An Oracle White Paper
July 2011

Oracle Data Pump Encrypted Dump File Support

ORACLE

Introduction 2
Encrypted Dump File Overview 3
Encryption-related Parameters 4
ENCRYPTION 4
ENCRYPTION_ALGORITHM 4
ENCRYPTION_MODE 5
ENCRYPTION_PASSWORD 6
Displaying Dump File Encryption Attributes 6
Using Oracle Data Pump to Create Encrypted Dump Files 7
Using Oracle Data Pump to Load Encrypted Dump Files 9
Encrypted Dump Files and External Tables 11
Conclusion 13

Introduction

The security and compliance requirements in today's business world present manifold challenges. As incidences of data theft increase, protecting data privacy continues to be of paramount importance. Now a de facto solution in meeting regulatory compliances, data encryption is one of a number of security tools in use. The Oracle Advanced Security features built into Oracle Data Pump assist customers in safeguarding sensitive data stored in dump files from unauthorized access. In Oracle Database 11*g* release 1, Oracle Data Pump introduced the encrypted dump file feature. Customers who take advantage of this feature can use Oracle Data Pump to encrypt all data as it is written to the export dump file set. The purpose of this whitepaper is to explain how the Oracle Data Pump encrypted dump file feature works. This paper does not apply to the original Export and Import utilities.

For information regarding the Oracle Data Pump encrypted columns feature released with Oracle Database 10*g* release 2, which provides the ability to re-encrypt all encrypted column data as it is written to the export dump file set, refer to the *Oracle Data Pump Encrypted Columns Support* white paper.

Encrypted Dump File Overview

With Oracle Database 11*g* release 1, Oracle Data Pump introduced a considerable improvement in security by providing dump file encryption using Oracle Advanced Security. Whereas Transparent Data Encryption (TDE) encrypted column support protects only individual columns in the dump file, dump file encryption support protects all table data and system metadata segments written to the dump file. The encrypted dump file feature is not dependent in any way on the TDE encrypted column feature described in the *Oracle Data Pump Encrypted Columns Support* white paper. Although both forms of encryption can be useful to keep information secure while it resides in the database, it is best to think of the two as separate features.

There may be cases in which encryption is not desirable, for performance reasons, or cases in which encryption is not needed because there is already sufficient security within the database. But whenever export dump file sets are used to transfer or archive information, security is important and the ability to encrypt the export dump file is required.

Encryption-related parameters have been added to Oracle Data Pump that provide considerable flexibility in determining how encryption can be applied to a particular export dump file set. Options exist which allow you to encrypt just the system metadata segments, just the table data segments, or the entire contents of the dump file. Additionally, you can choose to encrypt the dump file using only a user-provided password or you can make use of the Oracle encryption wallet to transparently encrypt the dump file without the need for a user-provided password. You can also choose from among three different encryption algorithms.

Encrypted dump files are decrypted automatically during import operations, provided that authorized personnel either specify the required user-supplied password or ensure that the wallet is open. Either the master key from the source database can be used, or, if the data is to be recovered into another database, the encryption key can be generated from a password, which removes the requirement to share the master key between databases.

Dump file encryption can also be used in conjunction with the dump file compression feature. During an export operation, data is first compressed and then encrypted before being written to the dump file set. After reading data blocks from the dump file set, Import decrypts and decompresses the data before loading it into the target database.

Oracle Data Pump performs encryption and decryption as inline operations; there is no need for separate encryption utilities, additional intermediate disk space, or operating system pipes. Moreover, system metadata segments and table data segments are encrypted independently of one another. This means that loading a subset of the data in a dump file set, as in a table-mode or a schema-mode import, does not require reading and decrypting the entire dump file set. Oracle Data Pump can directly access the target objects in the dump file via an index in the job's master table and then decrypt only the necessary data.

Exporting and importing encrypted dump files may have a slightly negative impact on the overall performance of Data Pump jobs because even though the data being processed is stored in memory

buffers, encryption and decryption are typically CPU intensive operations. However, there is very little space overhead added to the encrypted data because no data integrity checks are performed on encrypted dump files and because salt is added to entire metadata and table data segments rather than to individual columns within each data row, as is done for TDE encrypted columns.

Encryption-related Parameters

Several encryption parameters have been added to Oracle Data Pump that allow you to specify which components of an export dump file to encrypt, which algorithm to use when encrypting data, and which security mode to employ when you elect to encrypt your export dump file.

