Oracle TDE & A Case Study

Transparent Data Encryption

Database security & TDE

- ❑ Introduction to Database Security Issues
- ❑ Transparent Data Encryption
- ❑ Demo
- ❑ Conclusion

Introduction to Database Security Issues

- ➢ Threats to Database
- ➢ How to protect database?
- ➢ Introduction to DES & AES
  - ✓ History of DES & AES
  - ✓ Introduction to algorithm of DES & AES

Threats to Database

- ❖Loss of integrity
- ❖Loss of availability
- ❖Loss of confidentiality

How to protect database?

- ❖Access control
- ❖Inference control
- ❖Flow control
- ❖Data Encryption

Data Encryption Standard (DES) & Advance Encryption Standard (AES)

History:

- 15 May, 1973 NBS(National Bureau of Standards) — now named NIST (National Institute of Standards and Technology) publishes a first request for a standard encryption algorithm.
- 27 August, 1974 NBS publishes a second request for encryption algorithms
- 17 March, 1975 DES is published in the *Federal Register* for comment
- August, 1976 First workshop on DES
- September, 1976 Second workshop, discussing mathematical foundation of DES

Data Encryption Standard (DES) & Advance Encryption Standard (AES)

History:

- November, 1976 DES is approved as a standard

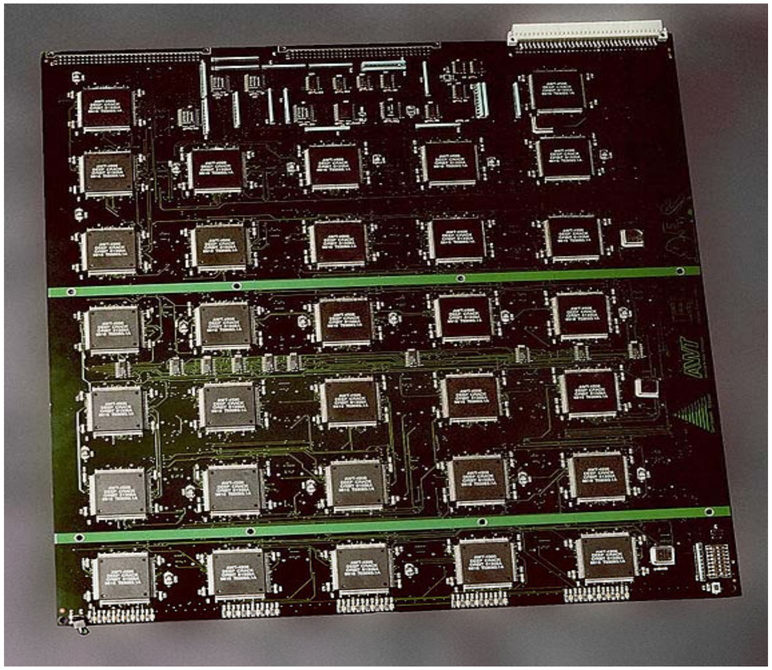
- 15 January, 1977 DES is published as a FIPS(Federal Information Processing Standard) standard FIPS PUB 46

- 22 January, 1988 DES is reaffirmed for the second time as FIPS 46-1, superseding FIPS PUB 46

- June, 1997 The DESCHALL Project breaks a message encrypted with DES for the first time in public.

- July 1998 The EFF(Electronic Frontier Foundation) 's DES cracker (Deep Crack) breaks a DES key in 56 hours.

Data Encryption Standard (DES) & Advance Encryption Standard (AES)



The EFF's US$250,000 DES cracking machine contained 1,856 custom chips and could brute force a DES key in a matter of days — the photo shows a DES Cracker circuit board fitted with several Deep Crack chips.

Data Encryption Standard (DES) & Advance Encryption Standard (AES)

History:

- January, 1999 Together, Deep Crack and distributed.net break a DES key in 22 hours and 15 minutes.

- 25 October, 1999 DES is reaffirmed for the fourth time as FIPS 46-3, which specifies the preferred use of Triple DES, with single DES permitted only in legacy systems.

- 26 November, 2001 The Advanced Encryption Standard is published in FIPS 197

- 26 May, 2002 The AES standard becomes effective

Data Encryption Standard (DES) & Advance Encryption Standard (AES)

History:

- 26 July, 2004 The withdrawal of FIPS 46-3 (and a couple of related standards) is proposed in the *Federal Register*

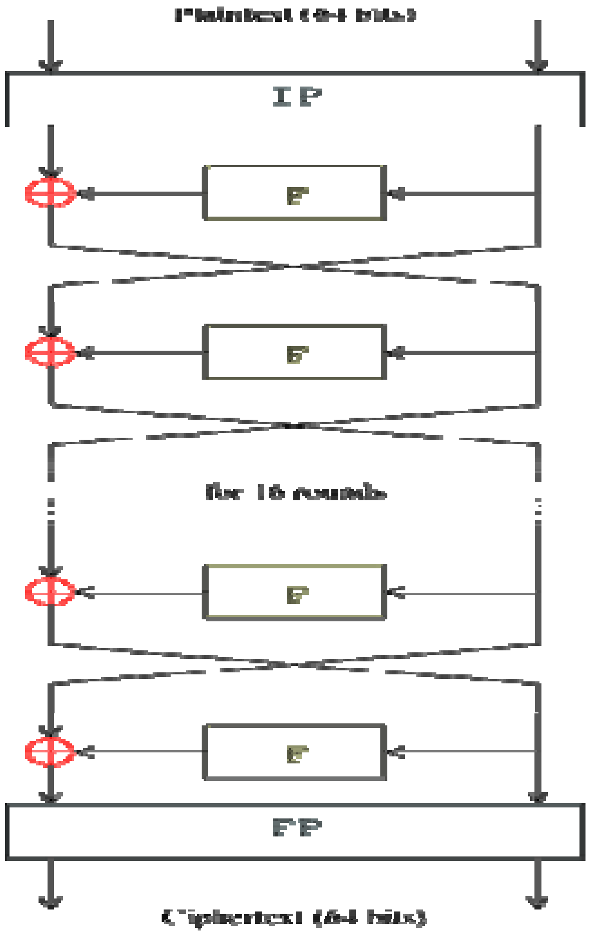
- 19 May 2005 NIST withdraws FIPS 46-3 (see Federal Register vol 70, number 96)

