Appendix A Encryption times for spatial files within DBFS

Table 1 shows encryption (*Enc.*) and decryption (*Dec.*) times of different sizes of spatial files using 256-bit AES encryption (32-character key) using three approaches:

- within ECMSDK using `DBMS_CRYPTO_TOOLKIT`,
- within ECMSDK using OpenSSL encryption tool, and
- within Windows NTFS using OpenSSL encryption tool.

The next three subsections include the encryption-decryption performance results from these approaches.

The first two approaches compare the encryption and decryption performance of files within ECMSDK using (a) the `DBMS_CRYPTO_TOOLKIT` within the Oracle 11g database (implemented in PL/SQL) against (b) the performance provided by the OpenSSL tool (a Windows executable, called externally using PL/SQL from our encryption scripts). The third approach compares the encryption of spatial files using the OpenSSL tool within ECMSDK versus those stored directly on the OS (i.e. Windows NTFS).

In all experiments, the tests were performed on a machine with the following specifications: an Intel 1.6 GHz i5-8265u processor with 8GB RAM on Dell Inspiron system and the software utilized consisted of Oracle 11g Database Release 2, running on Windows 10 Operating System.

GIS File	Filesize (Mb)	AES		OpenSSL on CMSDK		OpenSSL on NTFS	
		Enc.	Dec.	Enc.	Dec.	Enc.	Dec.
france-points.shp	1.50	1.13	1.07	0.48	0.17	0.04	0.04
france-waterways.shp	6.32	4.73	4.49	0.55	0.29	0.06	0.05
france-natural.shp	11.67	8.74	8.30	1.15	0.45	0.11	0.07
indonesia-natural.shp	11.95	8.95	8.50	1.20	0.55	0.07	0.07
germany-points.shp	13.89	10.41	9.88	1.25	0.92	0.07	0.14
britain-waterways.shp	16.12	12.08	11.46	1.93	1.31	0.08	0.09
china-buildings.shp	26.53	19.87	18.87	2.25	1.90	0.12	0.34
india-natural.shp	32.13	24.07	22.85	3.81	2.38	0.13	0.37
china-natural.shp	33.75	25.28	24.00	4.08	2.68	0.14	0.48
india-waterways.shp	36.21	27.13	25.75	4.18	2.77	0.15	0.56

Table 1: Comparing three approaches of applying AES encryption and decryption of spatial files[1].

[1] The ESRI shapefiles used in the investigation are sourced from publicly available datasets (https://mapcruzin.com/download-free-arcgis-shapefiles.htm).

A.1 Within ECMSDK using DBMS_CRYPTO_TOOLKIT

Figure A1 shows the encryption and decryption time for spatial files in ECMSDK being computed from within PL/SQL code using SQLplus. One by one, each file's ID is passed to the encryption procedure `encrypt()` along with the encryption key. The time capturing has been coded within PL/SQL code.

To retrieve the file id of a file, the following SQL can be used.

```
SELECT co.content, doc.name
FROM ecmsdk.odmv_document doc, ecmsdk.odm_contentobject co
WHERE doc.name LIKE '%.shp' AND doc.CONTENTOBJECT = co.id;
```

The id of files retrieved using the above query is used as parameter in the encryption and decryption processes:

```
set echo on
set serveroutput on

call encrypt(23402,'passwordpasswordpasswordpassword');
commit;
call decrypt(23402,'passwordpasswordpasswordpassword');
commit;
```

This PL/SQL encryption and decryption procedure is available within the "performanceResults\Approach1_WithinECMSDKUsingDBMSCrypto_PLSQL" folder in the GitHub project repository: https://github.com/sharmapn/DBFSFileCrypto/

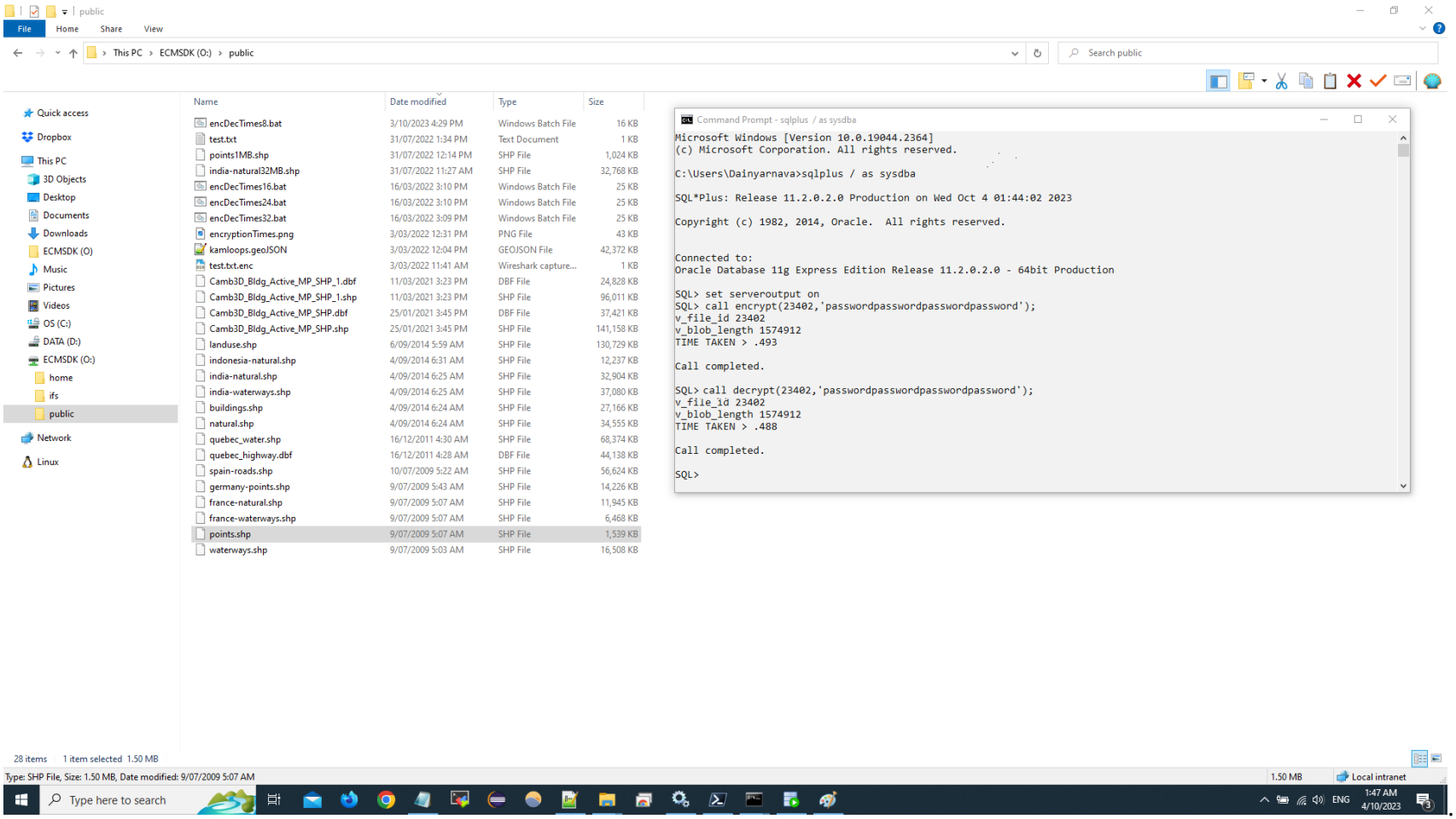


Figure A.1: Encryption and Decryption time for spatial files in ECMSDK being computed using SQLPlus.