ENCRYPTION

The ENCRYPTION parameter is used to specify whether or not to encrypt the export dump file. It can also be used to limit which components within the dump file get encrypted. Acceptable values for this parameter are as follows:

- ALL – encrypts all metadata and table data segments
- DATA_ONLY – encrypts only table data segments
- ENCRYPTED_COLUMNS_ONLY – re-encrypts only TDE encrypted columns
- METADATA_ONLY – encrypts only system metadata segments
- NONE – no encryption is performed

The ENCRYPTION parameter defaults to the value ALL if the ENCRYPTION_PASSWORD parameter is specified. But if neither ENCRYPTION nor ENCRYPTION_PASSWORD is specified, then ENCRYPTION defaults to NONE.

System metadata segments refer only to dictionary objects such as GRANTS, USERS, and INDEXES that are exported as part of a schema-mode or full-mode export. These objects are exported in XML format and then encrypted before being written to the dump file when ENCRYPTION is set to ALL or METADATA_ONLY.

A table data segment refers to a data structure stored in the dump file that contains a table's column definitions, row data, and other control information. The column definitions can also be thought of as table metadata and are exported in XML format, similar to system metadata segments. The entire table data segment, including column definitions, is encrypted before being written to the dump file when ENCRYPTION is set to ALL or DATA_ONLY.

ENCRYPTION_ALGORITHM

Oracle Data Pump employs the Advanced Encryption Standard (AES) cryptographic algorithm when performing encryption. The AES standard is a symmetric key algorithm that uses the same encryption

key for both encryption and decryption. It encrypts and decrypts data in blocks of 128 bits and can use encryption key sizes of 128, 192, and 256 bits.

The ENCRYPTION_ALGORITHM parameter accepts values of AES128, AES192, and AES256, allowing you to specify which algorithm you want Oracle Data Pump to use. The AES128 algorithm is the default.

ENCRYPTION_MODE

The ENCRYPTION_MODE parameter determines the method by which encryption keys are generated. Allowable values for this parameter are as follows:

- PASSWORD -This mode requires that the user provide a password when creating and reading encrypted dump file sets.
- Encryption keys used during the export operation are derived from the password.
- This is the default mode if the ENCRYPTION_PASSWORD parameter is specified and the wallet is closed.
- Importing a password-encrypted dump file set requires the same password that was used to create the dump file set.
- Password encryption is best suited for cases where the dump file set will be imported into a different or remote database but which must remain secure in transit.
- TRANSPARENT - This mode allows the creation and reading of encrypted dump file sets with no user password required, as long as the required wallet is available.
  - Encryption keys used during the export operation are based upon the database master key in the wallet.
  - This is the default mode if only the ENCRYPTION parameter is provided and the wallet is open.
  - Transparent encryption is best suited for cases where the dump file set will be imported into the same database from which it was exported.
- DUAL - This mode allows encrypted dump file sets to be imported either transparently or by specifying a password that was used when the dual-mode encrypted dump file set was created.
  - Encryption keys used during the export operation are based upon the database master key in the wallet.
  - This is the default mode if the ENCRYPTION_PASSWORD parameter is specified and the wallet is open.
  - When importing a dual-mode encrypted dump file set, either the wallet or the password can be used.

- Dual-mode encryption is best suited for cases where the dump file set will be imported on-site using the wallet, but which might also need to be imported off-site where the wallet is not available.

As these descriptions for the various encryption modes have shown, the default value for ENCRYPTION_MODE is dependent on which other encryption-related parameters are specified and on whether the wallet is open or closed. Note also that if only the ENCRYPTION_MODE parameter is specified and the wallet is closed, then an error is returned.

ENCRYPTION_PASSWORD

The ENCRYPTION_PASSWORD parameter is not new in Oracle Data Pump 11*g* release 1. It was first introduced in Oracle Data Pump 10*g* release 2 and was used when exporting TDE encrypted columns. It now can also be used when creating encrypted dump file sets.

The password value that is supplied specifies a key for re-encrypting encrypted table columns, metadata segments, or table data segments so that they are not written as clear text in the dump file set. If the export operation involves encrypted table columns, but an encryption password is not supplied, then the encrypted columns are written to the dump file set as clear text and a warning is issued.