Introduction to Algorithm of DES

- DES is the archetypal block cipher

- In the case of DES, the block size is 64 bits

- The key length of DES is 64 bits; however, only 56 of these are actually used by the algorithm



Introduction to Algorithm of DES

The Feistel (F) function

Expansion

↓

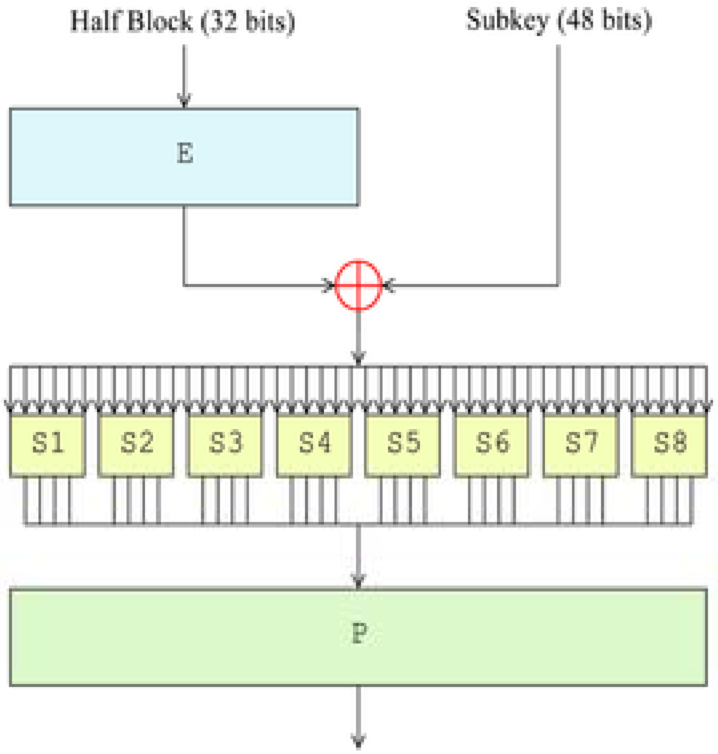
Key mixing

↓

Substitution

↓

Permutation



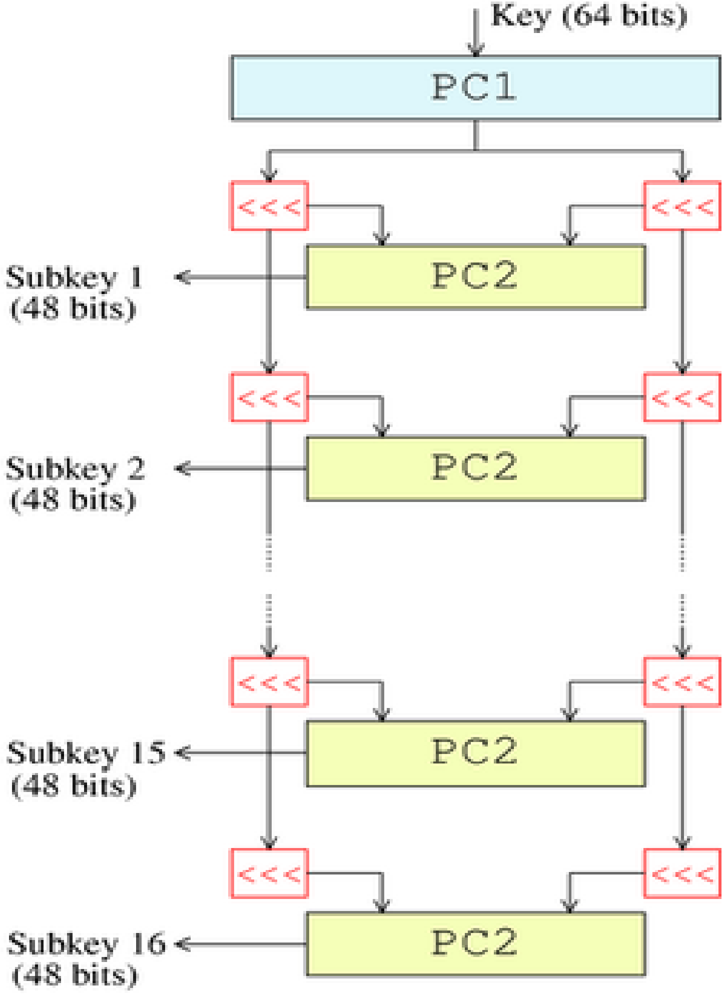
Introduction to Algorithm of DES

Key schedule

Initially, 56 bits of the key are selected from the initial 64 by *Permuted Choice 1* (*PC-1*)

The 56 bits are then divided into two 28-bit halves; each half is thereafter treated separately

Both halves are rotated left by one or two bits and then 48 subkey bits are selected by (*PC-2*)

