securosys

SWISS SECURITY TECHNOLOGIES
FOR COMMUNICATIONS SYSTEMS

Oracle 12 Transparent Database Encryption

Using Primus Hardware Security Module

Application Note

Primus HSM or CloudsHSM Integration Guide for
Oracle 12 Transparent Database Encryption using PKCS#11

Securosys SA, Förrlibuckstrasse 70
CH-8005 Zürich, Switzerland
Tel. +41 44 552 31 00 • www.securosys.com
info@securosys.com

Document Information and Revision Control

Version	Date	Author	Description, Changes
1	27.07.2021	PM	Initial draft
2	25.10.2021	PM	Moved from Primus PKCS#11 User Guide to separate AN

File: PrimusHSM_Oracle12-Integration_AN-E02.docx

Table of Contents

1 Introduction

This document describes how to integrate Securosys Primus HSM or CloudsHSM Service with Oracle 12 Transparent Database Encryption (TDE), to generate and use key material within the protected boundary of Securosys Hardware Security Modules.

1.1 Target Audience

This document is intended for HSM and Oracle DB administrators or integrators.

1.2 References and More Information

[1] Primus HSM, PKCS#11 Provider User Guide, Edition 18 or newer

[2] Primus HSM User Guide for v2.8/v2.11 Edition 13 or later

[3] Oracle Database documentation

[4] Securosys CloudsHSM Service
https://www.securosys.com/en/product/cloudshsm

All Securosys documentation is downloadable from the Securosys Support Portal.

1.3 Requirements

- OpenJDK 8
- Oracle 12 Database
- Securosys PKCS#11 Provider v1.8.6 or newer
- CloudsHSM Service (HSM as a Service) or
  Primus HSM, firmware v2.8.21, v2.9.2 or newer
  with PKCS#11 API and Session Object support enabled.

1.4 Software and Hardware used for this Documentation Setup

- Primus PKCS#11 provider v1.8.6

1.5 Notations and Symbols

Configuration files, command lines and their output are boxed using `Consolas` font. Commands are printed **bold**, values to adapt by the user are marked in **bold blue**, and comments are marked in *italic brown*, e.g.

```
ppin -a -e HSM_USERNAME
…
Provide setup password for 'HSM_USERNAME': <enter User Setup Password, no echo>
…
```

1.6 Support Contact

If you encounter a problem while installing/configuring the provider or integrating the HSM, make sure that you have read the referenced documentation. If you cannot resolve the issue, contact your supplier or Securosys Customer Support. The Securosys Support Portal for registered users is reachable under https://support.securosys.ch .

1.7 Glossary

Acronym	Definition
HSM	Hardware Security Module (physical or as a service)
CloudsHSM	HSM as a service, operated by Securosys
PKCS#11	Public-Key Cryptography Standard #11 – API to cryptographic tokens
AN	Application Note
UG	User Guide
PKI	Public Key Infrastructure
CA	Certification Authority
FIPS	Federal Information Processing Standard
CC	Common Criteria for IT Security Evaluation

2 Architecture Overview

Oracle Databases use authentication, authorization, and auditing mechanisms to secure data in the database, but not in the operating system files where the data is stored. To protect those files, Oracle Databases provide transparent data encryption (TDE). This feature enables you to protect sensitive data in database columns or files stored in operating system files by encrypting it. Then, to prevent unauthorized decryption, it stores encryption keys in a security module external to the database.

Securosys provides affordable network Hardware Security Modules, up to highest requirements in performance, availability, and safety. Start your integration today using the CloudsHSM service, not having to care about setup or operation, and the possibility of later migration to on-site infrastructure.

Benefit from fast regional access, load-balancing, and automatic redundancy failover, thanks to built-in large geo-redundant HA cluster mechanisms.

Multi-tenancy and large built-in storage allow separation of different instances, and usage for future applications.

Minimize risk of operational failures e.g. by additional Security Officer intervention for key deletion.

Ease and automate audit procedures using Securosys Key Attestation and Audit features, proofing all keys and relevant parameters with a chain of trust originating from our Securosys root certificate.

Decanus allows for easy and cost-efficient remote management of HSM clusters and CloudsHSM partitions (2-of-n, 2FA) without compromising security.

All HSMs are developed and manufactured in Switzerland using a trusted supply chain and are certified according FIPS140-2 level 3 and CC EAL4+ (EN419221-5) to fulfill strongest compliance regulations.

