Oracle Data Encryptio

Compliance with PCI DSS Regulatory Requirements

Robert Svoboda P.Eng.

Key 2 Data Ltd.

Calgary ORACLE Users Group

Introduction

- This presentation describes introduction of data encryption into Oracle databases and how “Transparent Data Encryption” in Oracle 11g can benefit DBAs in achieving compliancy with Payment Card Industry Data Security Standard.

Content

- Identification of threats
- Basic framework of Oracle security
- PCI requirements
- What is Encryption ?
- Encryption in Oracle:
  DBMS_OBFUSCATION_TOOLKIT,
  DBMS_CRYPTO,
  TDE
- Demo of Transparent Data Encryption

Identification of Threats

- What are the Common Security Threats ?
- Eavesdropping and Data Theft
- Data Tampering
- Falsifying User Identities
- Password Related Threats

Basic Framework of Oracle Security

- Securing database during installation
- Securing user accounts
- Managing user privileges
- Auditing database activity
- Securing network
- Securing data (encryption, VPD, Database Vault)

PCI Requirements

- What is Payment Card Industry Data Security Standard (PCI DSS) ?
- Founded by American Express, Visa, MasterCard, Discover Financial Services, and JCB
- The standards apply to all organizations that store, process or transmit cardholder data
- Any company processing, storing, or transmitting cardholder data must be PCI DSS compliant
- **https://www.pcisecuritystandards.org/**

The Core Elements of DSS

- Build and Maintain a Secure Network
- Protect Cardholder Data (encryption)
- Maintain a Vulnerability Management Program
- Implement Strong Access Control Measures
- Regularly Monitor and Test Networks
- Maintain an Information Security Policy

What is encryption ?

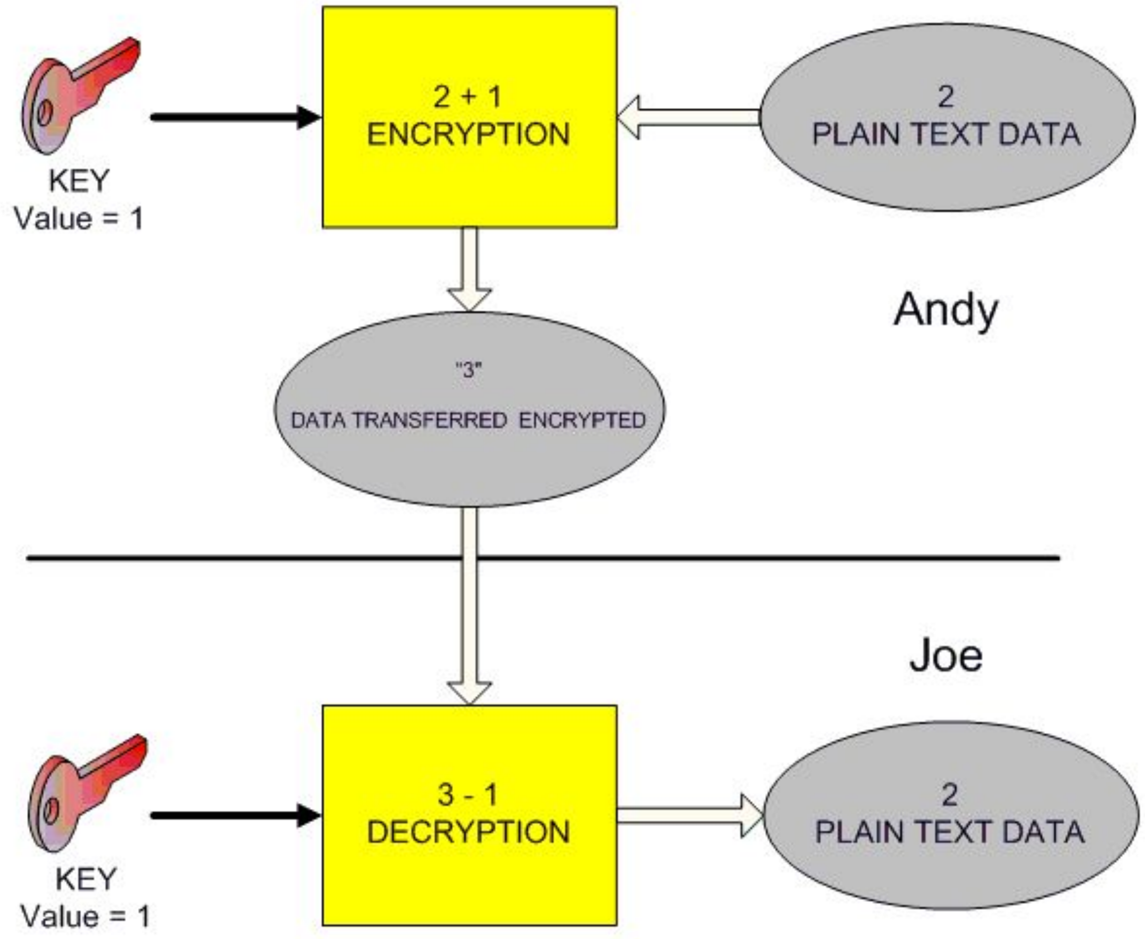
- Transformation of information using “encryption algorithm” into a form that can not be deciphered without a decryption key