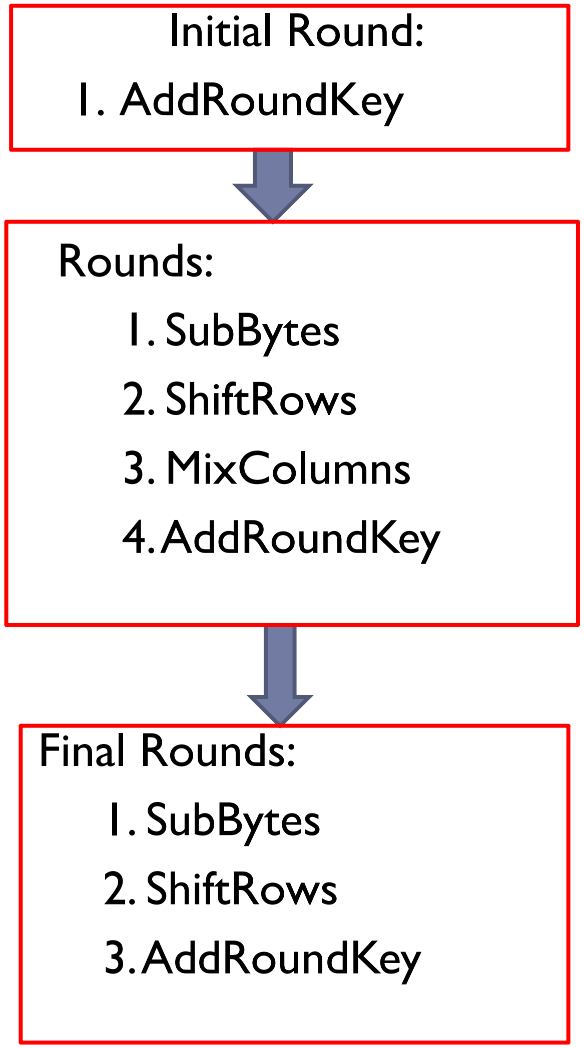


Introduction to Algorithm of AES

-AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits (10,12 and 14 rounds - depending on key size)

- AES operates on a 4×4 array of bytes and includes 4 basic steps in each round: AddRoundKey, SubBytes, ShiftRows and MixColumns

Introduction to Algorithm of AES



Initial Round:

1. AddRoundKey

Rounds:

1. SubBytes
2. ShiftRows
3. MixColumns
4. AddRoundKey

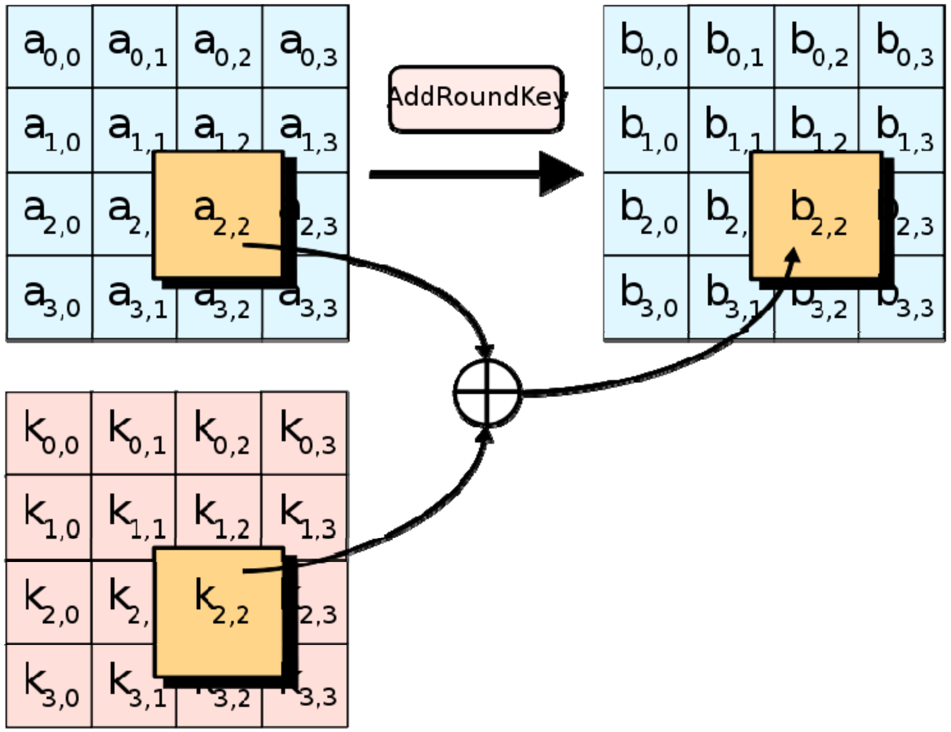
Final Rounds:

1. SubBytes
2. ShiftRows
3. AddRoundKey

Introduction to Algorithm of AES

AddRoundKey

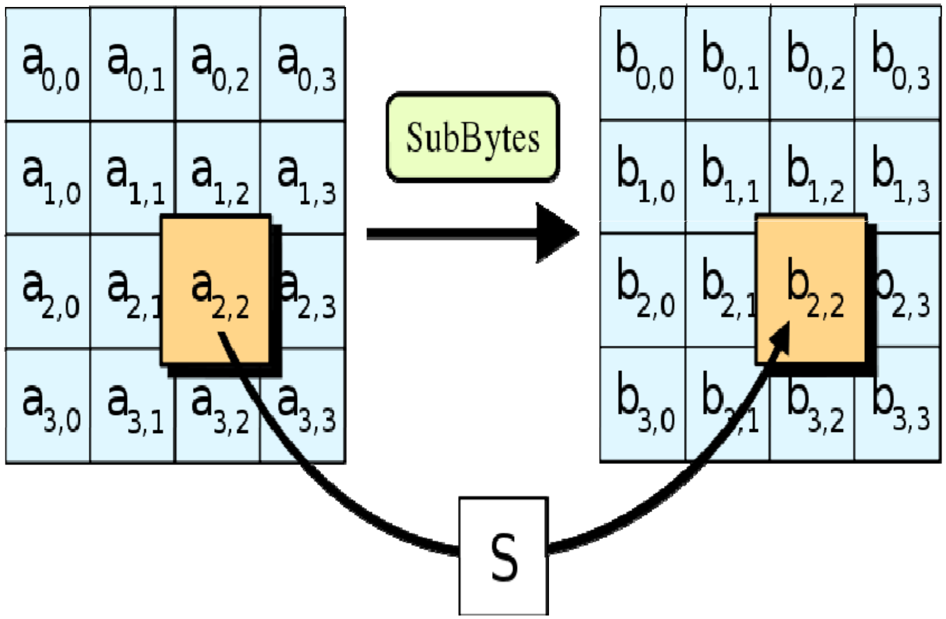
For each round, a subkey is derived from the main key using Rijndael’s key schedule


$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
AddRoundKey
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
$\oplus$
$k_{0,0}$ $k_{0,1}$ $k_{0,2}$ $k_{0,3}$
$k_{1,0}$ $k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,0}$ $k_{2,1}$ $k_{2,2}$ $k_{2,3}$
$k_{3,0}$ $k_{3,1}$ $k_{3,2}$ $k_{3,3}$

