ORACLE

PEOPLESOFT ENTERPRISE

Oracle Advanced Security TDE Tablespace Encryption for PeopleTools 8.48 and higher

Table of Contents

Copyright © 2010 Oracle USA, Inc. All rights reserved.

Chapter 1 - Introduction

This Red Paper is a practical guide for technical users, installers, system administrators, Oracle Database Security Administrators, and programmers who implement, maintain, or develop applications for your PeopleSoft system. In this Red Paper, we discuss guidelines on how to implement Oracle Transparent Data Encryption in a PeopleSoft environment, including installation, configuration, and related security considerations.

Much of the information contained in this document originated within PeopleSoft Development in collaboration with some of our big customers and is therefore based on "real-life" customer use cases encountered in the field. Although every conceivable problem that one could encounter in this document, the issues that appear are the problems that prove to be the most common.

STRUCTURE OF THIS RED PAPER

This Red Paper provides guidance for deploying Oracle Transparent Data Encryption tablespace encryption with PeopleTools 8.48 and higher on Oracle Database release 11gR1 and higher.

Keep in mind that PeopleSoft updates this document as needed so that it reflects the most current feedback we receive from the field. Therefore, the structure, headings, content and length of this document are likely to vary with each posted version. To see if the document has been updated since you last downloaded it, compare the date of your version to the date of the version posted on Customer Connection.

RELATED MATERIALS

We assume that our readers are experienced IT professionals, with a good understanding of PeopleSoft's Internet Architecture and Oracle Database administration. To take full advantage of the information covered in this document, we recommend that you have a basic understanding of system administration, basic Internet architecture, relational database concepts/SQL, and how to use PeopleSoft applications.

This document is not intended to replace the documentation delivered with the PeopleTools PeopleBooks. We recommend that before you read this document, you read the related information in the PeopleTools PeopleBooks as well as the

- Transparent Data Encryption Best Practices paper
- Transparent Data Encryption Frequently Asked Questions
- Oracle Advanced Security Administrator's Guide
- Oracle Database Security Guide
- Database 2 Day + Security Guide

Note: Much of the information in this document eventually gets incorporated into subsequent versions of the PeopleBooks. Should any issues arise for customers choosing to implement TDE tablespace encryption in a PeopleSoft Enterprise environment, Oracle support will help isolate any issues specific to TDE tablespace encryption and guide customers to the proper support channel.

Chapter 2 - Overview of how Oracle TDE Tablespace Encryption works

Oracle Transparent Data Encryption provides seamless encryption of sensitive PeopleSoft application data without changes to the application itself. Regulations such as the Payment Card Industry Data Security Standard (PCI-

Copyright © 2010 Oracle USA, Inc. All rights reserved.

DSS), the HiTECH act, as well as most Breach Notification laws require encryption of stored data as the strongest protection against identity theft once the data left the secure perimeter of the database, for example when a hard disk is exchanged for maintenance, or data copied to a backup tape never arrives at the storage facility. Complementing the strong access controls and separation of duty provided by Oracle Database Vault, Oracle Transparent Data Encryption brings you another step closer to compliance.

As opposed to invasive 3rd party encryption solutions, Oracle Transparent Data Encryption transparently encrypts data when it is written to the database files on disk, and decrypts it when it is read back to an authorized user. 3rd party encryption solutions require substantial changes to your applications; Oracle Transparent Data Encryption provides turnkey encryption with built-in key management, without any triggers, views, or stored procedures usually implemented in costly long-term consulting projects. With encrypted SQL*net traffic to and from the database, TDE tablespace encryption for encryption of stored data in your database, to encrypted export and backup files, Oracle offers encryption of data during it's life time.

TDE Tablespace Encryption

Tablespaces are the storage containers of all Oracle objects. With Oracle Database 11g, they can be encrypted with no additional storage space requirements. Data that is read from storage is automatically decrypted before it arrives in database memory (SGA), so that the database does not even 'see' that it processes data that is stored in encrypted form; this is especially important for the fact that the execution plans don't change, and the original caching mechanisms inside the database continue to work as before. TDE tablespace encryption uses a two-tier key architecture: The tablespace key that encrypts and decrypts the application data in that tablespace is stored in the tablespace header. These keys are encrypted with the master key, which is stored outside of the database, in the encrypted Oracle Wallet, or a PKSC#11 compatible Hardware Security Module, HSM.