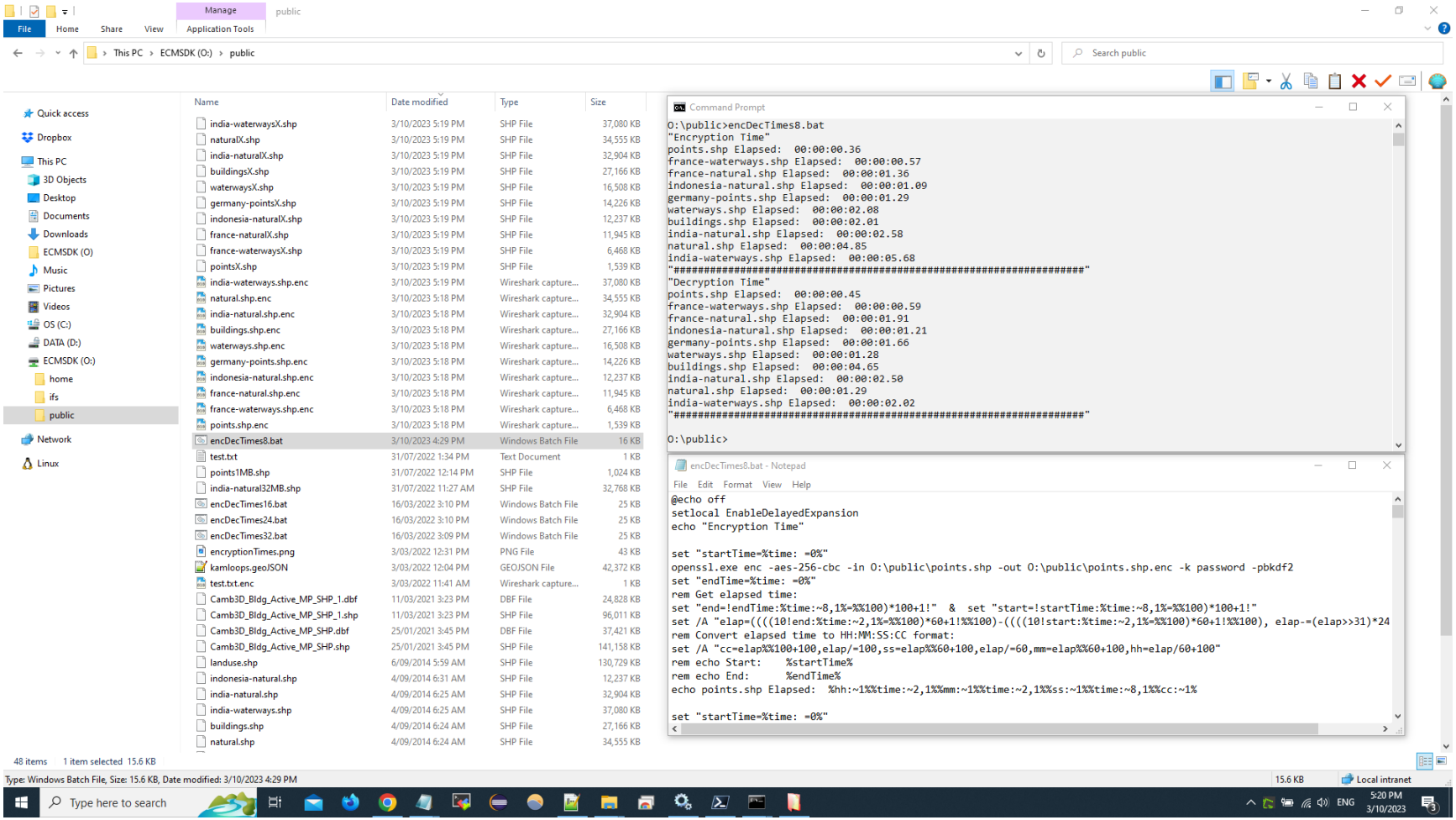
A.2 Within ECMSDK using OpenSSL Toolkit

Figure A2 shows the encryption and decryption time for spatial files in ECMSDK being computed using the OpenSSL tool. A batch script encrypts and then later decrypts each file one by one for different lengths of the encryption keys. The filename and the encryption key is passed as argument. The time capturing has been coded within the batch script.

Here is one encryption and decryption pair of code:

```
echo "Encryption Time"

set "startTime=%time: =0%"
openssl enc -aes-256-cbc -in O:\public\points.shp -out O:\public\points.shp.enc -k
password -pbkdf2
set "endTime=%time: =0%"
set "end=!endTime:%time:~8,1%=%%100)*100+1!"  &  set
"start=!startTime:%time:~8,1%=%%100)*100+1!"
set /A "elap=((((10!end:%time:~2,1%=%%100)*60+1!%%100)-
((((10!start:%time:~2,1%=%%100)*60+1!%%100), elap-=(elap>>31)*24*60*60*100"
set /A
"cc=elap%%100+100,elap/=100,ss=elap%%60+100,elap/=60,mm=elap%%60+100,hh=elap/60+100"
echo points.shp Elapsed:
%hh:~1%%time:~2,1%%mm:~1%%time:~2,1%%ss:~1%%time:~8,1%%cc:~1%;



echo "Decryption Time"

set "startTime=%time: =0%"
openssl enc -d -aes-256-cbc -in O:\public\points.shp.enc -out O:\public\pointsX.shp -k
password -pbkdf2
set "endTime=%time: =0%"
rem Get elapsed time:
set "end=!endTime:%time:~8,1%=%%100)*100+1!"  &  set
"start=!startTime:%time:~8,1%=%%100)*100+1!"
set /A "elap=((((10!end:%time:~2,1%=%%100)*60+1!%%100)-
((((10!start:%time:~2,1%=%%100)*60+1!%%100), elap-=(elap>>31)*24*60*60*100"
rem Convert elapsed time to HH:MM:SS:CC format:
set /A
"cc=elap%%100+100,elap/=100,ss=elap%%60+100,elap/=60,mm=elap%%60+100,hh=elap/60+100"
rem echo Start:    %startTime%
rem echo End:      %endTime%
echo points.shp Elapsed:
%hh:~1%%time:~2,1%%mm:~1%%time:~2,1%%ss:~1%%time:~8,1%%cc:~1%
```

The full batch scripts for the encryption and decryption procedures are available within the "performanceResults/Approach2_WithinECMSDKUsingOpenSSL" folder at the GitHub project repository: https://github.com/sharmapn/DBFSFileCrypto/



.

Figure A.2: Encryption and Decryption time for spatial files in ECMSDK using OpenSSL command line tool.

A.3 Within NTFS using OpenSSL Toolkit

Figure A3 shows the encryption and decryption time for spatial files in ECMSDK being computed from within PL/SQL code using SQLplus. One by one, each file's ID is passed to the encryption procedure `encrypt()` along with the encryption key. The time capturing has been coded within PL/SQL code.

To retrieve the file id of a file, the following SQL can be used.

```
echo "Encryption Time"

set "startTime=%time: =0%"
openssl enc -aes-256-cbc -in D:\OpenSSLencryptionTimes\points.shp -out
D:\OpenSSLencryptionTimes\points.shp.enc -k password -pbkdf2
set "endTime=%time: =0%"
rem Get elapsed time:
set "end=!endTime:%time:~8,1%=%%100)*100+1!"  &  set
"start=!startTime:%time:~8,1%=%%100)*100+1!"
set /A "elap=((((10!end:%time:~2,1%=%%100)*60+1!%%100)-
((((10!start:%time:~2,1%=%%100)*60+1!%%100), elap-=(elap>>31)*24*60*60*100"
rem Convert elapsed time to HH:MM:SS:CC format:
set /A
"cc=elap%%100+100,elap/=100,ss=elap%%60+100,elap/=60,mm=elap%%60+100,hh=elap/60+100"
rem echo Start:    %startTime%
rem echo End:      %endTime%
echo points.shp Elapsed:
%hh:~1%%time:~2,1%%mm:~1%%time:~2,1%%ss:~1%%time:~8,1%%cc:~1%;

echo "Decryption Time"

set "startTime=%time: =0%"
openssl enc -d -aes-256-cbc -in D:\OpenSSLencryptionTimes\points.shp.enc -out
D:\OpenSSLencryptionTimes\pointsX.shp -k password -pbkdf2
set "endTime=%time: =0%"
rem Get elapsed time:
set "end=!endTime:%time:~8,1%=%%100)*100+1!"  &  set
"start=!startTime:%time:~8,1%=%%100)*100+1!"
set /A "elap=((((10!end:%time:~2,1%=%%100)*60+1!%%100)-
((((10!start:%time:~2,1%=%%100)*60+1!%%100), elap-=(elap>>31)*24*60*60*100"
rem Convert elapsed time to HH:MM:SS:CC format:
set /A
"cc=elap%%100+100,elap/=100,ss=elap%%60+100,elap/=60,mm=elap%%60+100,hh=elap/60+100"
rem echo Start:    %startTime%
rem echo End:      %endTime%
echo points.shp Elapsed:
%hh:~1%%time:~2,1%%mm:~1%%time:~2,1%%ss:~1%%time:~8,1%%cc:~1%
```

The full batch scripts for the encryption and decryption procedures are available within the "performanceResults/Approach3_WithinNTFSUsingOpenSSL" folder at the GitHub project repository: https://github.com/sharmapn/DBFSFileCrypto/

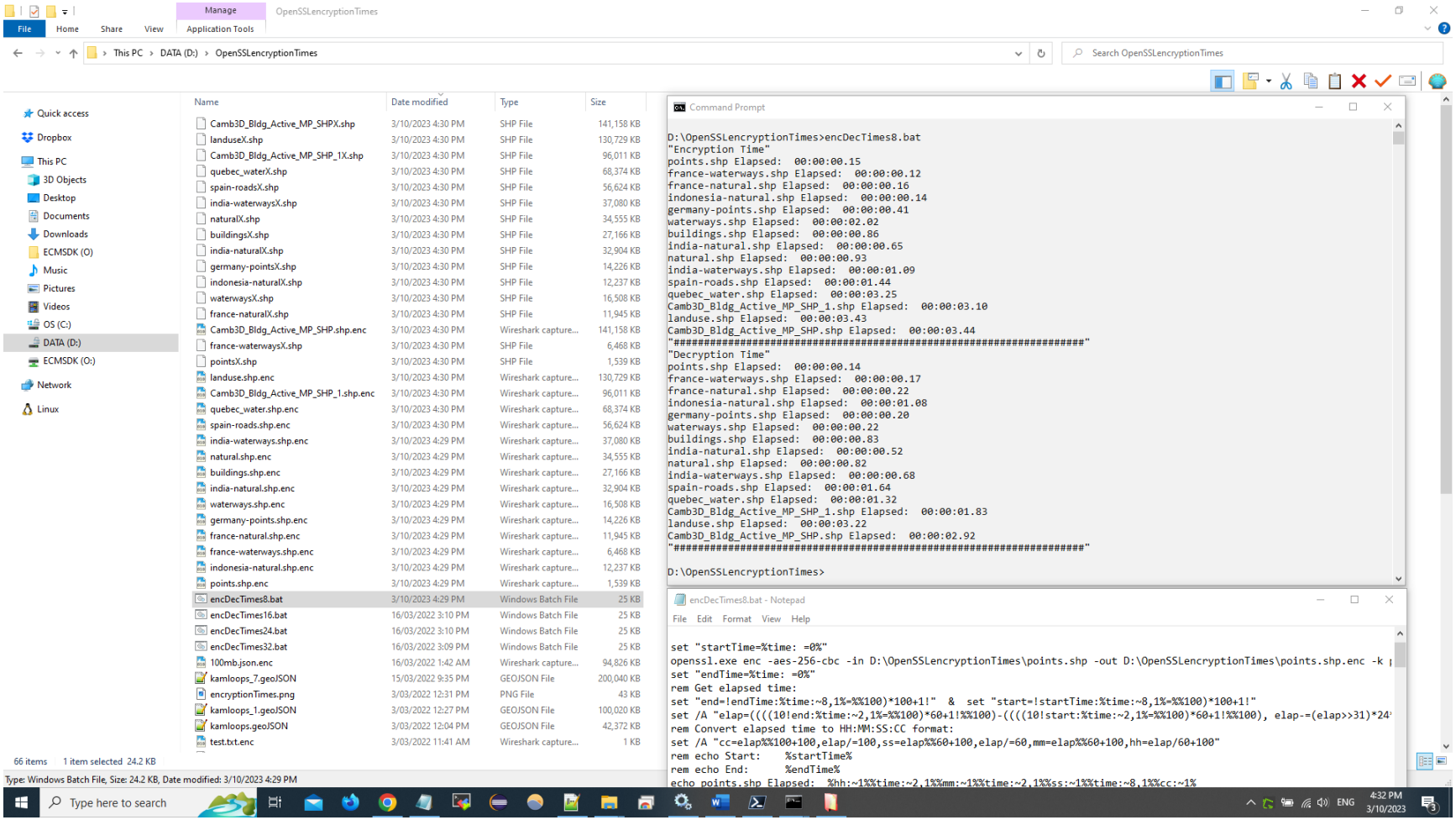


Figure A.3: Encryption and Decryption time for spatial files in Windows NTFS using OpenSSL command line tool.