Introduction to Algorithm of AES

SubBytes

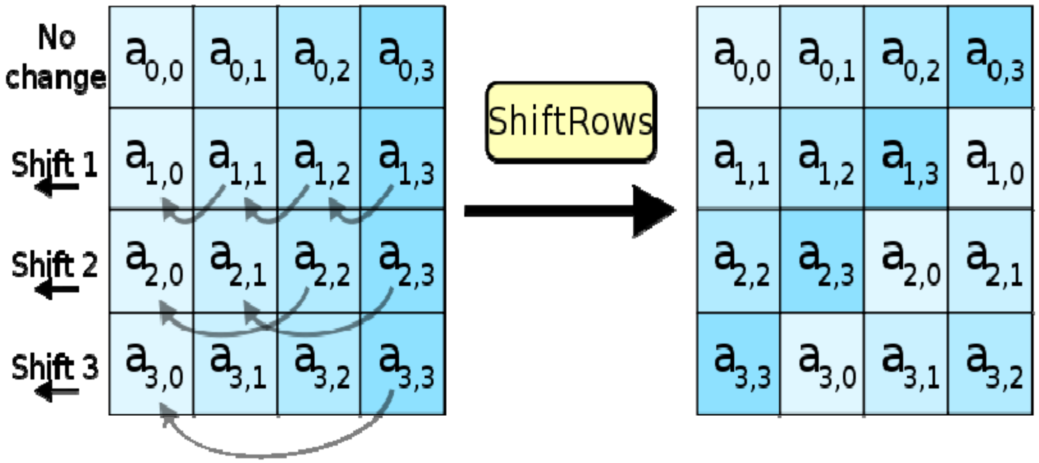
A non-linear substitution step where each byte is replaced with another according to a lookup table.


$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
SubBytes
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
S

Introduction to Algorithm of AES

ShiftRows

A transposition step where each row of the state is shifted cyclically a certain number of steps.



$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

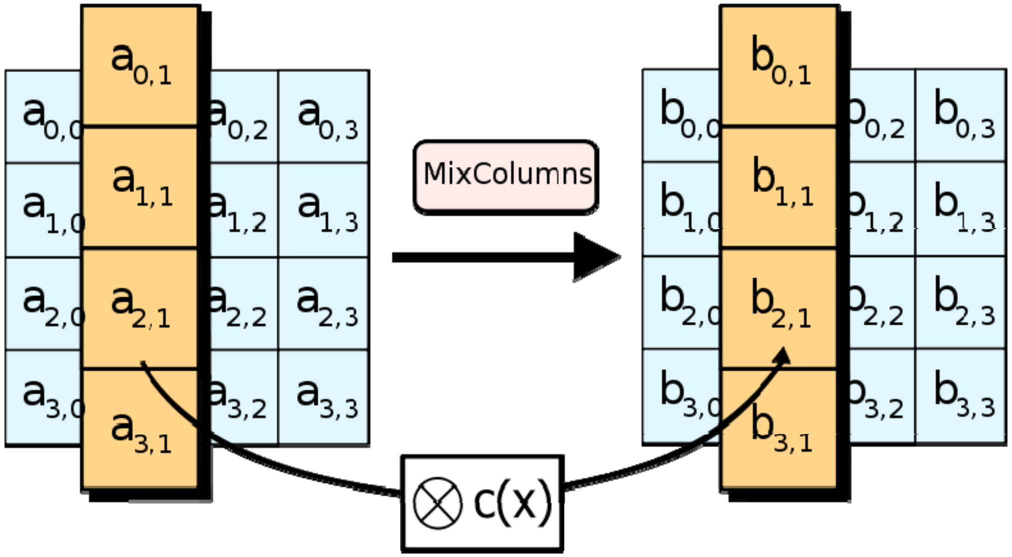
$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,0}$
$a_{2,2}$	$a_{2,3}$	$a_{2,0}$	$a_{2,1}$
$a_{3,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$

Introduction to Algorithm of AES

MixColumns

The four bytes of each column of the state are combined using an invertible linear information

$$C(x) = 3x^3 + x^2 + x + 2$$
$$(\text{modulo } x^4 + 1)$$



Database security & TDE

- ❑ Introduction to Database Security Issues
- ❑ Transparent Data Encryption
- ❑ Demo
- ❑ Conclusion