Oracle Transparent Data Encryption benefits Oracle PeopleSoft customers by encrypting sensitive, personally identifiable information (PII), or other information crucial to their business on storage media, as required by many security, privacy, and breach disclosure laws. By protecting the encryption master key with a password that may be unknown to the DBA, Oracle Transparent Data Encryption provides separation of duty.

The limitations known from TDE column encryption in terms of supported data types, indexes, foreign key relationships do *not* apply anymore; TDE tablespace encryption provides 100% transparency, requires no additional storage; the performance impact of TDE tablespace encryption has been measured to be between 2% and 5% for normal operations.

Chapter 3: Implementing TDE tablespace encryption:

Encrypt before installing Oracle PeopleSoft Enterprise

Locate the '`xxddl.sql`' script in your PeopleSoft staging directory, which contains the SQL commands to initially create the tablespaces that will later hold the entire application. Substitute '`xx`' with the 2 characters used for your specific application, for example '`epddl.sql`'. Commands for clear text tablespaces:

```
create tablespace AMAPP datafile '/opt/oracle/oradata/psft/amapp.dbf' SIZE 90M extent
management local autoallocate;
```

can be changed to:

```
create tablespace AMAPP datafile '/opt/oracle/oradata/psft/amapp.dbf' SIZE 90M extent
management local autoallocate encryption using 'AES256' default storage(encrypt);
```

After enabling TDE in your database (by creating a wallet and adding a master encryption key to it), the script is executed in place of the original script. Your tablespaces are now created in encrypted form, and all data that is later added to it, will be encrypted as well.

Copyright © 2010 Oracle USA, Inc. All rights reserved.

Optionally, only tablespaces storing sensitive data can be defined as encrypted, while others remain un-encrypted. If application tablespaces are encrypted, it is highly recommended to always encrypt the PSINDEX tablespace, since it will store the indexes used for tables in clear text and encrypted tablespaces.

Encrypt an existing Oracle PeopleSoft Enterprise application

Existing clear text tablespaces cannot be encrypted; their content needs to be migrated into new encrypted tablespaces. An easy, non-interruptive method of migrating to encrypted tablespaces is to leverage Online Table Redefinition, a mature High-Availability feature of the Oracle Database. Oracle provides a free Online Migration Guide (incl. complete SQL script) that automates the migration of your Oracle PeopleSoft instance from clear text to encrypted tablespaces, one tablespace at a time:
http://www.oracle.com/technology/deploy/security/database-security/pdf/tde_tabsp_enc_for_psft.zip

Re-key encrypted tablespaces

Starting with Oracle Database 11g Release 2, the unified master encryption keys for TDE column encryption and TDE tablespace encryption can be re-keyed, and migrated to a Hardware Security Module (HSM).

For TDE tablespace encryption in Oracle Database 11gR1 (11.1.0.7), the master key for TDE tablespace encryption cannot be re-keyed (regardless if stored in the Oracle Wallet or HSM). In 11.1.0.6, the master key for TDE tablespace encryption cannot be stored in an HSM device. If there is reason to believe the master key has been compromised, or a privacy and security law that applies to your specific industry requires a master key re-key operation, apply the script you used to encrypt your existing application to relocate the data from one encrypted tablespace to a new encrypted tablespace; this new tablespace will be encrypted with a new tablespace key, which equals to a re-key operation.

Chapter 4 - Considerations for Protecing the Oracle Wallet

Directory and file permissions

The encrypted Oracle Wallet is the default secure container of the TDE master encryption key.

It is recommended to define the location for the Oracle Wallet outside of the $ORACLE_BASE directory tree, in order to avoid that the wallet is stored with encrypted data on a backup tape; one suggestion is the default directory where Oracle Wallet Manager expects a wallet:

```
/etc/ORACLE/WALLETS/oracle
```

Since `/etc` is owned by '`root`', these directories need to be created by '`root`'; when done, change permissions so that only '`oracle`' and no other user has access:

```
# cd /etc
# mkdir –pv ORACLE/WALLETS/oracle
# chown –R oracle:oinstall ORACLE/*
# chmod –R 700 ORACLE/*
```

After setting the `ENCRYPTION_WALLET_LOCATION` parameter in `sqlnet.ora` to the newly created directory, create wallet and add master encryption key either via SQL*Plus command line:

```
SQL> alter system set encryption key identified by “password”;
```

or Enterprise Manager Database Control Web interface. After successful creation of wallet and master key, reduce permissions to the wallet from the initial value determined by `'umask'` for the `'oracle'` user to:

Copyright © 2010 Oracle USA, Inc. All rights reserved.

```
$ cd /etc/ORACLE/WALLETS/oracle
$ chmod 600 ewallet.p12
```

It is highly recommended to always backup the wallet at the same time when backing up your database, but do not include the wallet with the database backup! Also, backup the wallet before **any** manipulation of it's content.