For export operations, this parameter is required if the ENCRYPTION_MODE parameter is set to either PASSWORD or DUAL. It is also required if the ENCRYPTION parameter is set to ENCRYPTED_COLUMNS_ONLY.

Displaying Dump File Encryption Attributes

The Oracle Data Pump API procedure DBMS_DATAPUMP.GET_DUMPFILE_INFO can be used to determine whether or not a dump file is encrypted and if so, which encryption mode was used at the time it was created. Assume that the following anonymous PL/SQL block is contained within a script called **encrypt_attrs.sql**. This script is used in the examples that follow in order to verify which segments of the dump file were encrypted and which encryption mode was employed.

```
SET SERVEROUTPUT ON
SET VERIFY OFF
SET FEEDBACK OFF
DEFINE dumpfile = &1
DECLARE
  dfnum      NUMBER;
  ind        NUMBER;
  file_type  NUMBER;
  info_table KU$_DUMPFILE_INFO := KU$_DUMPFILE_INFO();
BEGIN
  DBMS_DATAPUMP.GET_DUMPFILE_INFO('&dumpfile','DPUMP_DIR',
                                  info_table,file_type);
  IF file_type = 1   -- 1 means it's Data Pump dump file
  THEN
    IF info_table IS NOT NULL
    THEN
      ind := info_table.FIRST;
      WHILE ind IS NOT NULL
      LOOP
        CASE info_table(ind).item_code
          WHEN DBMS_DATAPUMP.KU$_DFHDR_METADATA_ENCRYPTED THEN
            DBMS_OUTPUT.PUT_LINE('Metadata encrypted = ' ||
                                  info_table(ind).value);
          WHEN DBMS_DATAPUMP.KU$_DFHDR_DATA_ENCRYPTED THEN
            DBMS_OUTPUT.PUT_LINE('Table data encrypted = ' ||
                                  info_table(ind).value);
          WHEN DBMS_DATAPUMP.KU$_DFHDR_ENCPWD_MODE THEN
            CASE info_table(ind).value
              WHEN DBMS_DATAPUMP.KU$_DFHDR_ENCPWD_MODE_NONE THEN
                DBMS_OUTPUT.PUT_LINE('Encrypt Mode = None');
              WHEN DBMS_DATAPUMP.KU$_DFHDR_ENCPWD_MODE_PASSWORD THEN
                DBMS_OUTPUT.PUT_LINE('Encrypt Mode = Password');
              WHEN DBMS_DATAPUMP.KU$_DFHDR_ENCPWD_MODE_DUAL THEN
                DBMS_OUTPUT.PUT_LINE('Encrypt Mode = Dual');
              WHEN DBMS_DATAPUMP.KU$_DFHDR_ENCPWD_MODE_TRANS THEN
                DBMS_OUTPUT.PUT_LINE('Encrypt Mode = Transparent');
              WHEN DBMS_DATAPUMP.KU$_DFHDR_ENCPWD_MODE_UNKNOWN THEN
                DBMS_OUTPUT.PUT_LINE('Encrypt Mode = Unknown');
            END CASE;
          ELSE NULL;  -- Ignore other dump file attributes
        END CASE;
        ind := info_table.NEXT(ind);
      END LOOP;
    END IF;
  END IF;
END;
/
```

Using Oracle Data Pump to Create Encrypted Dump Files

The following example performs a schema-mode export using only the ENCRYPTION_PASSWORD parameter. Because only ENCRYPTION_PASSWORD is specified, ENCRYPTION defaults to ALL,

meaning that both metadata and table data is encrypted. Also, ENCRYPTION_MODE defaults to PASSWORD because the wallet is not open. The results of running the **encrypt_attrs.sql** script verify this.

```
$ expdp dp/dp DIRECTORY=dpump_dir DUMPFILE=dp1.dmp \
ENCRYPTION_PASSWORD=dump_pwd

$ sqlplus dp/dp
SQL> @encrypt_attrs dp1.dmp
Metadata segments encrypted = 1      (1 = Yes/True, 0 = No/False)
Table data segments encrypted = 1
Encrypt Mode = Password
```