Transparent Data Encryption

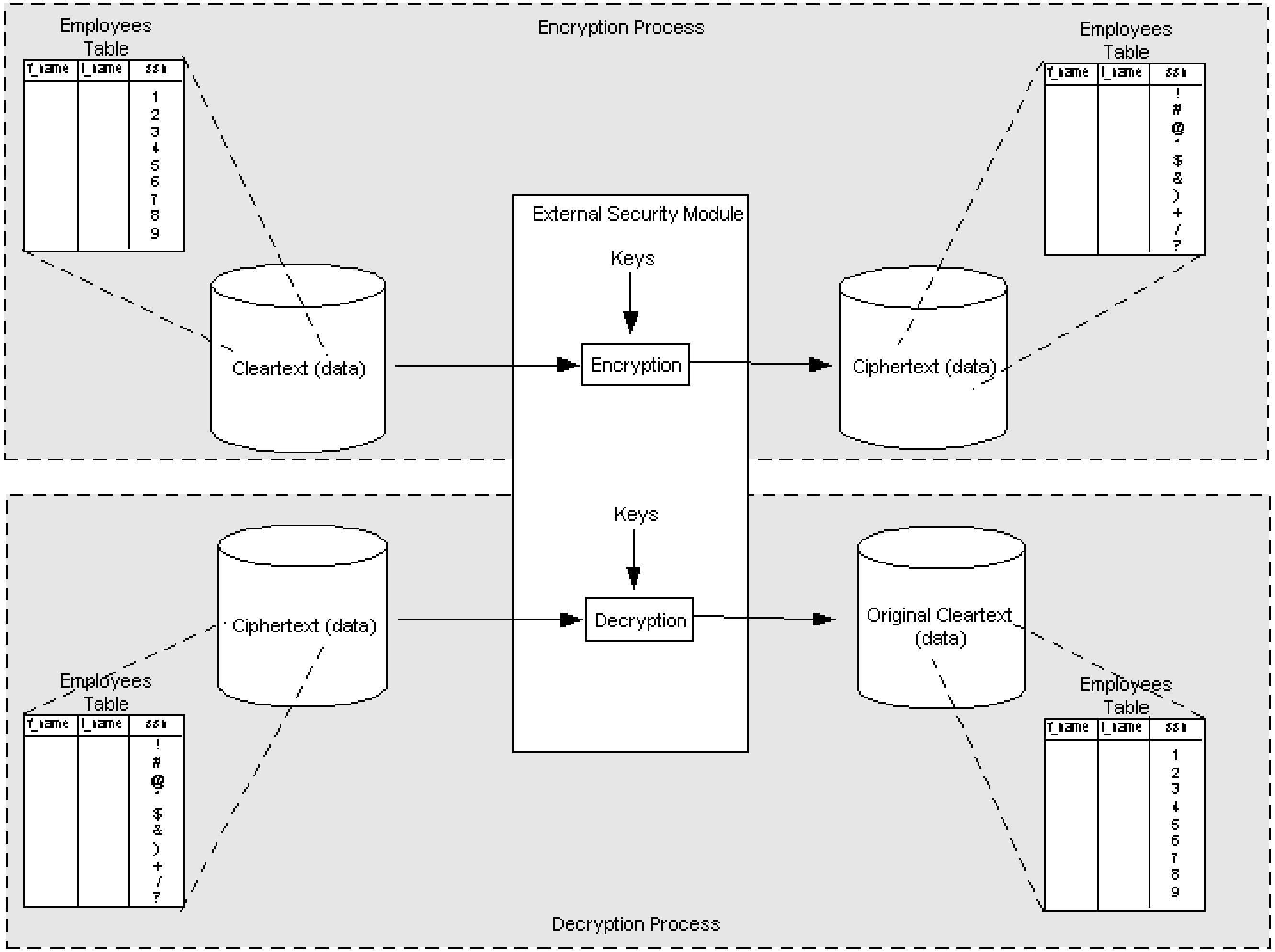
- ❑ About Transparent Data Encryption
- ❑ Using Transparent Data Encryption
- ❑ Managing Transparent Data Encryption

About Transparent Data Encryption

What Is The Transparent Data Encryption

A feature enables you to protect sensitive data in database columns 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.

About Transparent Data Encryption



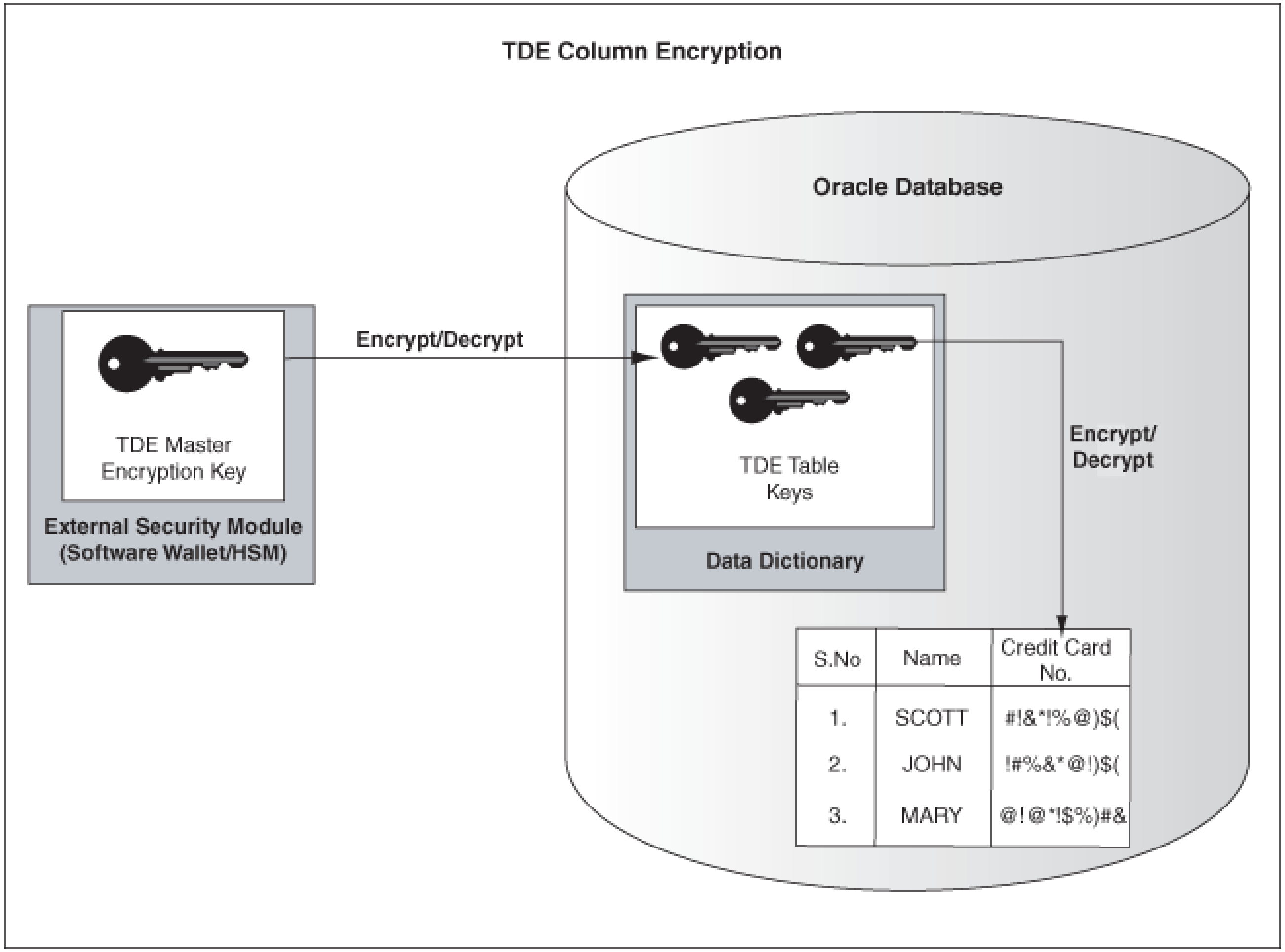
f_name	l_name	ssn
		1
		2
		3
		4
		5
		6
		7
		8
		9

f_name	l_name	ssn
		!
		#
		@
		'
		$
		&
		)
		+
		/
		?