Two types of encryption:

- Symmetric key encryption
- Public-key (asymmetric key) encryption

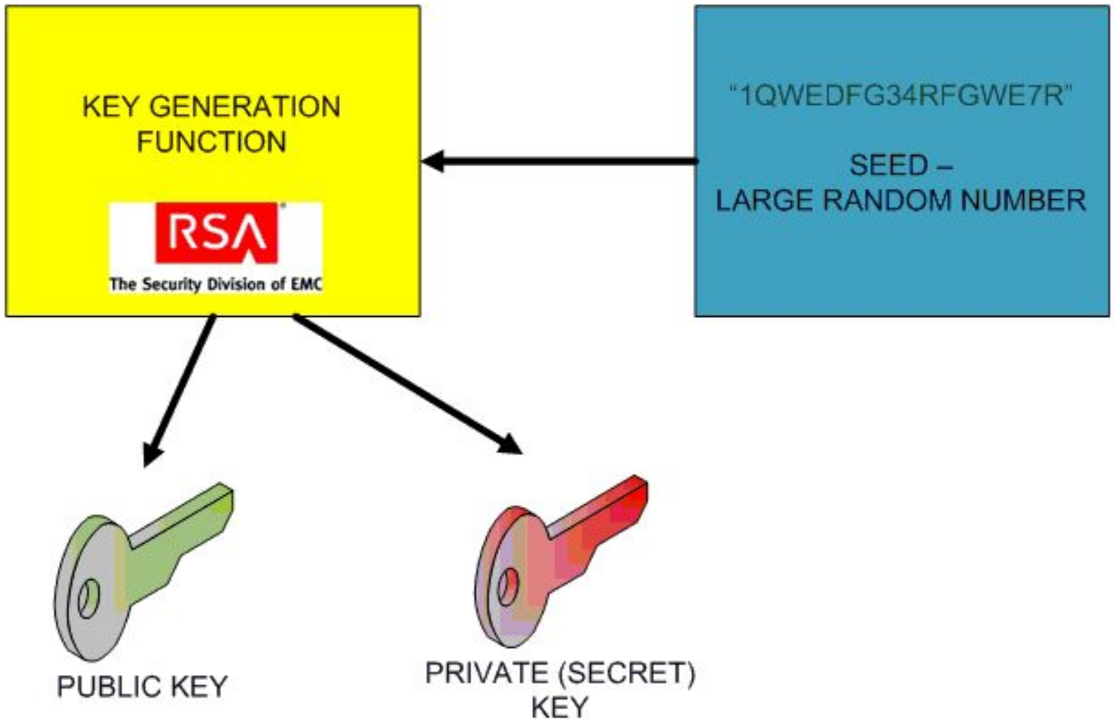