Appendix B 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. /
```

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 Spatial Crypt package, and
- the ECMSDK user-session based trigger.

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’;
```

Appendix D Security solution use with classical shared folders

We demonstrate the user-interaction model with the proposed storage security solution in a classical GIS enterprise setup. Users *Scott* and *Alan*, log onto the ECMSDK server and access single-user files in their respective home folders and multi-user files placed in a project directory. These files must be decrypted for use and encrypted back for protection.

Single-user & Multi-user files and privileges: Scott's single-user files within his home folder are eight files (*cpg, dbf, prj, shp, shx, txt, xml and pdf*) of the ***site-of-significance*** shapefile dataset. For brevity, Alan does not have single-user files. There are two sets of multi-user shapefiles; ***nz-bounty-islands*** and ***nz-historic-places*** in the "`O:\projects\NewZealand\`" directory.

Scott should have access to the eight ***single-user*** files of ***site-of-significance*** shapefile dataset in his home folder and to the seven multi-user GIS files of the ***the nz-bounty-islands-polygons-topo-125k*** dataset. Alan should have access to all 14 multi-user files, but not to Scott's eight single user files. Table 2 details these requirements. These files require protection with encryption.

Single-user shapefiles	Privileges	
	Scott	Alan
site-of-significance (only 2 of 8 files need encryption) • site-of-significance.dbf • site-of-significance.shp	☑	
Multi-user shapefiles		
nz-bounty-islands-polygons-topo-125k (7 files)		☑
nz-historic-places (7 files)	☑	☑

Table 2. User privileges on single-user and multi-user shapefiles.

Protocol services: We demonstrate the security solution using network folder sharing via the ECMSDK SMB server. In enterprise GIS, this is the most common mode of accessing spatial data; clients access files on the server through mapped network shares and interact with files through GIS applications on the client's computer. However, working with the security solution using these other protocol servers (e.g., FTP, HTTP, Command line, etc.,) provided by ECMSDK product will be similar, since the DBFS enforces access control at the repository level and the execution of the encryption-decryption procedures depend on DBFS user sessions, that are created when the user uses any protocol service.

Begin with files already encrypted: For brevity, we begin from a point when the single-user and multi-user cryptographic keys for users Scott and Alan have been created and when Scott's single-user files and multi-user files are in encrypted form. Furthermore, the administrator has encrypted all eight multi-user files and user Scott has already specified that all the eight files (***cpg, dbf, prj, shp, shx, txt, xml and pdf***) belonging to the ***site-of-significance*** shapefile dataset need encryption by appending each the filenames with the `_enc` file suffix. In addition, we assume that Scott has subsequently logged out of DBFS upon which the files have been encrypted (see Fig. 1 and 2). Thereafter, we illustrate GIS users Scott and Alan logging to the ECMSDK. Subsequently, their single-user and multi-user shapefiles are decrypted for use, are available for multiple users during their DBFS sessions, and are later encrypted back for protection.

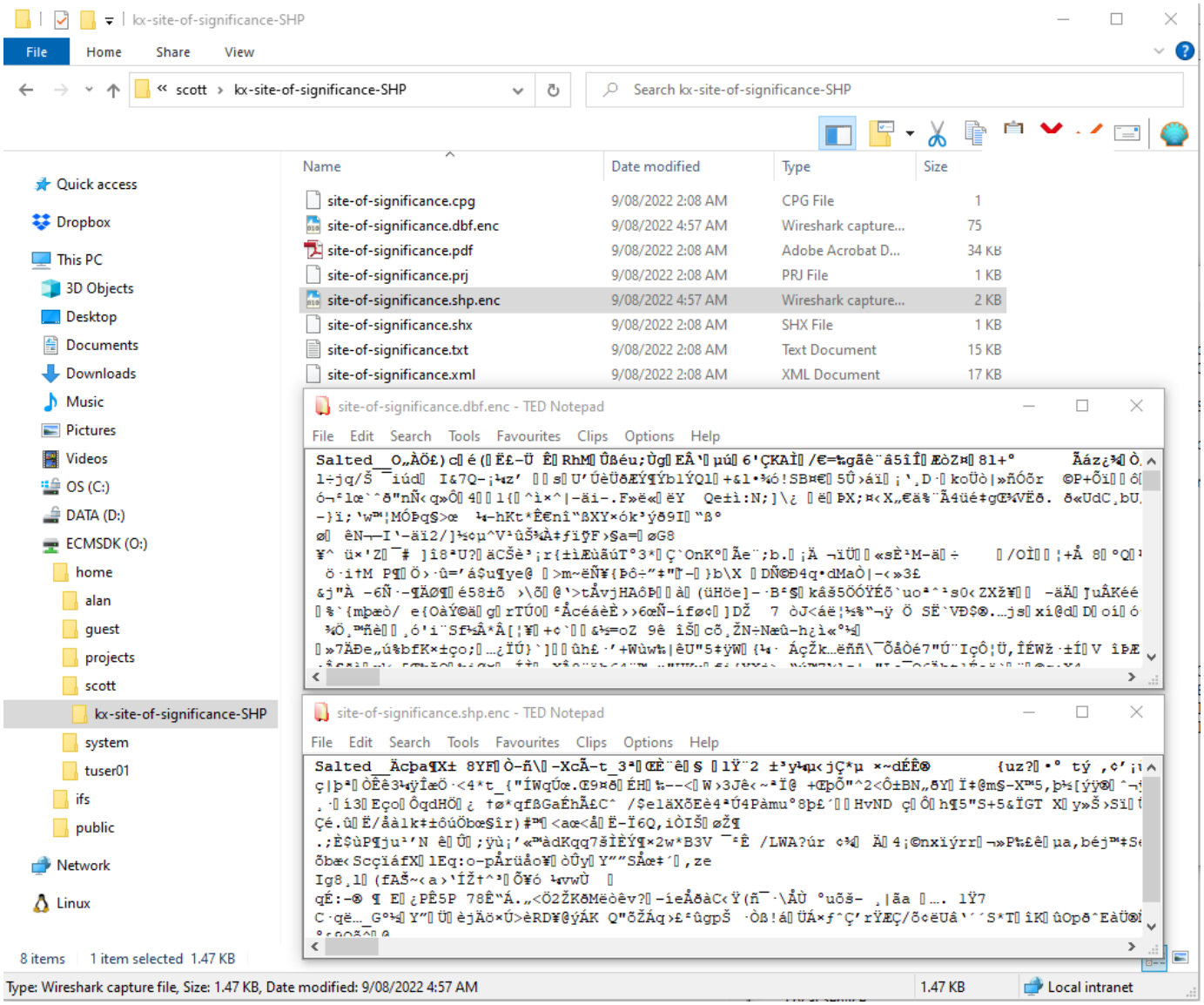


Figure 1: User Scott's single-user files (and multi-user files as shown in Fig 2.) are originally in encrypted form on the ECMSDK server.