f_name	l_name	ssn
		!
		#
		@
		'
		$
		&
		)
		+
		/
		?

f_name	l_name	ssn
		1
		2
		3
		4
		5
		6
		7
		8
		9

About Transparent Data Encryption



S.No	Name	Credit Card No.
1.	SCOTT	#!&*!%@)$(
2.	JOHN	!#%&*@!)$(
3.	MARY	@!@*!$%)#&

About Transparent Data Encryption

When to Use Transparent Data Encryption

Need to protect confidential data such as credit card,social security numbers vv…

When Do Not Use Transparent Data Encryption

- Range scan search through an index
- Large object datatypes such as BLOB and CLOB
- Original import/export utilities
- Other database tools and utilities that directly access data files

Transparent Data Encryption

❑ About Transparent Data Encryption

❑ Using Transparent Data Encryption

❑ Managing Transparent Data Encryption

Using Transparent Data Encryption

1. Specifying an Additional Wallet Location in SQLNET.ORA

```
ENCRYPTION_WALLET_LOCATION = (SOURCE
=(METHOD = FILE)(METHOD_DATA
=(DIRECTORY
=C:\oracle\dbsid\admin\pdcs11\wallet)))
```

2. Creating Wallets For Transparent Data Encryption

```
        ALTER SYSTEM SET ENCRYPTION KEY  IDENTIFIED
BY password
```

Using Transparent Data Encryption

3. Opening the Encrypted Wallet for Database Access to Encryption Keys.

```
ALTER SYSTEM SET ENCRYPTION WALLET
OPEN IDENTIFIED BY password
```

4. Setting and Resetting the Master Key.

```
ALTER SYSTEM SET ENCRYPTION KEY
IDENTIFIED BY password
```

Using Transparent Data Encryption

5. Creating Tables That Contain Encrypted Columns.

- CREATE TABLE employee (

```
                  First_name VARCHAR2(128),
                  last_name VARCHAR2(128),
                  empID NUMBER,
                  salary NUMBER(6) ENCRYPT);
```

- ALTER TABLE employee MODIFY (salary DECRYPT);

Using Transparent Data Encryption

6. Creating a Table with an Encrypted Column Using a Non-Default Algorithm and No Salt.

- CREATE TABLE employee (

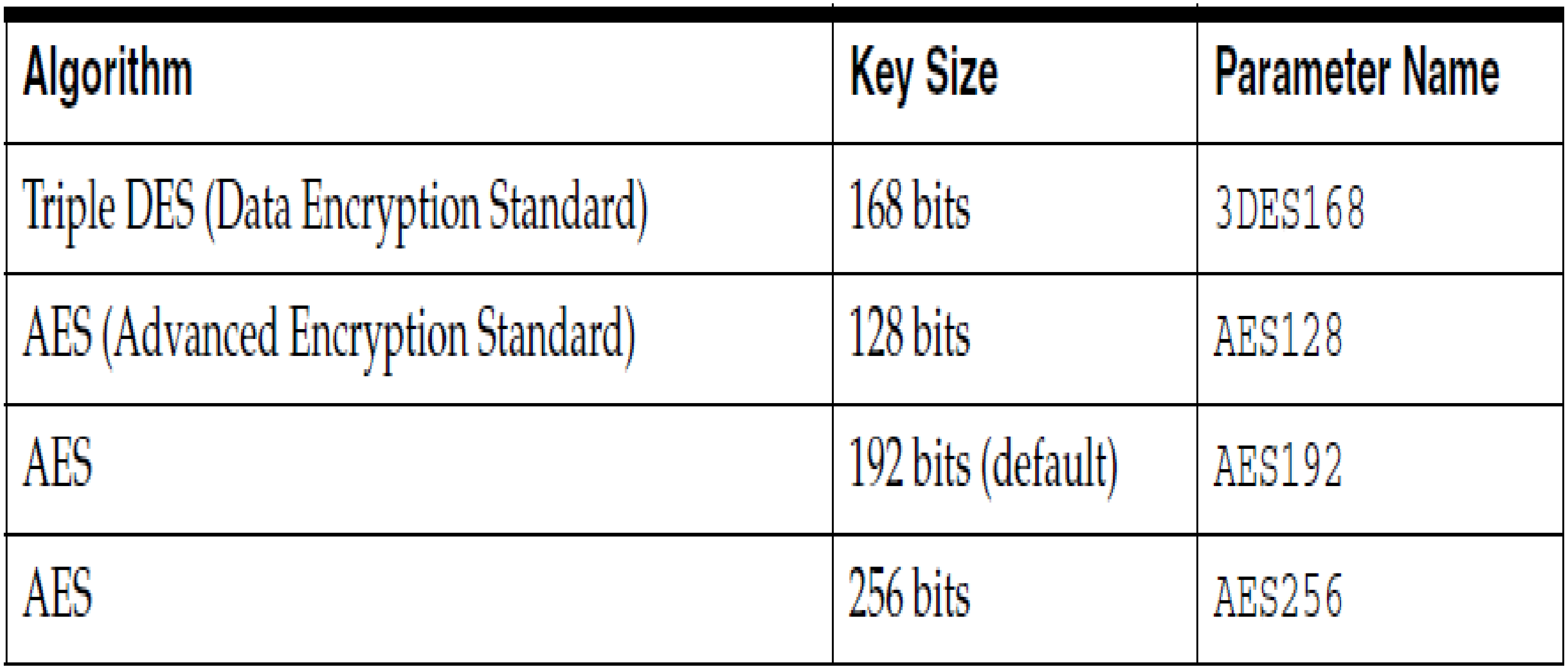
  first_name VARCHAR2(128),

  last_name VARCHAR2(128),

  empID NUMBER **ENCRYPT NO SALT,**

  salary NUMBER(6) **ENCRYPT USING ’3DES168’**);
- ALTER TABLE employee REKEY;
- ALTER TABLE employee REKEY USING ’3DES168’;