3 Integration Procedures

3.1 Securosys CloudsHSM or Primus HSM Setup and Configuration

Securosys CloudsHSM allows almost instant HSM operation by selecting and contracting the different services and options for your integration project:

- Standard or certified operation (CC EAL4+, FIPS 140-2 level 3)
- Shared or dedicated HSM
- Preferred datacenter region(s)
- Required integration API: PKCS#11
- Optional: Decanus remote management, up to full control
  (configuration, backup/restore, key attestation and audit for regulatory requirements, offline partition)

For available service packages and options consult our website [4] Securosys CloudsHSM Service and contact Securosys sales.

For on-premises Primus HSM hardware, HA Cluster setup and operation in FIPS or Common Criteria certified modes, refer to the corresponding [2] Primus HSM User Guide for details.
Check also [1] Primus HSM, PKCS#11 Provider User Guide, chapter 2, on how to enable PKCS#11 API, configuring the PKCS# Password/PIN, and enabling session objects and further settings.

3.2 Securosys Primus PKCS#11 Provider Installation and Configuration

The [1] Primus HSM, PKCS#11 Provider User Guide, downloadable from the Securosys Support Portal, describes in detail the installation and configuration of the provider for the different operating system platforms.

3.3 Oracle 12c

3.3.1 Supported Versions

The following platforms are tested with the Primus HSM

Platform	Oracle 12c	PKCS#11 Version
Oracle RHL 7.4 x64	Oracle 12c Release 2 12.2.0.1.0 Standard and Enterprise Edition	V1.8.6
CentOS 7.4 x64	Oracle 12c Release 2 12.2.0.1.0 Standard and Enterprise Edition	V1.8.6
Microsoft Windows x64	Oracle 12c Release 2 12.2.0.1.0 Standard and Enterprise Edition	V1.8.6

Supports both, unique dedicated partitions per Oracle database/cluster and a single shared partition configuration for all Oracle databases/clusters:

- New installation: only TDE is created
- Key migration: new TDE is created

3.3.3 Oracle Database Setup

The Oracle Database must be installed on the target machine to carry on with the integration process. Refer to the Oracle Database Documentation for a detailed installation procedure of Oracle Database.

3.3.4 Setup PKCS#11 for TDE

To set up Primus for Transparent Data Encryption, perform the following:

Copy the Primus HSM PKCS#11 library to the Oracle specified directory structure to ensure that the oracle database can find this library. Use the following directory structure:

```
/opt/oracle/extapi/64/hsm/primus/1.x.x/libprimusP11.so for Unix
```

```
%SYSTEMDRIVE%\oracle\extapi\64\hsm\primus\1.x.x\primusP11.dll for Windows
```

Ensure that the above have r/w permission to the 'oracle' directory (`chmod -R 0755`).

Export the following environment variables for the oracle account:

```
export ORACLE_SID=orcl
export ORACLE_BASE=/u01/app/oracle
export ORACLE_HOME=$ORACLE_BASE/product/12.2.0/dbhome_1
export PATH=$PATH:$ORACLE_HOME/bin
export TNS_ADMIN=$ORACLE_HOME/network/admin
```

3.3.5 HSM Master Encryption Key (MEK)

To start using TDE, you need to have a Master Encryption Key (MEK) which is stored inside the Primus HSM. The MEK is used to encrypt or decrypt column/tablespace using keys inside the Primus HSM. The Primus HSM can be used in the following ways to protect the Master Encryption Key:

- An existing Master Encryption Key can be migrated onto the HSM
- A Master Encryption Key can be directly generated onto the HSM
- Configure TDE Auto-Login for use with the HSM

3.3.5.1 Software Wallet

Test that the software wallet works on your installation:

Start with a clean 'orcl' (ORACLE_SID) setup (always start with a clean setup as the DB instance will store internal status information about TDE).

We assume that the 'oracle' home account directory is defined in '/home/oracle'

Log in as 'oracle' user and create a 'wallet' directory in the home directory:

```
mkdir wallet
```

Edit the sqlnet.ora file:

```
vi $ORACLE_HOME/network/admin/sqlnet.ora
```

and add at the end of the file the following entry (do not put the directory location between quotes or double quotes (', ") as it will cause errors in the next steps):

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIREC-
TORY = /home/oracle/wallet)))
```

If your DB instance is not started, fire it up and make sure the user 'system' has key management rights (we will use the 'system' user throughout this example):

```
sqlplus sys/oracle as sysdba
startup;
GRANT ADMINISTER KEY MANAGEMENT TO SYSTEM;
commit;
```

Log on or connect as 'system' user to create the key store (the key store is a PKS#12 file protected with a password. The key store password used in the example is 'password'):

```
sqlplus system/oracle
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/home/oracle/wallet' IDENTIFIED BY
password;
```

Open the key store:

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY password;
```

