CHAPTER 5
TRANSPARENT DATA ENCRYPTION

Transparent Data Encryption enables you to encrypt data[38]. Typically, you encrypt sensitive data, such as credit card numbers or Social Security numbers.

5.1. What Is Transparent Data Encryption?

Transparent Data Encryption (TDE) enables you to encrypt sensitive data, such as Personally Identifiable Information (PII), that you store in tables and tablespaces[39]. After it 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 [40]. 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 our 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.
5.2. Benefits of Using Transparent Data Encryption

Transparent Data Encryption (TDE) provides benefits that enable you to ensure that sensitive data is encrypted, meets compliance, and provides functionality that streamlines our day-to-day encryption operations.

Benefits are as follows:

- As a security administrator, you can be sure that sensitive data is encrypted and therefore safe in the event that the storage media or data file is stolen.
- Using TDE helps you address security-related regulatory compliance issues.
- You do not need to create auxiliary tables, triggers, or views to decrypt data for the authorized user or application. Data from tables is transparently decrypted for the database user and application. An application that processes sensitive data can use TDE to provide strong data encryption with little or no change to the application.
- Data is transparently decrypted for database users and applications that access this data. Database users and applications do not need to be aware that the data they are accessing is stored in encrypted form.
- You can encrypt data with zero downtime on production systems by using online table redefinition or you can encrypt it offline during maintenance periods. You do not need to modify our applications to handle the encrypted data. The database manages the data encryption and decryption.
- Oracle Database automates TDE master encryption key[53] and keystore[53] management operations. The user or application does not need to manage TDE master encryption keys.
5.3. Who Can Configure Transparent Data Encryption?

Before you can create and manage keystores and encryption keys, you must be granted the ADMINISTER KEY MANAGEMENT system privilege, which you can use for most operations when the database is open. If you must open the keystore at the mount stage, then you must be granted the SYSKM administrative privilege, which includes the ADMINISTER KEY MANAGEMENT system privilege and other necessary privileges.

When you grant the SYSKM administrative privilege to a user, ensure that you create a password file for it so that the user can connect to the database as SYSKM using a password. This enables the user to perform actions such as querying the V$DATABASE view.

To configure TDE column or tablespace encryption, you do not need the SYSKM or ADMINISTER KEY MANAGEMENT privileges. You must have the following additional privileges to create TDE policies on tables and tablespaces:

- CREATE TABLE
- ALTER TABLE
- CREATE TABLESPACE

5.4. Types and Components of Transparent Data Encryption

Transparent Data Encryption can be applied to individual columns or entire tablespaces. Transparent Data Encryption (TDE) column encryption enables you to encrypt sensitive data stored in selected table columns. TDE tablespace encryption enables you to encrypt all of the data stored in a tablespace.

Both TDE column encryption and TDE tablespace encryption use a two-tiered key-based architecture. Unauthorized users, such as intruders who are attempting security attacks, cannot read the data from storage and back up media unless they have the TDE master encryption key to decrypt it.
5.5. How Transparent Data Encryption Column Encryption Works

Transparent Data Encryption (TDE) column encryption protects confidential data, such as credit card and Social Security numbers, that is stored in table columns. TDE column encryption uses the two-tiered key-based architecture to transparently encrypt and decrypt sensitive table columns. The TDE master encryption key is stored in an external security module, which can be an Oracle software keystore or hardware keystore. This TDE master encryption key encrypts and decrypts the TDE table key, which in turn encrypts and decrypts data in the table column.

As shown in Figure 5.1, the TDE master encryption key is stored in an external security module that is outside of the database and accessible only to a user who was granted the appropriate privileges. For this external security module, Oracle Database uses an Oracle...
software keystore (wallet, in previous releases) or hardware security module (HSM) keystore. Storing the TDE master encryption key in this way prevents its unauthorized use.

Using an external security module separates ordinary program functions from encryption operations, making it possible to assign separate, distinct duties to database administrators and security administrators. Security is enhanced because the keystore password can be unknown to the database administrator, requiring the security administrator to provide the password.

When a table contains encrypted columns, TDE uses a single TDE table key[54] regardless of the number of encrypted columns. Each TDE table key is individually encrypted with the TDE master encryption key. All of the TDE table keys are located together in the colklc column of the ENCS$ data dictionary table. No keys are stored in plaintext.

5.6. How Transparent Data Encryption Tablespace Encryption Works

Transparent Data Encryption (TDE) tablespace encryption enables you to encrypt an entire tablespace. All of the objects that are created in the encrypted tablespace are automatically encrypted. TDE tablespace encryption is useful if our tables contain sensitive data in multiple columns, or if you want to protect the entire table and not just individual columns. You do not need to perform a granular analysis of each table column to determine the columns that need encryption.

In addition, TDE tablespace encryption takes advantage of bulk encryption and caching to provide enhanced performance. The actual performance impact on applications can vary.