Symmetric Key Encryption

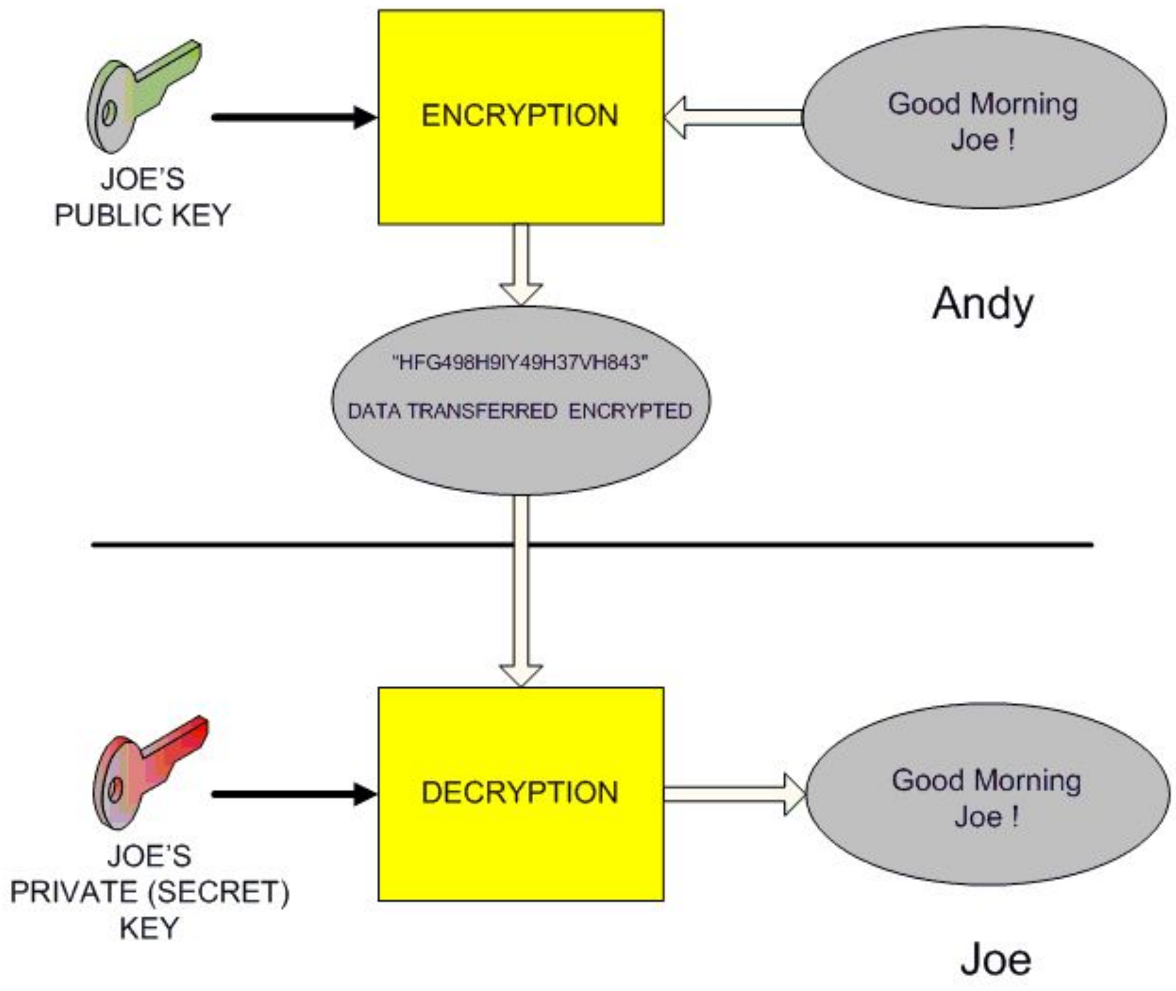
- Method in which both the sender and receiver share the same key