In the next example, a schema-mode export is attempted in which the ENCRYPTION parameter is provided but neither the ENCRYPTION_PASSWORD parameter nor the ENCRYPTION_MODE parameter is specified. In this case, Oracle Data Pump assumes that the encryption mode is TRANSPARENT and tries to access the wallet. Because the wallet is not open, an error is raised and the export job is aborted. After the wallet is manually opened, the export job completes successfully. The encryption attributes for the dp2.dmp dump file show that the encryption mode did in fact default to TRANSPARENT.

```
$ expdp dp/dp DIRECTORY=dpump_dir DUMPFILE=dp2.dmp \
ENCRYPTION=DATA_ONLY

Export: Release 11.2.0.2.0 - Production on Tue Aug 17 09:57:00 2010

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights
reserved.

Connected to: Oracle Database 11g Enterprise Edition Release
11.2.0.2.0 - Production
With the Partitioning, Data Mining and Real Application Testing
options
ORA-39002: invalid operation
ORA-39188: unable to encrypt dump file set
ORA-28365: wallet is not open

SQL> ALTER SYSTEM SET ENCRYPTION WALLET OPEN IDENTIFIED BY
"wallet_pwd";

$ expdp dp/dp DIRECTORY=dpump_dir DUMPFILE=dp2.dmp \
ENCRYPTION=DATA_ONLY

SQL> @encrypt_attrs dp2.dmp
Metadata segments encrypted = 0
Table data segments encrypted = 1
Encrypt Mode = Transparent
```

In the next and final export example, another schema-mode export is executed. This time, only metadata segments are to be encrypted, using the AES256 algorithm. Because the wallet is now open,

the encryption mode selected by Oracle Data Pump is DUAL even though the ENCRYPTION_PASSWORD parameter was specified.

```
$ expdp dp/dp DIRECTORY=dpump_dir DUMPFILE=dp3.dmp \
ENCRYPTION=METADATA_ONLY ENCRYPTION_PASSWORD=dump_pwd \
ENCRYPTION_ALGORITHM=AES256

SQL> @encrypt_attrs dp3.dmp
Metadata segments encrypted = 1
Table data segments encrypted = 0
Encrypt Mode = Dual
```

Using Oracle Data Pump to Load Encrypted Dump Files

Importing an encrypted dump file set is quite straightforward. The only possible encryption-related parameter is ENCRYPTION_PASSWORD and Oracle Data Pump automatically detects when it must be provided. In the following example, an attempt is made to load a password-mode encrypted dump file, dp1.dmp. Because ENCRYPTION_PASSWORD was not specified, an error is raised and the import job is aborted. The import job runs successfully after retrying it with the correct password.

```
$ impdp system/admin DIRECTORY=dpump_dir DUMPFILE=dp1.dmp

Import: Release 11.2.0.2.0 - Production on Tue Aug 17 09:58:04
2010

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release
11.2.0.2.0 - Production
With the Partitioning, Data Mining and Real Application Testing
options
ORA-39002: invalid operation
ORA-39174: Encryption password must be supplied.

$ impdp system/admin DIRECTORY=dpump_dir DUMPFILE=dp1.dmp
ENCRYPTION_PASSWORD=dump_pwd
```

In the next example, the wallet is manually closed and then an attempt is made to import a transparent-mode encrypted dump file, dp2.dmp. It is assumed that this is the same system on which the dp2.dmp dump file was created. Oracle Data Pump detects that the wallet is required to decrypt this dump file but finds the wallet closed, so an error is raised and the import job is aborted. The import job runs successfully after the wallet is manually opened.

```
SQL> ALTER SYSTEM SET WALLET CLOSE;

$ impdp system/admin DIRECTORY=dpump_dir DUMPFILE=dp2.dmp

Import: Release 11.2.0.2.0 - Production on Tue Aug 17 10:03:01
2010

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release
11.2.0.2.0 - Production
With the Partitioning, Data Mining and Real Application Testing
options
ORA-39002: invalid operation
ORA-39189: unable to decrypt dump file set
ORA-28365: wallet is not open

SQL> ALTER SYSTEM SET ENCRYPTION WALLET OPEN IDENTIFIED BY
"welcome1";

$ impdp system/manager DIRECTORY=dpump_dir DUMPFILE=dp2.dmp
```

