Appendix C DBFS encryption solution

The *DBFS Encryption Solution* consists of three parts:

- three database tables store cryptographic keys for single-user and multi-user files.
- the ECMSDK user-session based trigger, and
- the ECMSDK Spatial Crypt package.

C.1 Cryptographic key storage for DBFS single-user and multi-user files

Three database tables are used to store catalogs for:

1. encryption and decryption keys for spatial files belonging to specific users (user-specific),
2. files requiring access from multiple users (multi-user files), and
3. cryptographic keys together with the users who should have access to these multi-user files.

These tables and the encrypted column definitions are only possible with the TDE provided in Oracle Database 10gR2. The Oracle wallet can be initialised by issuing the following command, which creates a wallet, creates a master key for the entire database, and opens the wallet.

```
ALTER SYSTEM SET encryption key identified by "hdgr57fnle39dncv";
```

Subsequently, after each table creation with an encrypted column specification will cause the TDE to create a separate key for each table.

C.1.1 Key storage for single-user spatial files

Table used to store the DBFS user's spatial file encryption and decryption keys.

```
1.  CREATE TABLE ecmsdk_UserSpecificFile_keys(
2.    ecmsdk_user_id   NUMBER,             -- CMSDK user id
3.    enc_key       VARCHAR2(100)  ENCRYPT USING 'AES128',  -- encryption key
4.    dec_key       VARCHAR2(100)  ENCRYPT USING 'AES128'   -- decryption key
5.  );
```

The administrator simply creates or remove single-user keys for the DBFS users by issuing the following SQL commands. For instance, to add and delete keys for all files owned by user Scott, identified by ECMSDK user id 55514, the SQL commands are:

```
1.  INSERT INTO ecmsdk_UserSpecificFile_keys (ecmsdk_user_id, enc_key, dec_key) VALUES
    (55514,'G#^*!JS^MN!468h6', 'G#^*!JS^MN!468h6');
2.  DELETE FROM ecmsdk_UserSpecificFile_keys WHERE ECMSDK_user_id = 55514;
```

The similarity of the keys stored in this table for specific users depend on whether symmetric or asymmetric algorithms are used. To retrieve the encryption keys of a specific ECMSDK user from this table, the get_enc_key() and get_dec_key() functions.

C.1.2 Temporary LOB location storage for single-user spatial files

Table for storing user temporary LOB locators used internally by ***ecmsdk_spatial_crypt*** package. The table is populated with the ECMSDK user's file temporary LOB locator within the temporary tablespace as soon as the file is decrypted. Upon user logout, the LOB locator is used to copy the LOB back into its original location.

```
1.  CREATE TABLE ecmsdk_user_temp_lobs(
2.    ECMSDK_user_id   NUMBER,             -- CMSDK user id
3.    ECMSDK_file_id     NUMBER,           -- User File id
4.    ECMSDK_file_name VARCHAR(100),       -- CMSDK file name
5.    templob       BLOB                   -- temporary LOB
6.  );
```

C.1.3 Multi-user, project-specific file key storage

A database table is used to store a file's filepath, its encryption and decryption keys, and the state of file that is checked before trying to re-encrypt or re-decrypt files preventing corruption.

```
1.  CREATE TABLE ecmsdk_MultiUserFile_keys_state (
2.    ecmsdk_filepath   VARCHAR2(500),        -- Full ECMSDK filepath
3.    enc_key        VARCHAR2(100) ENCRYPT USING 'AES128',  -- encryption key
4.    dec_key        VARCHAR2(100) ENCRYPT USING 'AES128',  -- decryption key
5.    file_encrypted VARCHAR2(5) DEFAULT 'FALSE' -- file state
6.    templob        BLOB               -- temporary LOB
7.  );
```