Set the MEK in the key store. Note the 'WITH BACKUP' creates a backup of the key store:

```
ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY password WITH BACKUP USING
'backup';
```

Create a table in the DB instance:

```
CREATE TABLE FOO (ID NUMBER(5), BAR NUMBER(10));
```

Insert rows:

```
INSERT INTO FOO VALUES (001 , 10000);
INSERT INTO FOO VALUES (002 , 20000);
INSERT INTO FOO VALUES (003 , 30000);
```

Cipher the 'BAR' column in the 'FOO' table:

```
ALTER TABLE FOO MODIFY (BAR ENCRYPT);
```

Select the rows in the table. TDE returns clear text:

```
SELECT BAR FROM FOO;
```

List the ciphered columns in the DB instance:

```
SELECT * FROM DBA_ENCRYPTED_COLUMNS;
```

Display the software wallet properties:

```
SELECT * FROM V$ENCRYPTION_WALLET;
```

Create a ciphered table space:

```
CREATE TABLESPACE PRIMUS DATAFILE '/u01/app/oracle/oradata/orcl/PRIMUS.DBF'
SIZE 150M ENCRYPTION DEFAULT STORAGE (ENCRYPT);
```

Create a table in the table space:

```
CREATE TABLE CUSTOMERS (ID NUMBER(5),NAME VARCHAR(40), BALANCE NUMBER(10)) TA-
BLESPACE PRIMUS;
```

Insert values:

```
INSERT INTO CUSTOMERS VALUES (001,'Oracle',10000);
INSERT INTO CUSTOMERS VALUES (002,'Microsoft',15000);
INSERT INTO CUSTOMERS VALUES (003,'IBM',20000);
```

Select table content:

```
SELECT * FROM CUSTOMERS;
```

Close the key store:

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY password;
```

Try to run a select, you will get an error:

```
SELECT * FROM CUSTOMERS;
```

3.3.5.2 Migrate the Master Encryption Key to the HSM

Edit the `sqlnet.ora` file:

```
vi $ORACLE_HOME/network/admin/sqlnet.ora
```

Add at the end of the file the following entry (do not put the directory location between quotes or double quotes (‘,“) as it will cause errors in the next steps):

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY
= /home/oracle/wallet)))
```

Restart the DB:

```
sqlplus sys/oracle as sysdba
shutdown immediate;
startup;
```

Login as ‘system’:

```
sqlplus system/oracle
```

Migrate the software wallet to the Primus HSM:

```
ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY "WZIFY-TW30F-0KYQT-IT-
BMB-MBK14" MIGRATE USING password WITH BACKUP USING 'backup';
```

Using the HSM MEK, display the table columns in clear text:

```
SELECT BAR FROM FOO;
```

Close the key store:

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY "WZIFY-TW30F-0KYQT-IT-
BMB-MBK14";
```

At this point, you can create an AUTO LOGIN or LOCAL AUTO LOGIN wallet which will open automatically when you first connect. Refer to the Oracle documentation on how to create an auto-open wallet.

3.3.5.3 Use the Master Encryption Key on the HSM

Start with a clean DB instance (re-using the previous samples will likely not work unless you clean all internal DB wallet structures):

Edit the `sqlnet.ora` file:

```
vi $ORACLE_HOME/network/admin/sqlnet.ora
```

Add at the end of the file the following entry:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM))
```

Restart the DB:

```
sqlplus sys/oracle as sysdba
shutdown immediate;
startup;
```

Login as 'system':

```
sqlplus system/oracle
```

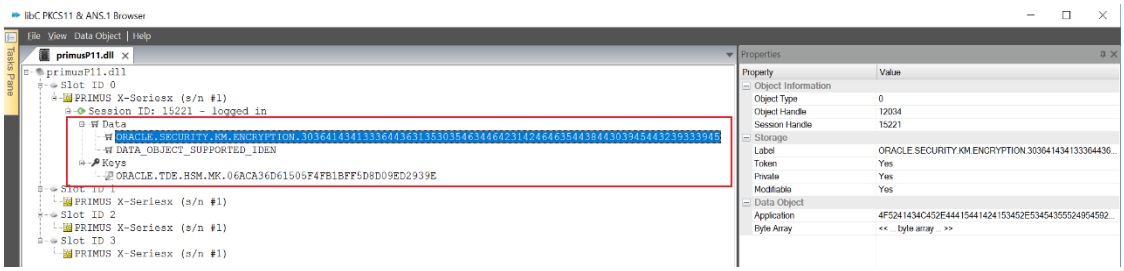
Open the Primus HSM key store:

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "WZIFY-TW30F-0KYQT-IT-
BMB-MBK14";
```

Set the Master Encryption Key:

```
ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "WZIFY-TW30F-0KYQT-ITBMB-
MBK14";
```

Display the HSM content (it is recommended you use one predefined slot in the HSM configuration: file as Oracle will select the last slotId in the ids configuration section)



Create a table in the DB instance:

```
CREATE TABLE FOO (ID NUMBER(5), BAR NUMBER(10));
```

Insert rows:

```
INSERT INTO FOO VALUES (001 , 10000);
INSERT INTO FOO VALUES (002 , 20000);
INSERT INTO FOO VALUES (003 , 30000);
```

Cipher the 'BAR' column in the 'FOO' table:

```
ALTER TABLE FOO MODIFY (BAR ENCRYPT);
```

Select the rows in the table. TDE returns clear text:

```
SELECT BAR FROM FOO;
```

List the ciphered columns in the DB instance:

```
SELECT * FROM DBA_ENCRYPTED_COLUMNS;
```

Display the software wallet properties:

```
SELECT * FROM V$ENCRYPTION_WALLET;
```

Create a ciphered table space:

```
CREATE TABLESPACE PRIMUS DATAFILE '/u01/app/oracle/oradata/orcl/PRIMUS.DBF'
SIZE 150M ENCRYPTION DEFAULT STORAGE (ENCRYPT);
```

Create a table in the table space:

```
CREATE TABLE CUSTOMERS (ID NUMBER(5),NAME VARCHAR(40), BALANCE NUMBER(10)) TA-
BLESPACE PRIMUS;
```

Insert values:

```
INSERT INTO CUSTOMERS VALUES (001,'Oracle',10000);
INSERT INTO CUSTOMERS VALUES (002,'Microsoft',15000);
INSERT INTO CUSTOMERS VALUES (003,'IBM',20000);
```

Select table content:

```
SELECT * FROM CUSTOMERS;
```

Close the HSM key store:

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY "WZIFY-TW30F-0KYQT-IT-
BMB-MBK14";
```

At this point, selecting rows from one of the tables will cause an error.
From here on, you can create an AUTO LOGIN or LOCAL AUTO LOGIN wallet, which will open automatically when you first connect. Refer to the Oracle documentation on how to create an auto-open wallet. HSM auto-open wallets requires a PKCS#12 with a specified client password identifier containing the HSM password. This requires editing the `sqlnet.ora` file and restarting the DB instance.

3.3.5.4 Pluggable Databases

The **multitenant architecture** enables an Oracle database to function as a multitenant container database (CDB).

A CBD includes zero, one, or many customer-created pluggable databases (PDBs). APDB is a portable collection of schemas, schema objects, and non-schema objects that appears to an Oracle Net client as a non-CDB. All Oracle databases before Oracle Database 12c were non-CDBs.

A container is either a PDB or the root. The root container is a collection of schemas, schema objects, and non-schema objects to which all PDBs belong.

Every CDB has the following containers:

- **Exactly one root**
  The root stores Oracle-supplied metadata and common users. An example of metadata is the source code for Oracle-supplied PL/SQL packages. A common user is a database user known in every container. The root container is named CDB$ROOT.
- **Exactly one seed PDB**
  The seed PDB is a system-supplied template that the CDB can use to create new PDBs. The seed PDB is named. PDB$SEED. You cannot add or modify objects in PDB$SEED.

- **Zero or more user-created PDBs**
  A PDB is a user-created entity that contains the data and code required for a specific set of features. For example, a PDB can support a specific application, such as a human resources or sales application. No PDBs exist at creation of the CDB. You add PDBs based on your business requirements.

To enable TDE when using pluggable databases, make sure that the key store for CDB (root container) is opened and that the Master Key for CDB is generated before opening the key store and generating the Master Key for PDB. Do not configure the HSM auto login for CDB until you generate the Master Key for PDB (all PDB in case multiple PDB are using the TDE). After generating the Master key for all PDBs you can configure the CDB for auto login and it will work for all PDBs as well.

3.3.6 Encrypting Tablespaces

Backup your Oracle databases

Note that you can only create encrypted tablespaces; you cannot modify existing tablespaces to encrypt them. So, when you need existing data in encrypted tablespaces, the best solution is to first create encrypted tablespaces and then move the objects from the unencrypted tablespaces to them.

3.3.7 Encrypting Existing Tables

Backup your Oracle Databases

If an existing table has columns that require encryption, then use the 'ALTER TABLE 'command in the following form:

```
ALTER TABLE table_name MODIFY ( column_name column_type ENCRYPT,...);
```