Supported Encryption and Integrity Algorithms



Algorithm	Key Size	Parameter Name
Triple DES (Data Encryption Standard)	168 bits	3DES168
AES (Advanced Encryption Standard)	128 bits	AES128
AES	192 bits (default)	AES192
AES	256 bits	AES256

Transparent Data Encryption

- ❑ About Transparent Data Encryption
- ❑ Using Transparent Data Encryption
- ❑ Managing Transparent Data Encryption

Managing Transparent Data Encryption

- ✓ Creating Wallets
- ✓ Specifying a Separate Wallet for Transparent Data Encryption
- ✓ Backup and Recovery of Master Keys
- ✓ Export and Import of Tables with Encrypted Columns
- ✓ Performance Effects of Transparent Data Encryption

Oracle TDE & A Case Study

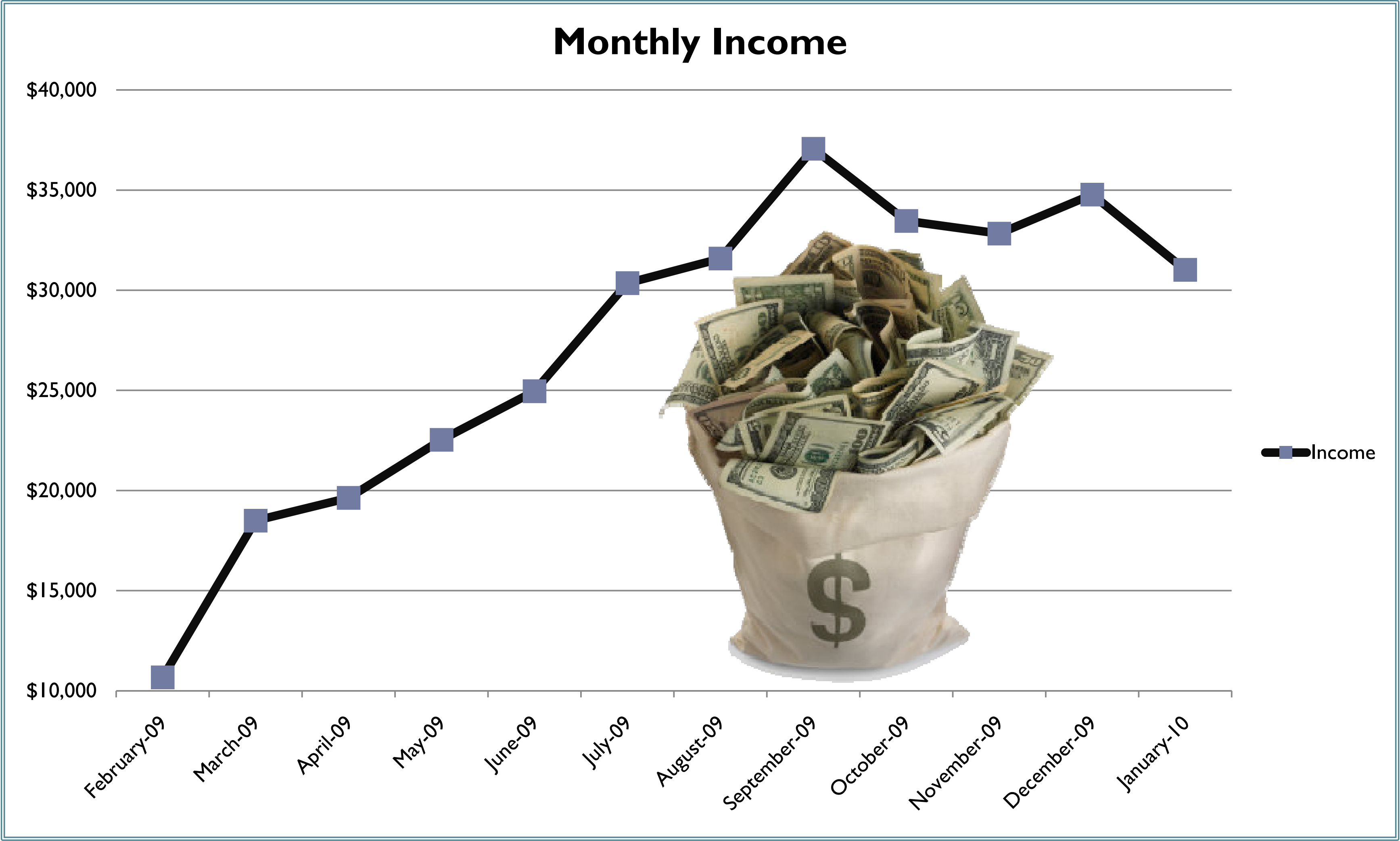
Demonstration

thanh.phamhong@niit.edu.vn

Company Overview

- Business Fields
  - Herbs for Dogs
  - Herbs for Cats
  - Herbs for People
- Online operations since 2004
- Potential customers:
  - From United States, Canada and United Kingdom
  - Age range: 40-80

