directory-where-FCBCRYPTO-software-was-unpacked\dat\* CUSTOM_KEY_DIR5

Z:\%ORACLE_HOME%\bin\sqlplus.exe owner/password @cre_DemoAndTestsObjects.sql
```

you have installed and valid Oracle database demo and test objects (triggers and sequences are not listed):

- **PKG_FCBCRYPTO_DEMO** package
- **VIW_FCBCRYPTO** view
- **VIW_FCBCRYPTO_SHORT** view
- **TBL_FCBCRYPTO** table
- **TBL_FCBCRYPTO_BLOB** table
- **GTT_FCBCRYPTO** global temporary table

So how demos and **PKG_FCBCRYPTO** package in general can be used?

Variant A.

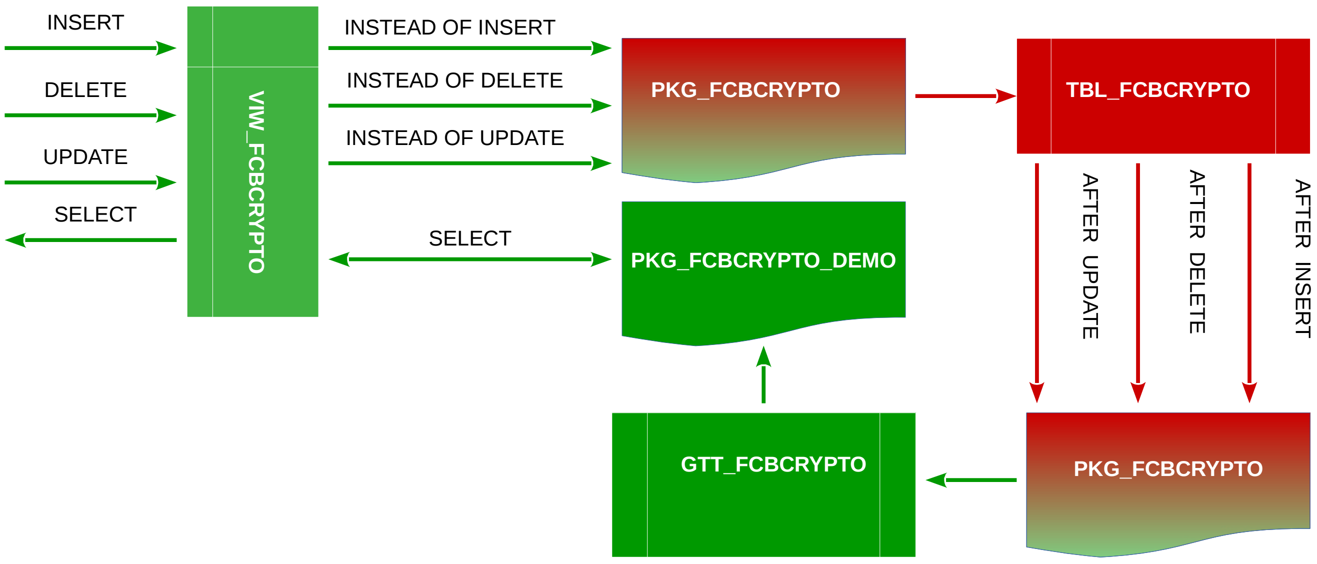


VIW_FCBCRYPTO_SHORT view is based on the statement

```
SQL> select *
       from table(PKG_FCBCRYPTO_DEMO.FNC_UNVEIL_BULK_SHORT(cursor(select * from
TBL_FCBCRYPTO)));
```

Plain data go to **VIW_FCBCRYPTO_SHORT** view. **VIW_FCBCRYPTO_SHORT** view contains three “instead of insert, update and delete” triggers. When insert, delete or update statements are performed, triggers catch data, encrypt data via **PKG_FCBCRYPTO** package and insert encrypted data into **TBL_FCBCRYPTO** table. **TBL_FCBCRYPTO** table stores encrypted data only. When select statement is performed from **VIW_FCBCRYPTO_SHORT** view a **PKG_FCBCRYPTO_DEMO.FNC_UNVEIL_BULK_SHORT** function requests encrypted data from **TBL_FCBCRYPTO** table, decrypts data via **PKG_FCBCRYPTO** package and return plain data to **VIW_FCBCRYPTO_SHORT** view.

Variant B.



VIW_FCBCRYPTO view is based on the statement

```
SQL>  select *
       from table(PKG_FCBCRYPTO_DEMO.FNC_UNVEIL_BULK(cursor(select * from
GTT_FCBCRYPTO)));
```

Plain data go to **VIW_FCBCRYPTO** view. **VIW_FCBCRYPTO** view contains three “instead of insert, update and delete” triggers. When insert, delete or update statements are performed, triggers catch data, encrypt data via **PKG_FCBCRYPTO** package and insert encrypted data into **TBL_FCBCRYPTO** table. **TBL_FCBCRYPTO** table stores encrypted data only. **TBL_FCBCRYPTO** table contains three “after insert, update and delete” triggers also. After data committing triggers call **PKG_FCBCRYPTO** package, **PKG_FCBCRYPTO** package decrypts data and plain decrypted data are inserted into global temporary **GTT_FCBCRYPTO** table. Why global temporary table? Because indexes can be created on it. **GTT_FCBCRYPTO** table contains session level only plain decrypted data. When select statement is performed from **VIW_FCBCRYPTO** view a **PKG_FCBCRYPTO_DEMO.FNC_UNVEIL** function requests data from **GTT_FCBCRYPTO** table and returns plain data to **VIW_FCBCRYPTO** view.

See more details in files

- sql/step_5_test.sql
- sql/cre_pkg_fcbcrypto_demo.pck
- sql/step_7_test.sql
- sql/step_8_test.sql

- sql/step_10_test.sql
- sql/step_11_test.sql
- sql/step_12_test.sql
- sql/step_14_test.sql
- sql/step_15_test.sql
- sql/step_17_test.sql
- sql/step_18_test.sql

VERSION

```
Z:\%ORACLE_HOME%\bin\sqlplus.exe ******/******
...
SQL> set serveroutput on
SQL> exec pkg_fcbcrypto.prc_about;

FCBCrypto software v.1.2.131
Copyright (c) 2018, Olexandr Siroklyn. All rights reserved.
...
```

COPYRIGHTS

CONTACTS

Olexandr Siroklyn
+380505771900