2 + 1
ENCRYPTION
KEY
Value = 1
2
PLAIN TEXT DATA
Andy
"3"
DATA TRANSFERRED ENCRYPTED
Joe
3 - 1
DECRYPTION
KEY
Value = 1
2
PLAIN TEXT DATA

Public Key Encryption

- The public key is freely distributed, while its paired private key remains secret
- The public key is typically used for encryption, while the private or secret key is used for decryption


KEY GENERATION FUNCTION
RSA
The Security Division of EMC
"1QWEDFG34RFGWE7R"
SEED –
LARGE RANDOM NUMBER
PUBLIC KEY
PRIVATE (SECRET) KEY


JOE'S PUBLIC KEY
ENCRYPTION
Good Morning Joe !
Andy
"HFG498H9IY49H37VH843"
DATA TRANSFERRED ENCRYPTED
JOE'S PRIVATE (SECRET) KEY
DECRYPTION
Good Morning Joe !
Joe

Encryption Algorithms Supported by Oracle

- RC4
- DES (Oracle 8 and 9)
- 3DES (Oracle 10)
- AES (Oracle 11)

DBMS_OBFUSCATION_TOOLKIT

- Introduced in Oracle 8i
- Uses DES algorithm

Syntax

- DBMS_OBFUSCATION_TOOLKIT.DES3Encrypt(
  ```
  input_string  IN VARCHAR2,
    key_string    IN VARCHAR2,
    which         IN PLS_INTEGER DEFAULT TwoKeyMode
    iv_string     IN VARCHAR2    DEFAULT NULL)  RETURN
  VARCHAR2;
  ```

- DBMS_OBFUSCATION_TOOLKIT.DES3DECRYPT(
  ```
    input_string  IN  VARCHAR2,
            key_string    IN  VARCHAR2,
            which         IN  PLS_INTEGER DEFAULT TwoKeyMode
            iv_string     IN  VARCHAR2    DEFAULT NULL)
        RETURN VARCHAR2;
  ```

Key Management

- Store the key in the database
- Store the key in the operating system
- Have the user manage the key

DBMS_CRYPTO

- Released in Oracle 10.1
- Supports AES
- Provides automatic padding
- Different options for block chaining
- Support for CLOB and BLOB
- Will deprecate dbms_obfuscation_toolkit

Real Life

- Both packages are complicated to use
- Key management represents a problem
- Encryption / decryption must be done through the application
- Not used as often as it should be
- Solution ?

Transparent Data Encryption (TDE)