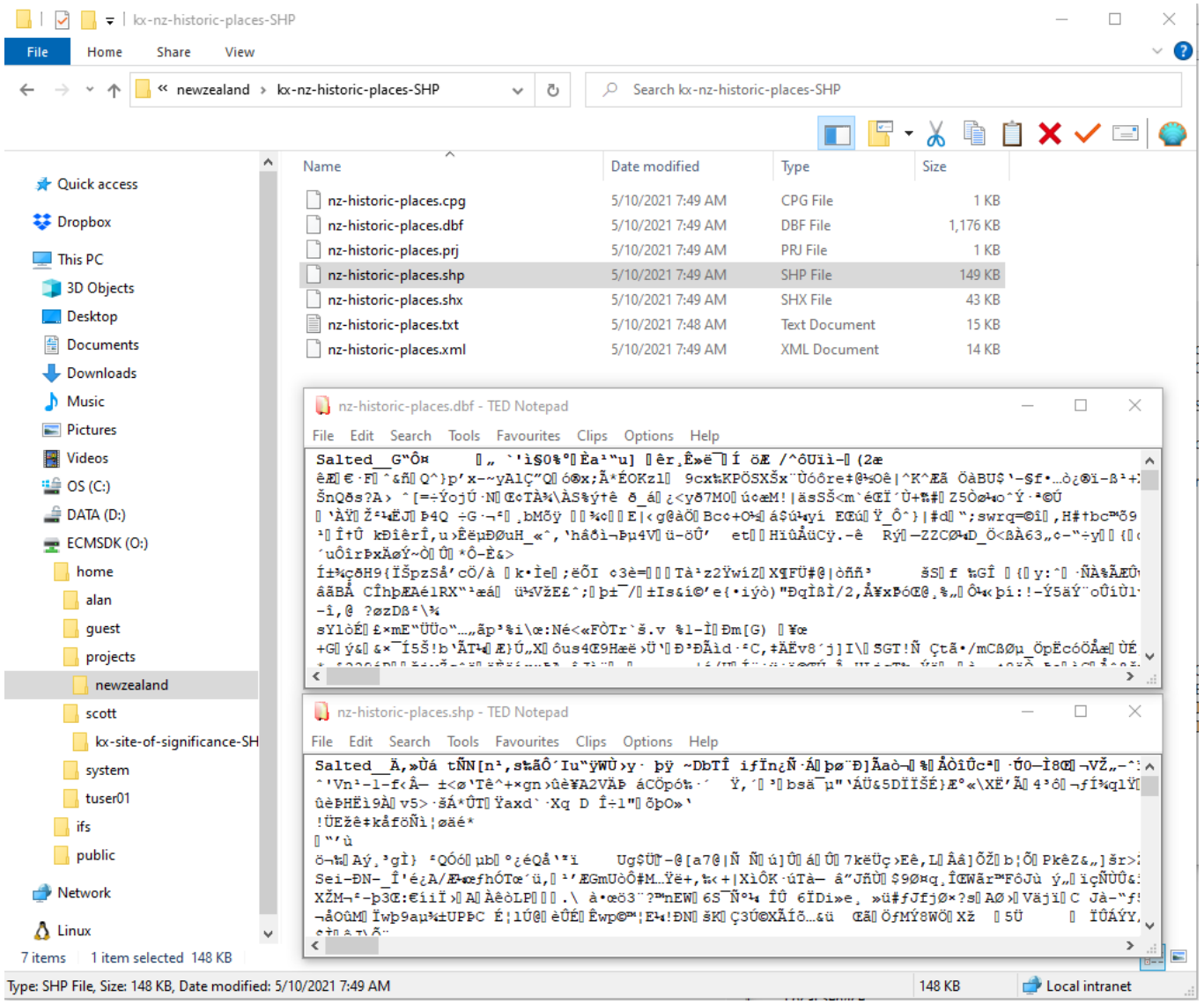


Figure 2: The seven multi-user GIS files to which Scott has privilege are also originally in encrypted form on the ECMSDK server.

Illustrating proposed solution using folder sharing and GIS application

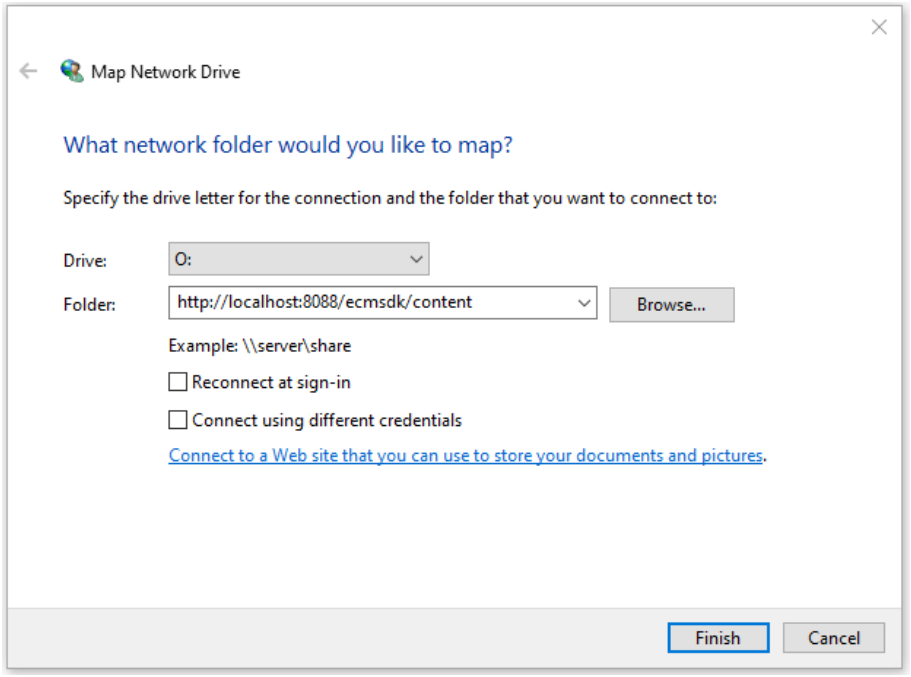


Figure 3: GIS user Scott chooses ECMSDK shared resource to map a network share.

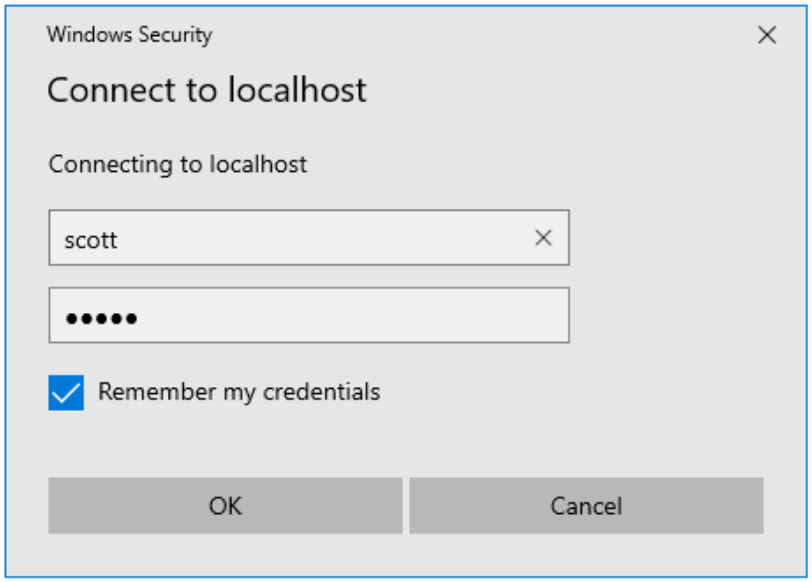


Figure 4: User "Scott" provides credentials for authentication to the ECMSDK protocol server; in this case, it is the ECMSDK SMB server.

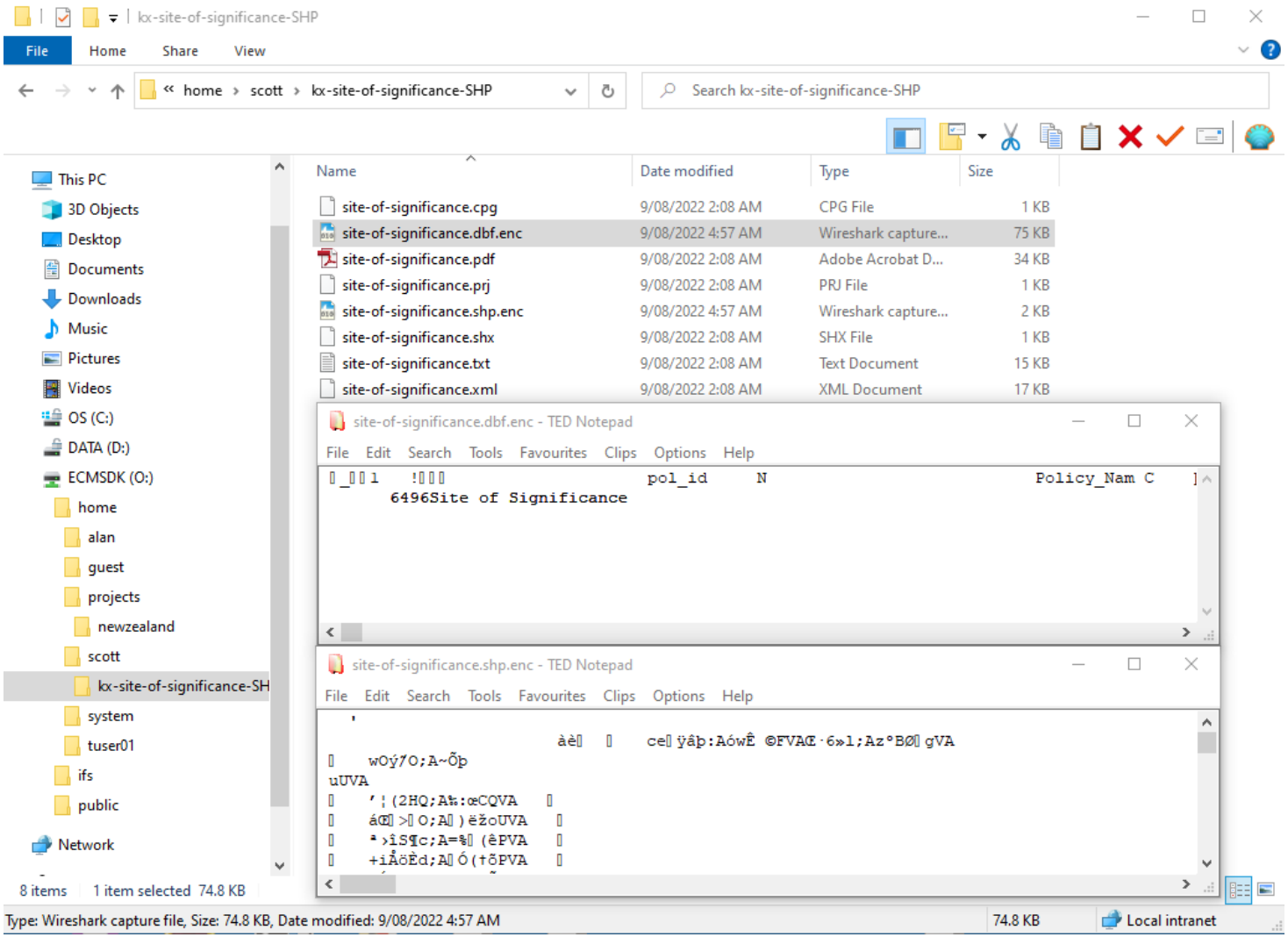


Figure 5: Once user Scott logs in to the ECMSDK, the user session-based trigger is fired, the files are decrypted for use. This includes his eight single-user files (shown decrypted) as well as seven multi-user GIS files to which Scott has privilege (shown decrypted in Fig. 6).