The following example is similar to the previous one. An attempt is made to import a dual-mode encrypted dump file after first manually closing the wallet. Oracle Data Pump detects that either an encryption password or the wallet is required to decrypt this dump file. Since no encryption password was provided and the wallet is closed, an error is raised and the import job is aborted. The import job runs successfully after the wallet is manually opened. Alternatively, the wallet could have been kept closed and the correct encryption password could have been used.

```
SQL> ALTER SYSTEM SET WALLET CLOSE;

$ impdp system/admin DIRECTORY=dpump_dir dumpfile=dp3.dmp

Import: Release 11.2.0.2.0 - Production on Tue Aug 17 10:04:23
2010

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release
11.2.0.2.0 - Production
With the Partitioning, Data Mining and Real Application Testing
options
ORA-39002: invalid operation
ORA-39189: unable to decrypt dump file set
ORA-28365: wallet is not open

SQL> ALTER SYSTEM SET ENCRYPTION WALLET OPEN IDENTIFIED BY
"welcome1";

$ impdp system/admin DIRECTORY=dpump_dir DUMPFILE=dp3.dmp
```

This last example is again similar to the previous one. However, in this case an attempt is made to import a dual-mode encrypted dump file, dp3.dmp, on a different database from that used to create it. This example assumes that the wallet on the target database does not contain the same master key that was used on the source database when the export was performed. Oracle Data Pump detects that either an encryption password or the wallet is required to decrypt this dump file. Since no encryption password was provided and the wallet does not contain the required master key, an error is raised and the import job is aborted. The import job runs successfully after the correct password is provided.

```
$ impdp system/admin DIRECTORY=dpump_dir dumpfile=dp3.dmp

Import: Release 11.2.0.2.0 - Production on Tue Aug 17 10:06:08
2010

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release
11.2.0.2.0 - Production
With the Partitioning, Data Mining and Real Application Testing
options
ORA-39002: invalid operation
ORA-39189: unable to decrypt dump file set
ORA-28362: master key not found

$ impdp system/admin DIRECTORY=dpump_dir DUMPFILE=dp3.dmp \
ENCRYPTION_PASSWORD=dump_pwd
```

Encrypted Dump Files and External Tables

The external tables feature allows you to access data in an external operating system file as if it were inside a table residing in the database. An external table definition is created using the SQL syntax CREATE TABLE … ORGANIZATION EXTERNAL and specifying the ORACLE_DATAPUMP access driver in the TYPE clause.

Note that an external table export dump file is not the same export dump file as produced by the Oracle Data Pump Export utility (expdp). The dump file can be encrypted but the wallet must first be open before creating the external table. It is not possible to use an encryption password when creating encrypted dump files for external tables.

The following example creates an external table called SCOTT.XDEPT and populates it using the data in the SCOTT.DEPT table. The datatypes for the columns are not specified because they are determined by the column datatypes in the source table named in the SELECT subquery. The ENCRYPTION ENABLED clause instructs the ORACLE_DATAPUMP access driver to encrypt the contents of the dump file.

```
SQL> CREATE TABLE SCOTT.XDEPT (
      deptno,
      dname,
      loc)
      ORGANIZATION EXTERNAL
     (
      TYPE ORACLE_DATAPUMP
      DEFAULT DIRECTORY dpump_dir
      ACCESS_PARAMETERS (ENCRYPTION ENABLED)
      LOCATION ('xdept.dmp')
     )
      REJECT LIMIT UNLIMITED
     AS SELECT * FROM SCOTT.DEPT;
```

When the following SELECT statement is executed, the ORACLE_DATAPUMP access driver recognizes that the dump file is encrypted. After it checks to make sure that the wallet is open, it proceeds to decrypt the data.

```
SQL> SELECT * FROM DP.XDEPT;
```

Conclusion

The encrypted dump file support within Oracle Data Pump allows you to maximize your investment in Oracle Advanced Security technology by providing a simple mechanism to ensure that sensitive data remains protected when it is exported out of the database. The examples in this paper illustrate how simple it is to export and import encrypted dump files. DBAs must always weigh security concerns against any performance impact that will be incurred as part of encrypting and decrypting data.

References

1. *Oracle Database Utilities* (Part Number E10701)
2. *Oracle Database SQL Language Reference* (Part Number E10592)
3. The "Using Oracle Wallet Manager" and "Securing Stored Data Using Transparent Data Encryption" chapters in *Oracle Database Advanced Security Administrator's Guide* (Part Number E10746)
4. Oracle Data Pump Encrypted Columns Support (White Paper)

Oracle is committed to developing practices and products that help protect the environment

Oracle Data Pump Encrypted Dump File Support
Author: John Kalogeropoulos
Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Hardware and Software, Engineered to Work Together