**Conestoga College**

**Reporting Systems And Database Development**

**(1517)**

**Implementing Data Security (PROG8640)**

**Group Assignment**

***Submitted By:***

***Basavraj Jaliminche (8800149)***

***Preeja Anilal(8791796)***

***Jino Wiseson Anbu Alaguraj- 8807543***

***kirtvir singh swatch (8800968)***

**AUG 8, 2022**

In this tutorial, you will enable SSL authentication on your Oracle database instance using Oracle Wallet Manager (owm).  Oracle Wallet Manager enables wallet owners to manage and edit the security credentials in their Oracle wallets. A wallet is a password-protected container used to store authentication and signing credentials, including private keys, certificates, and trusted certificates needed by SSL.

As the first step, we are going to create an Oracle Wallet to help configure the strong authentication parameters in our Oracle instance…

From the Windows start menu, type in Oracle Wallet and click on the icon that appears.  You should see the following dialog:

Open wallet manager from start> program

Graphical user interface, application

Description automatically generated

To create a New Wallet, go to the Wallet menu and click on New. Click on Yes on the next prompt to create a default wallet directory, and provided a password when prompted.

Graphical user interface, application

Description automatically generated

Choose **Operations** from the menu in Oracle Wallet Manager. Select **Create Certificate Request** to open the dialog box.

Graphical user interface, text, application, chat or text message

Description automatically generated

After filling the form, choose OK. A message informs you that the Certificate Request has been successfully created.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Certificate Request

-----BEGIN NEW CERTIFICATE REQUEST-----

MIIBuDCCASECAQAweDEWMBQGA1UEAxMNUHJlZWphIEFuaWxhbDEQMA4GA1UECxMH

ODc5MTc5NjEaMBgGA1UEChMRQ29uZXN0b2dhIENvbGxlZ2UxETAPBgNVBAcTCFdh

dGVybG9vMRAwDgYDVQQIEwdPbnRhcmlvMQswCQYDVQQGEwJDQTCBnzANBgkqhkiG

9w0BAQEFAAOBjQAwgYkCgYEArGbkGh6E6L3hmfqvFILgPyNRqIrCltkQ2w6hjwPI

prvTLNXeP47p5hIYizQ7W3emj+is2HOcIP/OyHmaqv/XNMRFEp5VDOQLZ3GprsXy

CYtgNaMBGYNxhu+Rd1eATXBoX/4KaYehmf6ZYBVXP4UvjSFKKzq6qEFzeDsOAHHP

9KsCAwEAAaAAMA0GCSqGSIb3DQEBCwUAA4GBAAt+dhSpGnKd/1fCxAIp945GdLmM

TKJzsTjTDih8AecqX5gmoIzpSfug1NGEtnuAaMVv79TOLLFJcOB8gSaUY+adh16r

7Wc0YzOzDP178Dcko3G6ZEBE32QPddB3udNlqfr6nyt33kFUWToX0H3iflixWHVC

tFHtnALItT77l/6c

-----END NEW CERTIFICATE REQUEST-----

After this step we can notice that the status of the certificate has changed to “Requested”. After you receive the certificate from SSL.com, you can install it in Oracle Wallet Manager. Oracle Wallet Manager stores X.509 certificates and private key s in PKCS #12 format and generates certificate requests according to the PKCS #10 specification. These capabilities make the Oracle wallet structure interoperable with supported third-party PKI applications and provide wallet portability across operating systems.

# Transparent Data Encryption

Transparent Data Encryption enables you to encrypt data. Typically, you encrypt sensitive data, such as credit card numbers or Social Security numbers.

Transparent Data Encryption (TDE) enables you to encrypt sensitive data that you store in tables and tablespaces.

After the data is encrypted, this data is transparently decrypted for authorized users or applications when they access this data. TDE helps protect data stored on media (also called data at rest) in the event that the storage media or data file is stolen.

Oracle Database uses authentication, authorization, and auditing mechanisms to secure data in the database, but not in the operating system data files where data is stored. To protect these data files, Oracle Database provides Transparent Data Encryption (TDE). TDE encrypts sensitive data stored in data files. To prevent unauthorized decryption, TDE stores the encryption keys in a security module external to the database, called a keystore.

You can configure Oracle Key Vault as part of the TDE implementation. This enables you to centrally manage TDE keystores (called TDE wallets in Oracle Key Vault) in your enterprise. For example, you can upload a software keystore to Oracle Key Vault and then make the contents of this keystore available to other TDE-enabled databases.

By default, Transparent Data Encryption (TDE) Column encryption uses the Advanced Encryption Standard with a 192-bit length cipher key (AES192).

In addition, [salt](https://docs.oracle.com/database/121/ASOAG/glossary.htm#GUID-6803F75B-7883-4D0B-B1FF-0084AD65E001) is added by default to plaintext before encryption unless specified otherwise. You cannot add salt to indexed columns that you want to encrypt. For indexed columns, choose the NO SALT parameter for the SQL ENCRYPT clause.

For Transparent Data Encryption (TDE) Tablespace encryption, the default is to use the Advanced Encryption Standard with a 128-bit length cipher key (AES128). In addition, salt is always added to plaintext before encryption.

You can change encryption algorithms and encryption keys on existing encrypted columns by setting a different algorithm with the SQL ENCRYPT clause.

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Key Size** | **Parameter Name** |
| triple Encryption Standard (DES) | 168 bits | 3DES168 |
| Advanced Encryption Standard (AES) | 128 bits | AES128 |
| AES | * Default for column level encryption is 192 bits * Default for tablespace encryption is 128 bits | * AES192 for column level encryption   AES128 for tablespace encryption |
| AES | 256 bits | AES256 |

For integrity protection of TDE column encryption, the SHA-1 hashing algorithm is used. If you have storage restrictions, then use the NOMAC option.

References:

* **https://docs.oracle.com/database/121/ASOAG/introduction-to-transparent-data-encryption.htm#ASOAG10139**