(Local) auto-open vs. encryption wallet

The encryption wallet (`'ewallet.p12'`) offers strong protection of the master key, by encrypting the wallet with the wallet password (based on PKCS#5, which also implies that the wallet password is not stored anywhere and hence, cannot be recovered if lost). It requires human intervention to open the wallet in order to make the master encryption key available to the database after the database has been started. For un-attended, 'lights-out' operations, the master key can be copied into an 'auto-open' wallet. This can be done either using Oracle Wallet Manager or the orapki command line interface:

```
$ orapki wallet create –wallet <wallet_location> -auto_login
```

This creates the auto-open wallet ('`cwallet.sso`'). In order to significantly strengthen your security when using an auto-open wallet, a **local** auto-open wallet can be created in Oracle Database 11.1.0.7; it does not open on any other machine than the one it was created on:

```
$ orapki wallet create –wallet <wallet_location> -auto_login_local;
```

Do not delete the encryption wallet, since otherwise re-key operations of the master encryption key fail. Re-keying the master key in the encryption wallet automatically updates the master key in the auto-open wallet.

Backing up the auto-open wallet **away from the encrypted data is even more important** than with the password-protected encryption wallet; should an auto-open wallet be found with the encrypted data, encryption is worthless.

Strong wallet password

In a worst-case scenario, all that stands between your sensitive, encrypted data and an intruder who happened to find the backup tape **and** the encryption wallet is the wallet password. It's easy to see that the password needs to be strong, yet easy to remember, since a forgotten wallet password cannot be recovered. One way to come up with a strong yet easy to remember password is to take the first characters of each word in an easy-to-remember sentence: "I work from 9 to 5 almost every day of the week" would give "Iwf9t5aedotw", which satisfies the common requirements for good passwords: It contains numbers as well as upper- and lower case characters, and it has 12 characters (a minimum of 10 is recommended), and the sentence is very easy to remember, while you don't have to remember the complex password itself at all.

Your security policies may dictate that a DBA is not allowed to know the wallet password; this can easily be accomplished in Enterprise Manager Database Control, where the password is displayed in a masked fashion. To tighten security even more, two or more people can enter a portion of a longer password in the right sequence; this way, the complete password will never be known, and you need to have a quorum of two or more people to open the wallet.

It is recommended to create the wallet and master key on the database server itself. Otherwise, when the command to create wallet and master key are issued on a remote machine, the wallet password would show in clear text in the local trace files. If you have to use a remote machine, at least enable Network Encryption between the client and database server so that the communication between both machines is secure.

Chapter 5 - Key management

Key management encompasses 4 segments: Key generation, key storage, key exchange/rotation, and key destruction/shredding:

Copyright © 2010 Oracle USA, Inc. All rights reserved.

Key generation

The DBA or Database Security Administrator (DSA) generates the TDE master encryption key automatically using Oracle's built-in pseudo Random Number Generator (RNG) using the SQL*Plus command line interface, or the Enterprise Manager Database Control GUI. If the master encryption key is stored in a Hardware Security Module (HSM), the master encryption key is generated by the HSM.

Key storage

The master key is stored in an external security module: Either the Oracle Wallet (default), or a Hardware Security Module. If the master encryption key is stored in the Oracle Wallet, directory and file permissions should be as strict as possible, and the wallet password needs to be well designed.

Key exchange / rotation

Starting with Oracle 11gR2, the unified master encryption key can be re-key in the Oracle Wallet and in an HSM, and it can be migrated from Oracle Wallet to HSM. In Oracle Database 11gR1 **TDE tablespace encryption**, neither master key nor individual tablespace keys can be re-keyed; see page 5 for a work-around.

Key destruction / shredding

Retired master keys inside the Oracle Wallet should never be deleted, since they are needed to read data from backup tapes, or data from redo and undo files that are encrypted with a retired master key.

Copyright © 2010 Oracle USA, Inc. All rights reserved.

Oracle Database 11g TDE tablespace encryption for Enterprise Tools 8.48 and higher
February 2010
Author: Peter A. Wahl
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

Copyright © 2009, Oracle. All rights reserved.
This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.