# Comprehensive Guide to Encryption and Decryption in Oracle Database

## 1. Introduction to Encryption in Oracle

Encryption is a critical technique used to protect sensitive data from unauthorized access. Oracle provides encryption functions through the DBMS\_CRYPTO package, which supports various encryption algorithms, including AES (Advanced Encryption Standard).

## 2. Prerequisites & Setup

Before implementing encryption and decryption in Oracle, ensure the following prerequisites:

1. The database user must have permission to use the DBMS\_CRYPTO package.

2. The encryption key must be securely stored and not hardcoded in queries or functions.

3. The database version must support DBMS\_CRYPTO (Oracle 10g and later).

Run the following command to grant the necessary privileges:

```sql  
GRANT EXECUTE ON DBMS\_CRYPTO TO your\_user;  
```

## 3. Creating the Encryption Function

The following function encrypts a given text using AES-128 encryption with CBC mode and PKCS5 padding.

```sql  
CREATE OR REPLACE FUNCTION encrypt\_text(p\_text IN VARCHAR2) RETURN RAW IS  
 v\_key RAW(16) := UTL\_RAW.CAST\_TO\_RAW('123456789abcdefg'); -- 16-byte key  
 v\_iv RAW(16) := UTL\_RAW.CAST\_TO\_RAW('abcdef9876543210'); -- 16-byte IV  
 v\_encrypted RAW(2000);  
BEGIN  
 v\_encrypted := DBMS\_CRYPTO.ENCRYPT(  
 src => UTL\_RAW.CAST\_TO\_RAW(p\_text),  
 typ => DBMS\_CRYPTO.ENCRYPT\_AES128 + DBMS\_CRYPTO.CHAIN\_CBC + DBMS\_CRYPTO.PAD\_PKCS5,  
 key => v\_key,  
 iv => v\_iv  
 );  
 RETURN v\_encrypted;  
END;  
/  
```

## 4. Creating the Decryption Function

The decryption function reverses the encryption process to retrieve the original text.

```sql  
CREATE OR REPLACE FUNCTION decrypt\_text(p\_encrypted IN RAW) RETURN VARCHAR2 IS  
 v\_key RAW(16) := UTL\_RAW.CAST\_TO\_RAW('123456789abcdefg'); -- 16-byte key  
 v\_iv RAW(16) := UTL\_RAW.CAST\_TO\_RAW('abcdef9876543210'); -- 16-byte IV  
 v\_decrypted VARCHAR2(2000);  
BEGIN  
 v\_decrypted := UTL\_RAW.CAST\_TO\_VARCHAR2(DBMS\_CRYPTO.DECRYPT(  
 src => p\_encrypted,  
 typ => DBMS\_CRYPTO.ENCRYPT\_AES128 + DBMS\_CRYPTO.CHAIN\_CBC + DBMS\_CRYPTO.PAD\_PKCS5,  
 key => v\_key,  
 iv => v\_iv  
 ));  
 RETURN v\_decrypted;  
END;  
/  
```

## 5. Creating a Table to Store Encrypted Data

Create a table to store employee salary information, with the salary column encrypted.

```sql  
CREATE TABLE emp\_sal\_details (  
 empid NUMBER PRIMARY KEY,  
 name VARCHAR2(100),  
 salary RAW(2000),  
 deptid NUMBER  
);  
```

## 6. Inserting and Retrieving Encrypted Data

Use the encrypt\_text function to store encrypted salary values.

```sql  
INSERT INTO emp\_sal\_details (empid, name, salary, deptid)  
VALUES (1, 'John Doe', encrypt\_text('75000'), 101);  
COMMIT;  
```

Retrieve and decrypt the salary data using the decrypt\_text function.

```sql  
SELECT empid, name, decrypt\_text(salary) AS salary, deptid FROM emp\_sal\_details;  
```

## 7. Best Practices for Secure Key Management

To ensure the security of encrypted data, follow these best practices:

1. \*\*Never hardcode encryption keys\*\* in stored procedures or functions.

2. \*\*Use Oracle Wallet or HSM (Hardware Security Module)\*\* to securely store keys.

3. \*\*Restrict access\*\* to encryption and decryption functions to authorized users only.

4. \*\*Regularly rotate encryption keys\*\* to enhance security.

5. \*\*Audit encryption usage\*\* to monitor and detect unauthorized access attempts.