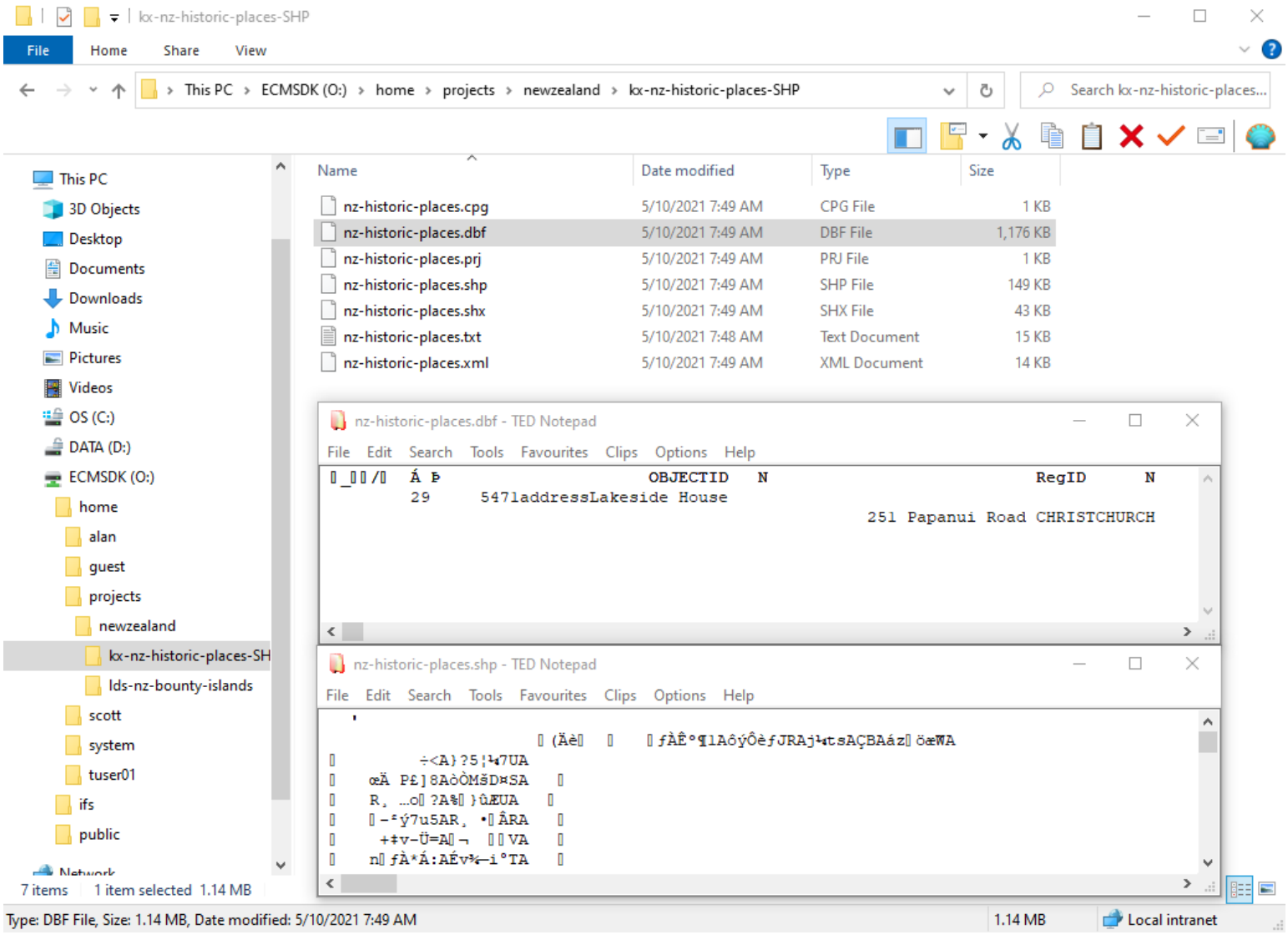


Figure 6: The seven multi-user GIS files to which user Scott has privilege are also decrypted for use. In addition, Scott's actions on other user's folders and files are protected by ECMSDK ACM as Scott cannot access user Alan's home folder which is private.

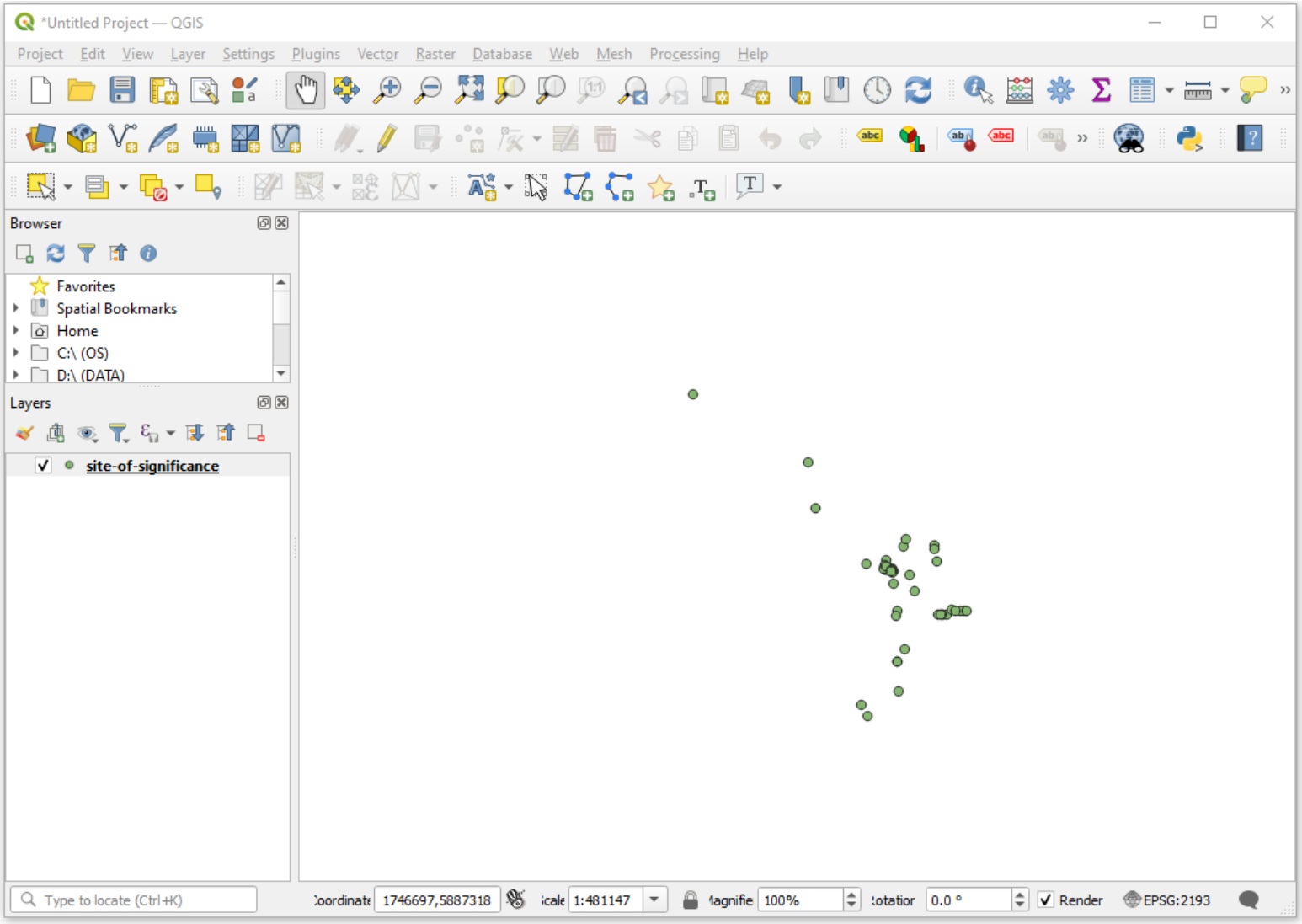


Figure 8: Scott can now remove the ***.enc*** file extension, access, and work with the shapefiles using GIS software such as QGIS.