TDE tablespace encryption encrypts all of the data stored in an encrypted tablespace including its redo data. TDE tablespace encryption does not encrypt data that is stored outside of the tablespace. For example, BFILE data is not encrypted because it is stored
outside the database. If you create a table with a BFILE column in an encrypted tablespace, then this particular column will not be encrypted.

All of the data in an encrypted tablespace is stored in encrypted format on the disk. Data is transparently decrypted for an authorized user having the necessary privileges to view or modify the data. A database user or application does not need to know if the data in a particular table is encrypted on the disk. In the event that the data files on a disk or backup media is stolen, the data is not compromised.

TDE tablespace encryption uses the two-tiered, key-based architecture to transparently encrypt (and decrypt) tablespaces. The TDE master encryption key is stored in an external security module (software or hardware keystore). This TDE master encryption key is used to encrypt the TDE tablespace encryption key, which in turn is used to encrypt and decrypt data in the tablespace[41].

![Figure 5.2 Overview of the TDE tablespace encryption process.](image)
The encrypted data is protected during operations such as JOIN and SORT. This means that the data is safe when it is moved to temporary tablespaces. Data in undo and redo logs is also protected.

TDE tablespace encryption also allows index range scans on data in encrypted tablespaces. This is not possible with TDE column encryption.

Oracle Database implements the following features to TDE tablespace encryption:

- It uses a unified TDE master encryption key for both TDE column encryption and TDE tablespace encryption.
- You can reset the unified TDE master encryption key. This provides enhanced security and helps meet security and compliance requirements.

5.7. How the Keystore for the Storage of TDE Master Encryption Keys Works

To control the encryption, you use a keystore and TDE master encryption key.

5.7.1. Keystore Storage of TDE Master Encryption Keys

Oracle Database provides a key management framework for Transparent Data Encryption that stores and manages keys and credentials. The key management framework includes the keystore to securely store the TDE master encryption keys and the management framework to securely and efficiently manage keystore and key operations for various database components.

The Oracle keystore stores a history of retired TDE master encryption keys, which enables you to change them and still be able to decrypt data that was encrypted under an earlier TDE master encryption key.
5.7.2. Benefits of the Keystore Storage Framework

The key management framework provides several benefits for Transparent Data Encryption.

- Enables separation of duty between the database administrator and the security administrator who manages the keys. You can grant the `ADMINISTER KEY MANAGEMENT` or `SYSKM` privilege to users who are responsible for managing the keystore and key operations.

- Facilitates compliance, because it helps you to track encryption keys and implement requirements such as keystore password rotation and TDE master encryption key reset or rekey operations.

- Facilitates and helps enforce keystore backup requirements. A backup is a copy of the password-based software keystore that is created for all of the critical keystore operations.

  You must make a backup of the keystore for all of the critical keystore operations. You must also make a backup of the TDE master encryption key before you reset or rekey this TDE master encryption key.

- Enables the keystore to be stored on an ASM file system. This is particularly useful for Oracle Real Application Clusters (Oracle RAC) environments where database instances share a unified file system view.

- Enables reverse migration from a Hardware Security Module (HSM) keystore to a file system-based software keystore. This option is useful if you must migrate back to a software keystore.
5.7.3. Types of Keystores

Oracle Database supports both software keystores and hardware, or HSM-based, keystores. Software keystores can be stored on ASM disk groups or in a regular file system.

You can configure the following types of software keystores:

- **Password-based software keystores**: Password-based software keystores are protected by using a password that you create. You must open this type of keystore before the keys can be retrieved or used.
- **Auto-login software keystores**: Auto-login software keystores are protected by a system-generated password, and do not need to be explicitly opened by a security administrator. Auto-login software keystores are automatically opened when accessed. Auto-login software keystores can be used across different systems. If our environment does not require the extra security provided by a keystore that must be explicitly opened for use, then you can use an auto-login software keystore. Auto-login software keystores are ideal for unattended scenarios.
- **Local auto-login software keystores**: Local auto-login software keystores are auto-login software keystores that are local to the computer on which they are created. Local auto-login keystores cannot be opened on any computer other than the one on which they are created. This type of keystore is typically used for scenarios where additional security is required (that is, to limit the use of the auto-login for that computer) while supporting an unattended operation.

Hardware Security Modules are physical devices that provide secure storage for encryption keys, in hardware keystores. HSMs also provide secure computational space (memory) to perform encryption and decryption operations.

When using an HSM, all encryption and decryption operations that use the TDE master encryption key are performed inside the HSM. This means that the TDE master encryption key is never exposed in insecure memory.
5.8. Supported Encryption and Integrity Algorithms

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Key size</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triple DES (Data Encryption Standard)</td>
<td>168 bits</td>
<td>3DES168</td>
</tr>
<tr>
<td>AES (Advanced Encryption Standard)</td>
<td>128 bits</td>
<td>AES128</td>
</tr>
<tr>
<td>AES</td>
<td>192 bits (default)</td>
<td>AES192</td>
</tr>
<tr>
<td>AES</td>
<td>256 bits</td>
<td>AES256</td>
</tr>
</tbody>
</table>

Table 5.1 Supported Encryption Algorithms for Transparent Data Encryption