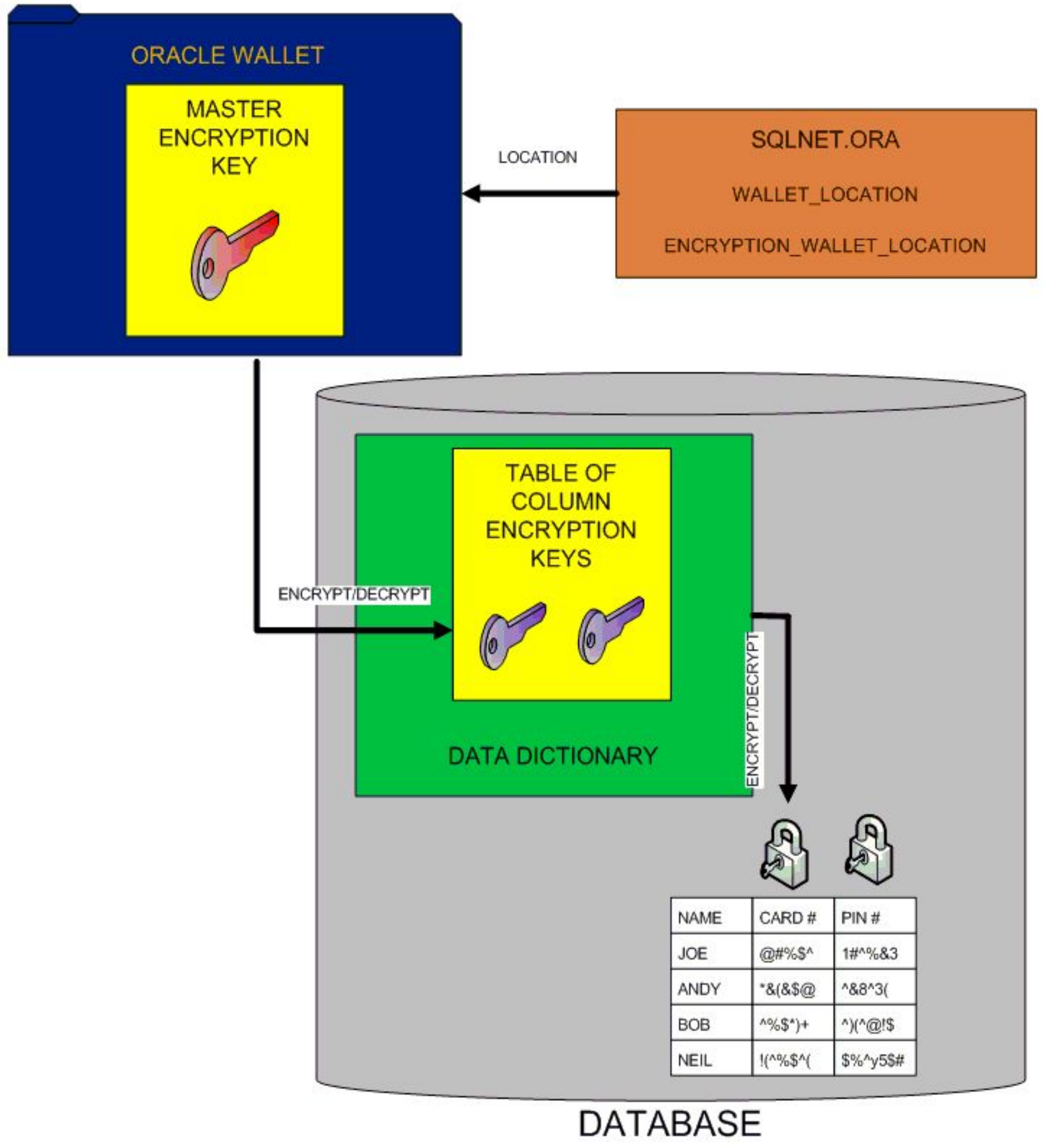
- Introduced in Oracle 10.2
  – column encryption
- Enhanced in Oracle 11.1
  - tablespace encryption

How is TDE Implemented?

1 Setup Wallet and Master Key
2 Identify columns with sensitive data
3 Review constraints
4 Encrypt existing and new data

Wallet

- Default wallet location
  $ORACLE_BASE/admin/$ORACLE_SID/wallet
- Alternative location specified in sqlnet.ora
  wallet_location
  encryption_wallet_location
- ewallet.p12
- Created by creating a new Master key:
  alter system set encryption key identified by "password ";
- Load the Master key into the database:
  alter system set encryption wallet open identified by "password";



NAME	CARD #	PIN #
JOE	@#%$^	1#^%&3
ANDY	*&(&$@	^&8^3(
BOB	^%$*)+	^)(^@!$
NEIL	!(^%$^(	$%^y5$#

Wallet Maintenance

- To disable all encryption columns in database: alter system set encryption wallet close;
- Wallet must be done after database restart:

  alter system set encryption wallet open authenticated by “password";
- Enable auto logging using Wallet Manager or mkwallet utility
- cwallet.sso

Wallet Backups

- Back up the wallet to a secure location (HSM), <u>separately</u> from the tape backups.
- Use RMAN backups which automatically excludes the wallet.
- During the OS backups exclude files *.p12 and *.sso