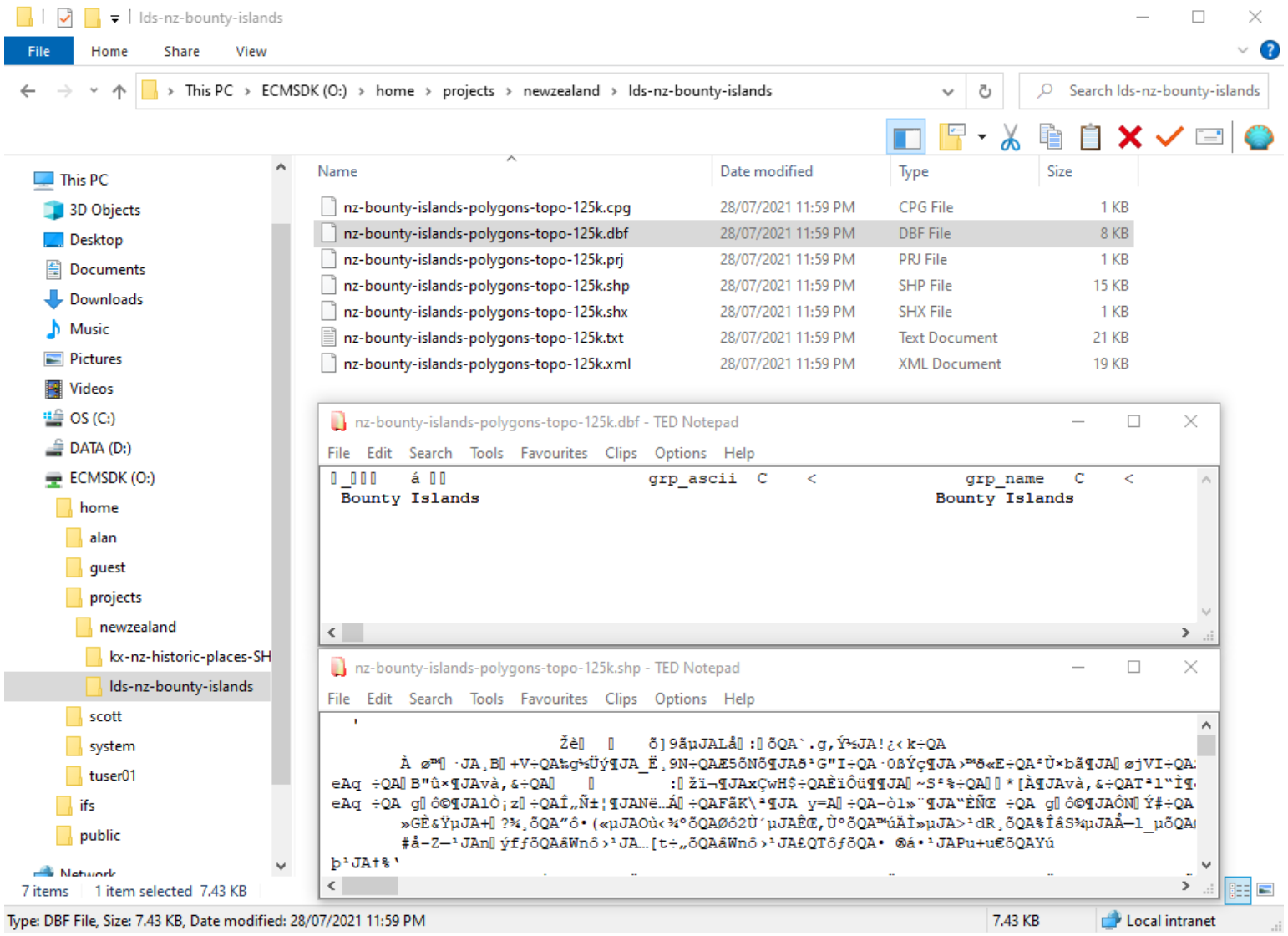


Figure 9: GIS user Alan's login decrypts the remaining seven multi-user GIS files to which only he should have access. The seven multi-user files already decrypted for Scott are not decrypted again. Alan can now access the decrypted contents of the multi-user GIS files.

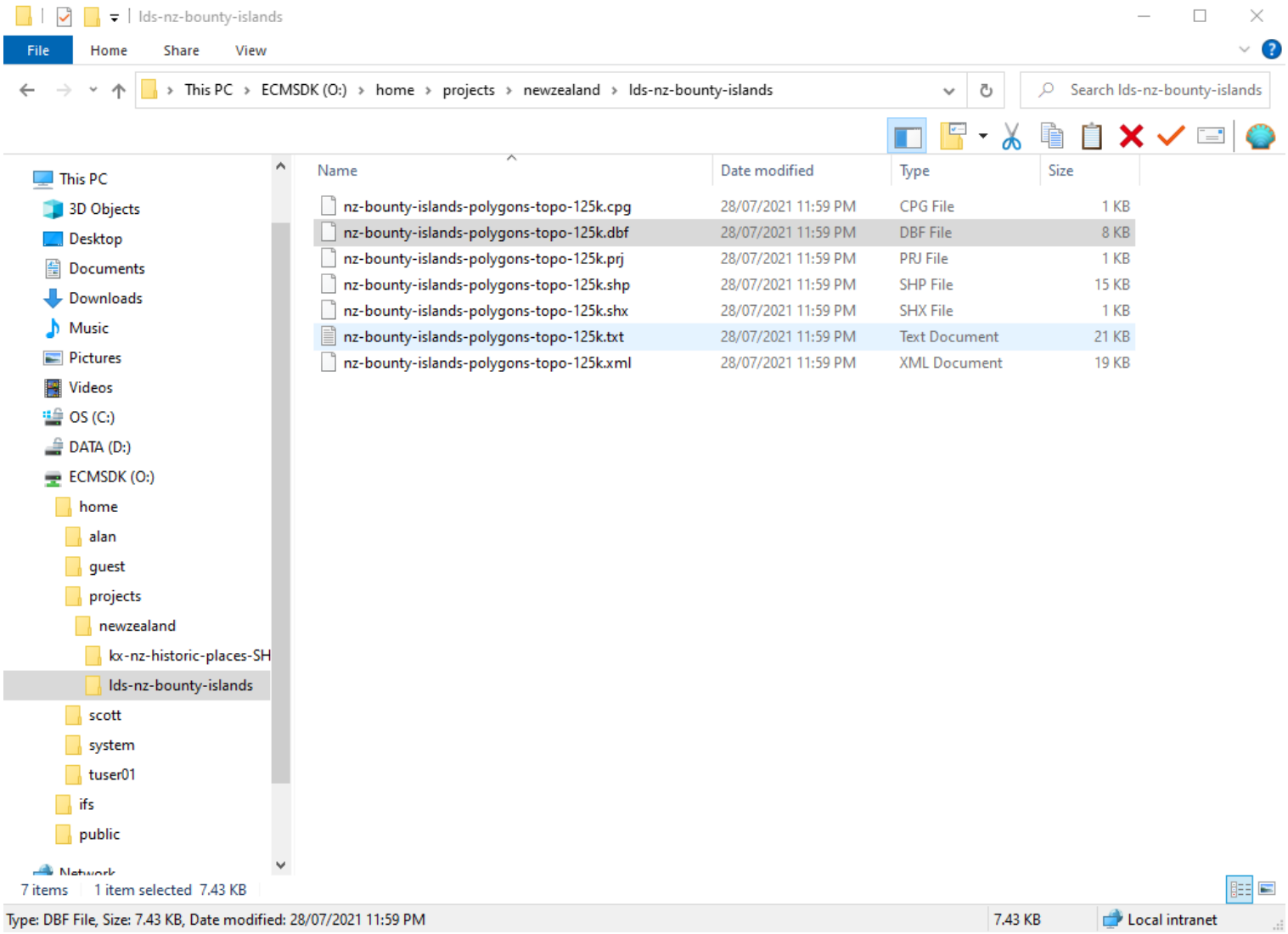


Figure 10: User Alan's logout encrypts only those multi-user GIS files not in use by Scott, as Scott has active DBFS user session. Scott can continue using the seven multi-user files.

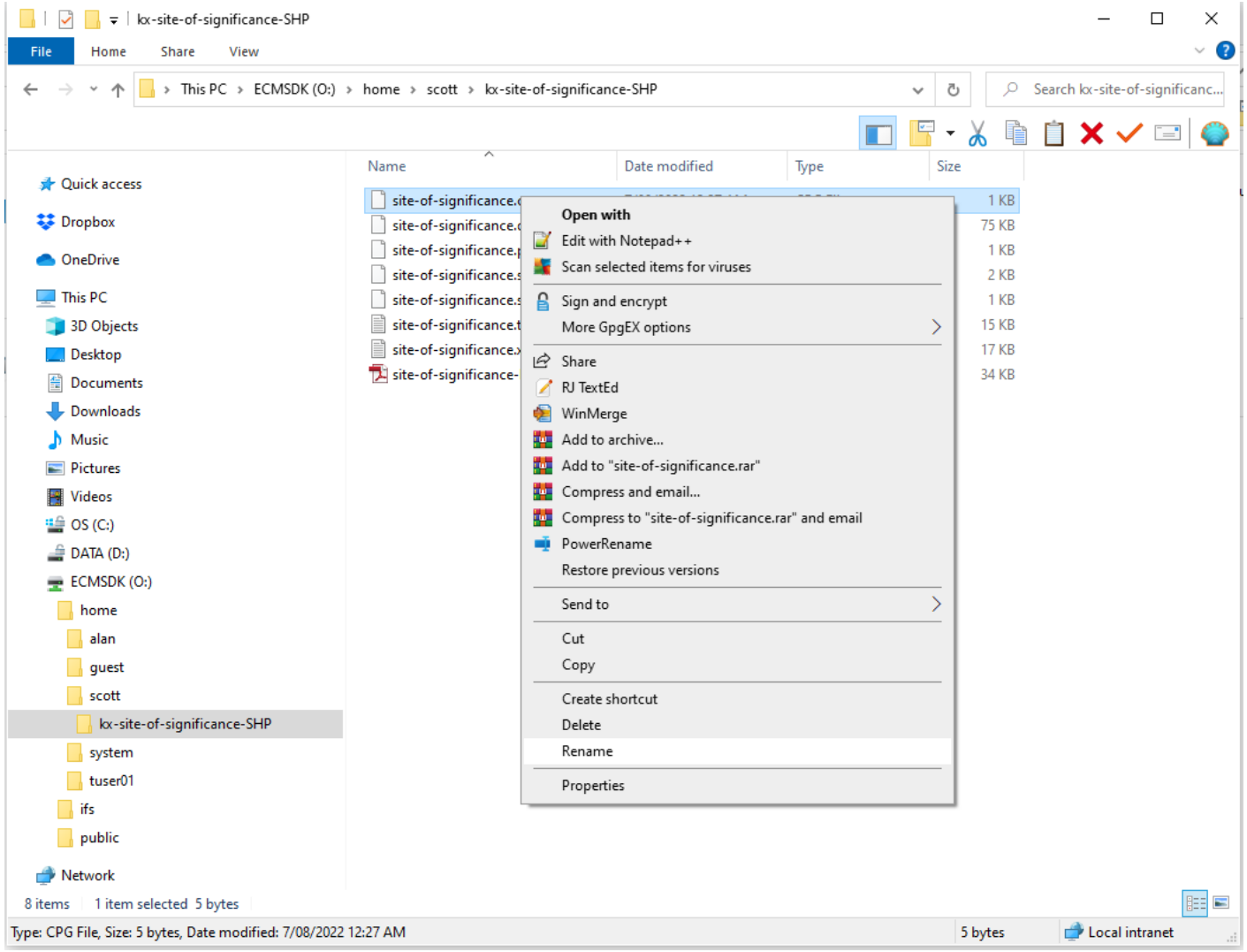


Figure 11: After using the single user files, Scott can specify them for encryption again by adding an *.enc* file extension to them.