The administrator can issue the following commands to add or remove filepaths, the encrypted state, and the encryption and decryption keys of the file. For instance, SQL to add and delete spatial file **spottext.MAP** with full ECMSDK filepath **"projects\NewZealand\spottext.MAP"** are:

```
1.  INSERT INTO ecmsdk_MultiUserFile_keys_state (ecmsdk_filepath,file_state, enc_key,dec_key) VALUES
    ('projects\newzealand\spottext.MAP','hfghfdg657hvg^*T' ,'hfghfdg657hvg^*T');
2.  DELETE FROM ecmsdk_MultiUserFile_keys_state WHERE ecmsdk_filepath LIKE
    "projects\newzealand\spottext.MAP';
```

Once again, the similarity of the keys stored in this table for users depend on whether symmetric or asymmetric encryption algorithms are used. To retrieve the encryption keys of a specific ECMSDK user from this table, the get_file_enc_key() and get_file_dec_key() functions are used.

C.1.4 Multi-user, project specific user-file catalog

The following script describes the database table that stores the filepath and the different ECMSDK users who require access to them in decrypted form.

```
1.  CREATE TABLE ECMSDK_MultiUserFile_user_catalog(
2.      ecmsdk_filepath  VARCHAR2(500),    -- ECMSDK MultiUserFile Full filepath
3.      ecmsdk_user_id   NUMBER            -- ECMSDK user id
4.  );
```

Using the following SQL commands, the administrator can insert or delete files, ECMSDK users, and the files that should be decrypted for use by a particular user and encrypted back for storage. For instance, to add and delete cryptographic keys for all files owned by GIS user Scott, identified by ECMSDK user id 55514, the SQL commands are:

```
1.  INSERT INTO ecmsdk_MultiUserFile_user_catalog (ecmsdk_user_id, CMSDK _filepath) VALUES (55514,
    ‘projects\newzealand\spottext.MAP’);
2.  DELETE FROM ecmsdk_MultiUserFile_user_catalog WHERE ecmsdk_user_id = 55514 AND WHERE
    ecmsdk_filepath LIKE ‘projects\newzealand\spottext.MAP’;
```

C.2 Oracle ECMSDK session-based trigger

On each ECMSDK user session creation and termination, the following session-based trigger is used to call decryption and encryption procedures, respectively. Those two procedures first obtain the *single-user* as well as the *multi-user* GIS files to which a user has privilege, and then calls either the decryption or the encryption procedure.

To prevent the execution of this trigger when multiple users are using the same account (which would corrupt the file resulting from double encryption and decryption), the trigger first queries the number of existing sessions of the user; only if no existing session of the user exists, then the files are decrypted or encrypted.

```
1.  CREATE OR REPLACE TRIGGER dec_enc_spatial_files
2.    BEFORE INSERT OR DELETE ON ifssys.odmz_session
3.    FOR EACH ROW
4.    DECLARE
5.      v_sessions NUMBER;
6.      CURSOR active_CMSDK_user_sessions IS
7.        SELECT USERID FROM IFSSYS.ODMZ_SESSION
8.     WHERE USERID = (:NEW.USERID) OR USERID = (:OLD.USERID);
9.    BEGIN
10.   OPEN active_CMSDK_user_sessions;
11.   FETCH active_CMSDK_user_sessions INTO v_sessions;
12.   -- Only if no existing sessions of this user, then decrypt or encrypt
13.   IF active_CMSDK_user_sessions%NOTFOUND THEN
14.     IF INSERTING THEN
15.        CMSDK_spatial_crypt.get_11gcmsdk_user_files_dec (:NEW.USERID);
16.     ELSIF DELETING THEN
17.        CMSDK_spatial_crypt.get_11gcmsdk_user_files_enc (:OLD.USERID);
18.     END IF;
19.   END IF;
20.   CLOSE active_CMSDK_user_sessions;
21. END dec_enc_spatial_files;
22. /
```

C.3 Oracle ECMSDK Spatial Crypt package.

The package is included as a SQL file named ECMSDK_SpatialCrypt.sql in this repository accessible through:
https://github.com/sharmapn/DBFSFileCrypto/blob/main/ECMSDK_SpatialCrypt.sql.