Column Encryption

- CREATE TABLE employee (name VARCHAR2(128), salary NUMBER(6) ENCRYPT);
- ALTER TABLE employee ADD (ssn VARCHAR2(11) ENCRYPT);
- ALTER TABLE employee MODIFY (first_name ENCRYPT);
- ALTER TABLE employee MODIFY (first_name DECRYPT);

Salt

- CREATE TABLE employee
  (name VARCHAR2(128),
  empID NUMBER ENCRYPT NO SALT,
  salary NUMBER(6) ENCRYPT USING
  '3DES168');

- CREATE INDEX employee_idx on
  employee (empID);

- You cannot create an index on a column that has been encrypted with salt.
- ORA-28338: cannot encrypt indexed column(s) with salt

Export / Import

- Must use Datapump

-
  ```
  expdp hr TABLES=emp
  DIRECTORY=dpump_dir
  DUMPFILE=dumpemp.dmp
  ENCRYPTION=ENCRYPTED_COLUMNS_ONLY
  ENCRYPTION_PASSWORD=pw2encrypt
  ```

-
  ```
  impdp hr TABLES=employee_data
  DIRECTORY=dpump_dir
  DUMPFILE= dumpemp.dmp
  ENCRYPTION_PASSWORD=pw2encrypt
  ```

- ENCRYPTION_MODE=DUAL
- ENCRYPTION_MODE=TRANSPARENT

Overheads

- 5 % – 35 % performance overhead
- Indexes are using encrypted values
- Each encrypted value needs 20 bytes for integrity check
- Encrypted value padded to 16 bytes
- If using salt, additional 16 bytes needed
- NOMAC parameter skips integrity check

```
ALTER TABLE employee MODIFY
(salary ENCRYPT 'NOMAC');
```

Incompatible Features

- Index types other than B-tree
- Range scan search through an index
- External large objects (BFILE)
- Materialized View Logs
- Transportable Tablespaces
- Original import/export utilities

TDE - Advantages

- Simple - can be done in four easy steps!
- Automatically encrypts database column data before it's written to disk
- Encryption and decryption is performed through the SQL interface
- No need for triggers to call encryption API's
- Views to decrypt data are completely eliminated
- Encryption is completely transparent to the application

TDE - Disadvantages

- Will not use indexes where the search criteria requires a range scan

“where

account number > 10000 or < 20000” will not work with TDE

- Indexes not possible if using ‘salt’
- Performance hit
- Requires more space

Data Dictionary Views

- DBA_ENCRYPTED_COLUMNS
- USER_ENCRYPTED_COLUMNS
- ALL_ENCRYPTED_COLUMNS
- V$RMAN_ENCRYPTION_ALGORITHMS
- V$ENCRYPTED_TABLESPACES
- V$ENCRYPTION_WALLET

Tablespace Encryption

- Compatibility = 11.0.0 or higher
- CREATE TABLESPACE encryptblspc DATAFILE '/u01/oradata/encryptblspc01.dbf‘ SIZE 200M ENCRYPTION USING '3DES168‘ DEFAULT STORAGE(ENCRYPT);
- DBA_TABLESPACES

Considerations

- Great for encrypting whole tables
- Objects automatically created encrypted
- All data encrypted including data in TEMP, UNDO, REDO (except BFILEs)
- Data protected during JOIN and SORT
- Allows index range scan
- Can not encrypt existing tablespace
- Use datapump, “create table as select”, “alter table move”
- Tablespace can not be enctypted with NO SALT option

Transparent Data Encryption cont.

- Example

Encryption in Practice

- Not a solution to all security problems
- Represents only one layer of Oracle security model
- Should be implemented in combination with Data Pump, RMAN, VPD and Data Masking
- PCI’s requirement to change regularly the encryption key is difficult to achieve
- Only as safe as your wallet
- With TDE there is no reason why your datafiles should stay unsecured

This presentation explained:

- What is data encryption
- Why sensitive data should be secured using encryption
- Demonstrated how TDE in Oracle 11 can help DBAs to encrypt data in an elegant and easy way

With Oracle 11g there is no reason to fail PCI audit !

Questions ?