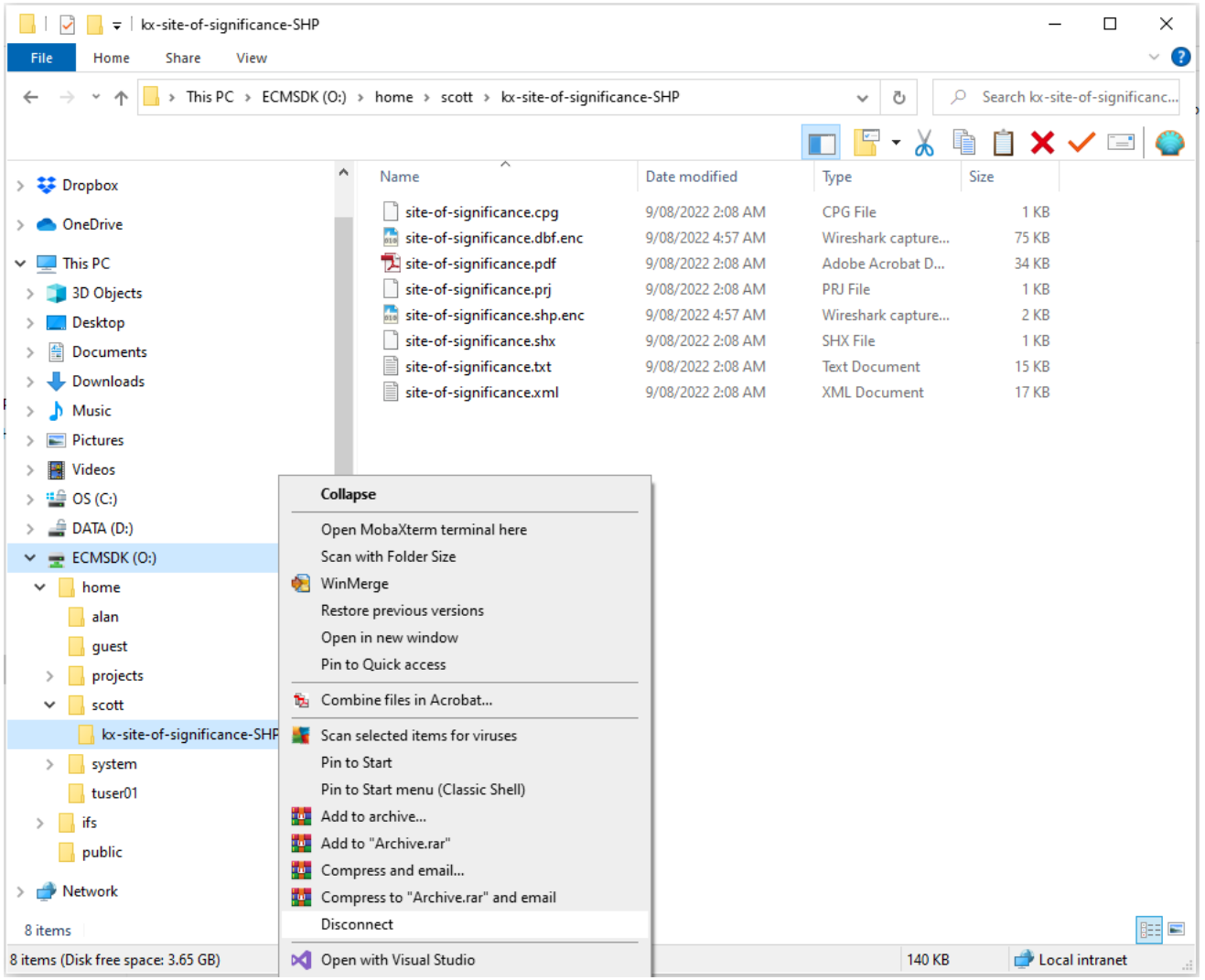


Figure 12: User Scott logs out by disconnecting from the connected ECMSDK shared drive.

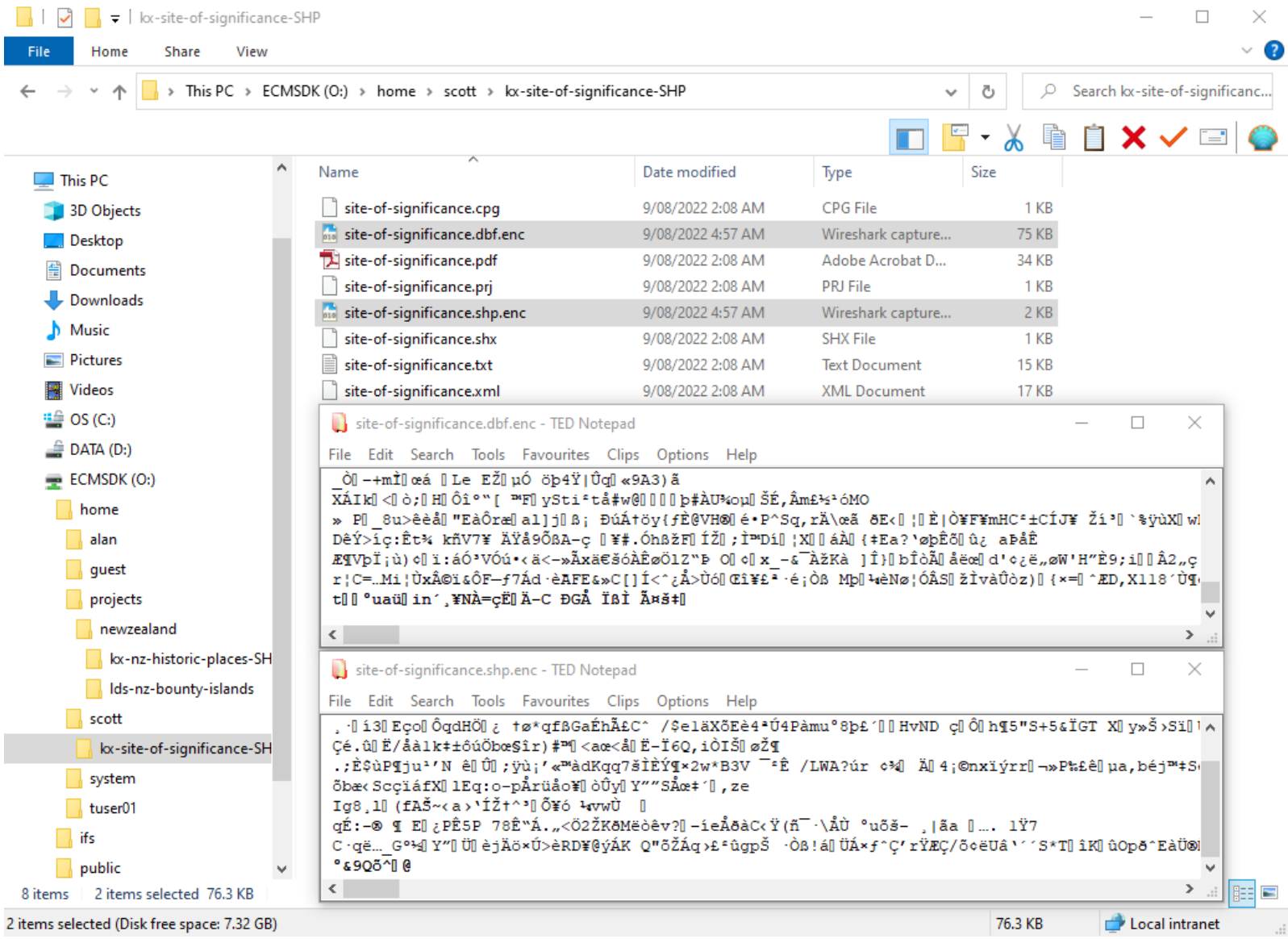


Figure 13: Upon logout, Scott's single-user GIS files are encrypted back for protection.

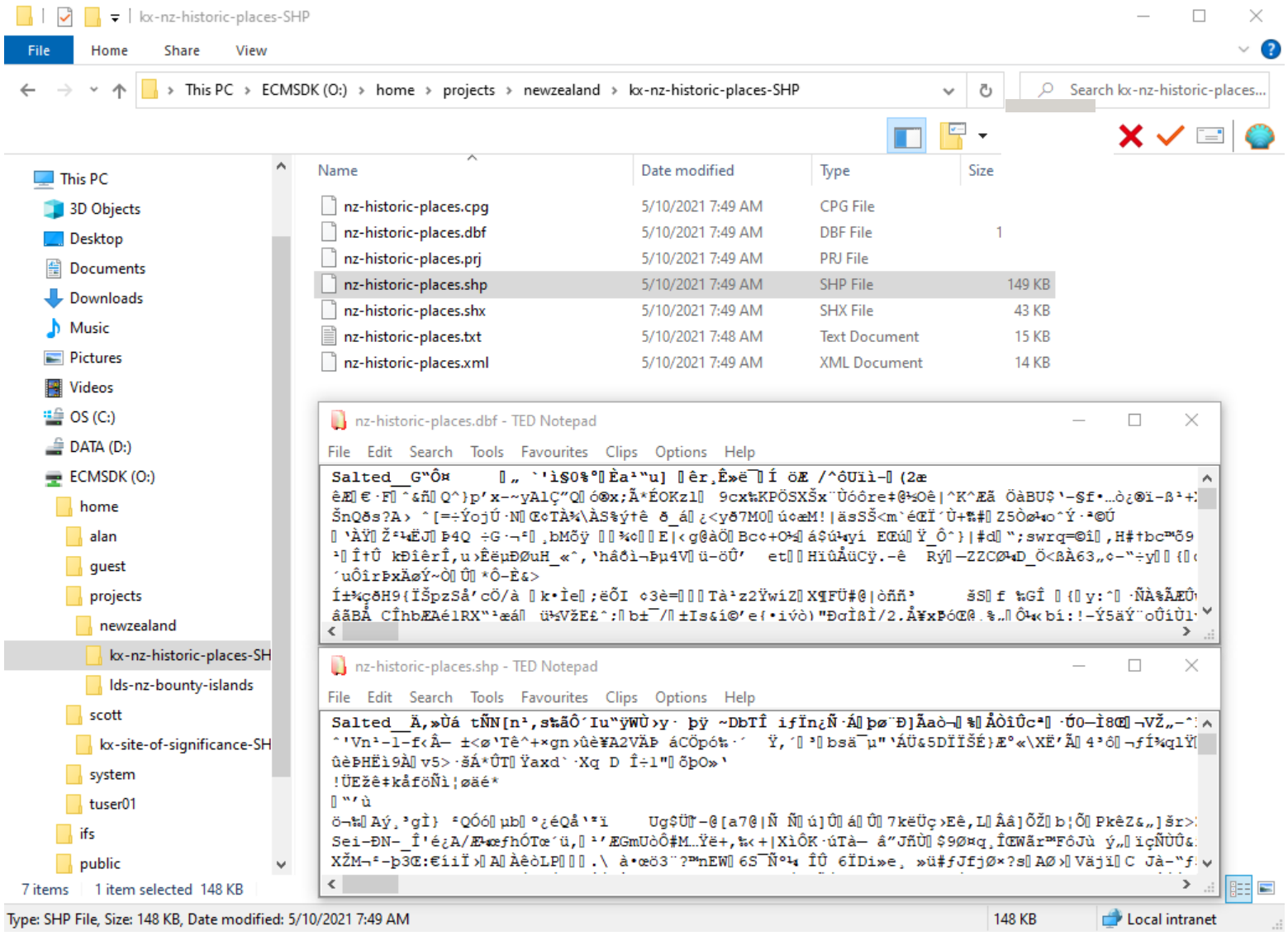


Figure 14: Scott's multi-user GIS files are also encrypted back for protection. The multi-user files are encrypted back only when the last privileged user with active sessions (i.e. Scott) logs out.

Appendix E Security solution implemented within database-driven GIS Web portals & Web maps

The encryption solution can be used for protection of spatial files within a database-centric Web Portal. When GIS users first access the Portal, they see only the public pages and content. To see the protected GIS content, they must be authenticated as one of the authorized Portal user or administrators. Furthermore, the encryption model for files stored in Oracle Portal is similar to our encryption solution for DBFS using ECMSDK, with slight modifications to the procedures, since both products store file content and metadata within the database.

E.1 Single-user and multi-user spatial file encryption schemes

In contrast to ECMSDK, Oracle Portal stores the document metadata and content in the same table; in the **WWDOC_DOCUMENT$**. Thus, information on user-owned sensitive spatial files were retrieved from the same table. The only distinction to the above procedures was the Portal schema, table, and column names in the cursor definition statement. Code segment attached below generates the list of Portal single-user files that have an ***_enc*** file suffix from the Portal schema tables. The ids of these files can then be passed to the encryption and decryption script one by one for encryption and decryption respectively.

```
1.  CURSOR portal_user_files IS
2.     SELECT name, id FROM PORTAL30.WWDOC_DOCUMENT$ WHERE creator = v_userid
3.     AND (name LIKE '%_enc');
```

For sensitive files requiring project level multi-user access, the encryption scheme prepared for ECMSDK users can be developed with slight modification to the content and metadata tables.

E.2 Automated decryption and encryption

Implementation of an automated decryption and encryption model for sensitive files in Portal can be based on Portal user session table **WWCTX_SSO_SESSION\$** within the **PORTAL30** schema table which call the decryption and encryption procedures passing the Portal **user id** from the ID column

However, these sensitive files can be vulnerable when Portal users do not log out properly by shutting down the browser instead of clicking the logout link. In this case, the Portal user session will still be active, and the decrypted files would not be encrypted back for protection. To resolve this, a session cleanup procedure provided by Oracle 9i Portal can be used, namely **ctxjsub**, which can be run periodically as a DBMS job to perform session cleanup for the Portal session table **WWCTX_SSO_SESSION$**. This would delete all inactive sessions, and subsequently the triggers will be fired which would call the encryption procedures to encrypt files back for protection.

An absolute security approach for the GIS Portal could require encryption of Web Mapping files as well. The following section describes how the Web maps personal geodatabase flat files can be encrypted within in Portal as well as.

E.3 Encryption of WebMap geodatabase files

The flat files geodatabases within Web Map applications can be stored and protected within a DBFS. For instance, WebMap geodatabases in Ms Access format can be protected with Oracle ECMSDK folder and file ACL security. Thereafter, these files can be protected against storage media compromise via encryption and decrypted only for use.

The personal geodatabases can be protected using the multi-user access encryption scheme. The end-users would login credentials would be authenticated against those for ECMSDK that will be retrieved from Portal SSO servers and OID. Subsequently, the ECMSDK user sessions triggers would decrypt the files for use and encrypt back for protection when the user session(s) terminate with the logout link.

Figure 17 illustrates the Oracle Portal security model. When users first access the Portal, they only see the public pages and content. To see the protected GIS content, they must be authenticated as an authorized Portal user. The Portal first provides SSO page for users to enter credentials for authentication, which are encrypted in-transit. Upon successful authentication and subsequent Portal user session creation, project-specific and single-user Portal files are decrypted for use.

Thereafter, end-user actions on Portal are controlled through page, content area, folder, and item level ACL security. These are enforced regardless of whether users access through the page interface or directly through the content area. Requests for Web Mapping services are directed to Portal external application provider that authenticates users against their accounts in the Web Map server on their behalf after retrieving them from the OID, providing single sign-on for Web Map external application. After user has finished working with the Portal and logging out successfully, the files would be encrypted back for secure storage.

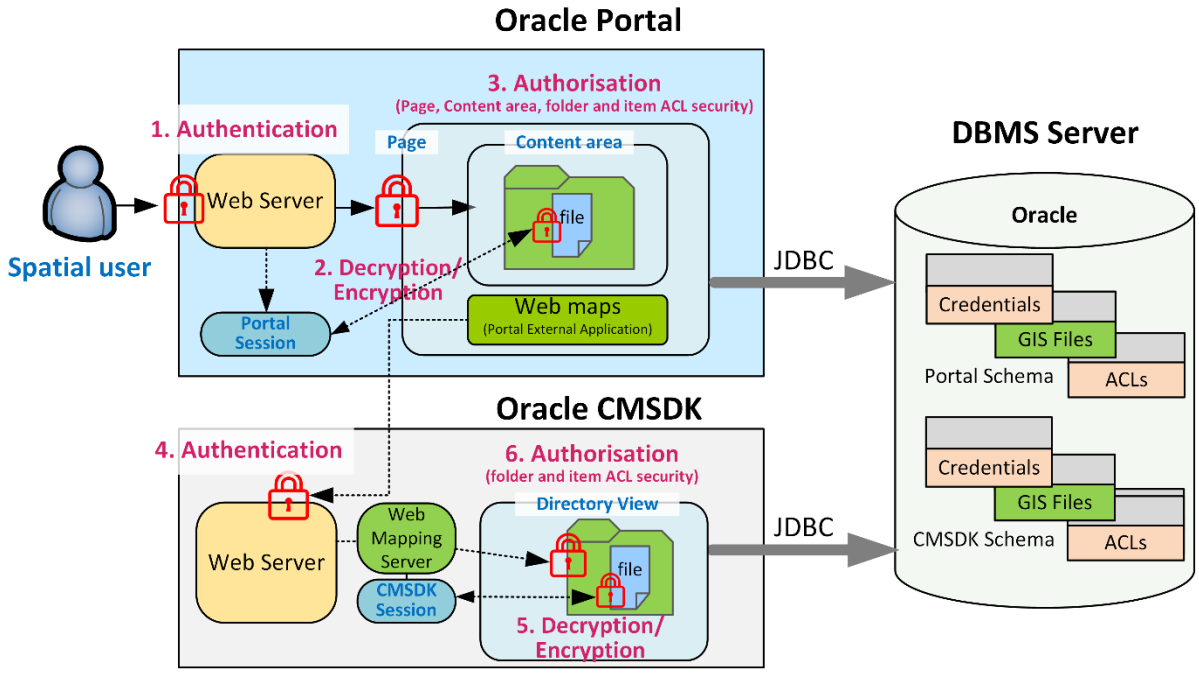


Figure 17: User interaction model for the proposed storage security solution implemented for spatial files within a database-centric web portal and for geodatabases in Web mapping service. Sensitive spatial files are automatically decrypted for use and encrypted back after use.

Security principles to authenticate and authorize users for accessing the protected GIS content through Oracle Portal are as follows:

1. The Portal HTTP Server displays the users with Single Sign-On (SSO) login page for submitting their SSO username and password; this will be the only set of credentials they will have to provide; which are protected in-transit using Portal SSL encryption.
2. The SSO Server verifies the credentials against those stored in OID. If the authentication is successful, the SSO Server creates a session cookie for the user. Information within this session cookie is later used to query the user's privileges specified in OID.

3. Only authorized Portal users can have access to the protected content within the Portal.

4. Portal user-session creation will fire the session-based trigger, calling the decryption procedure upon which the project-specific and single-user files will be decrypted for use.

5. Portal page ACL security controls access to pages; the first part of the Portal authorization process. From within these pages, requests for accessing protected GIS content, are directed to the associated category through the respective portlet.

6. All actions on content are controlled at multiple levels: Portal content area, folder, and item ACL security. Categories are used to display the list of items associated with that category at the same time preventing direct user access to folders and its content.

7. Folder and item ACL security is used to control permissions on submission folders and content.

8. User request for Web Map is passed to Portal external application provider.

9. The external application provider requests the Single Sign-On server (SSO) for user's credentials in the Web Map Server (WMS). SSO retrieves this from Oracle Internet Directory and returns it to the application provider.

10. The external application provider authenticates users against their accounts in the WMS on their behalf, providing single sign-on for Web Mapping services.

11. When Portal user logs out, the user-session termination fires the session-based trigger which calls the encryption procedure for encryption of the project-specific and single-user files back for secure storage.