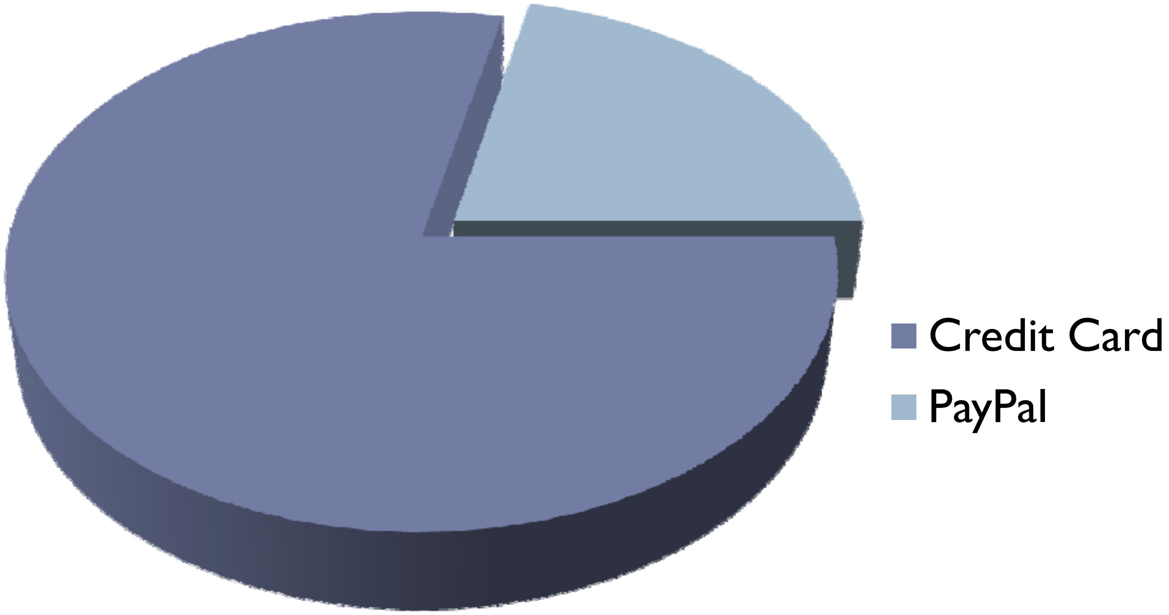
Summary Reports



Online Payment Options

- Credit Cards & PayPal

VISA MasterCard DISCOVER NETWORK AMERICAN EXPRESS PayPal



Customers from last 6 months

Business Problems

- Hack attempts in 2009:
  - ~ 3 attempts / month
  - Methods: mostly SQL Injection.

- California State Law - The California Online Privacy Protection Act of 2003 (OPPA)

- Payment Card Industry Data Security Standard (PCI DSS)

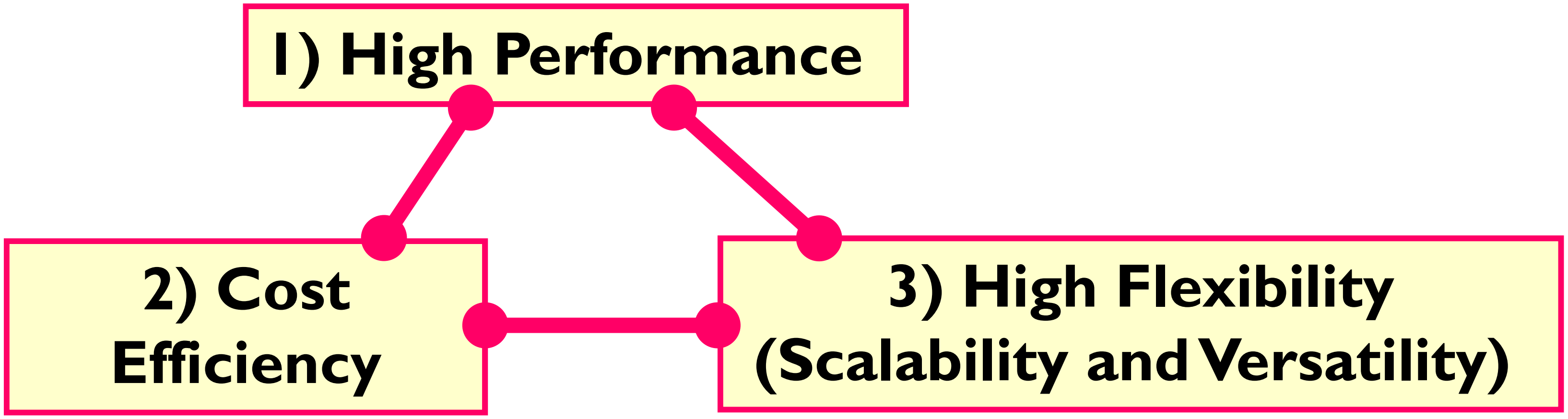
- Remember:
  - o Hackers are everywhere
  - o One lawsuit can put you under.

PCi Security Standards Council™
PARTICIPATING ORGANIZATION

Security Threats

- Improper Storage
- Insecure Transactions
- SQL Injection Attacks
- Software Vulnerabilities
- Spam/Phishing
- Poor Server Security
- Backups

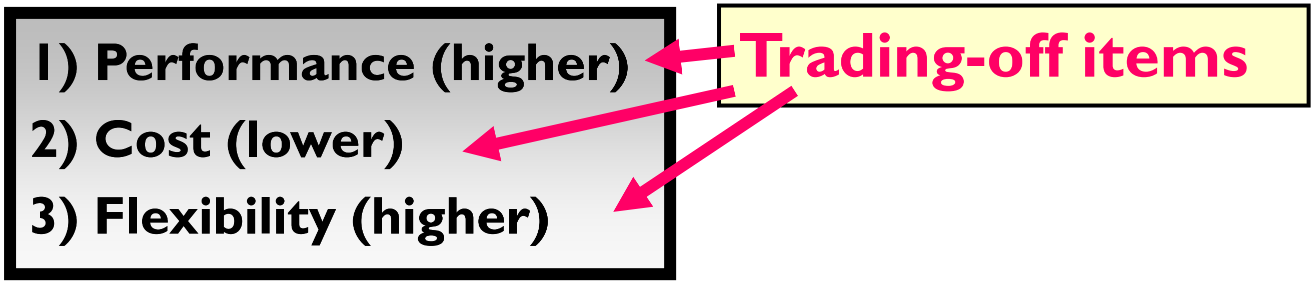
Changing Constraints



1) High Performance

2) Cost Efficiency

3) High Flexibility (Scalability and Versatility)



1) Performance (higher)

2) Cost (lower)

3) Flexibility (higher)

Trading-off items

Envisioned System

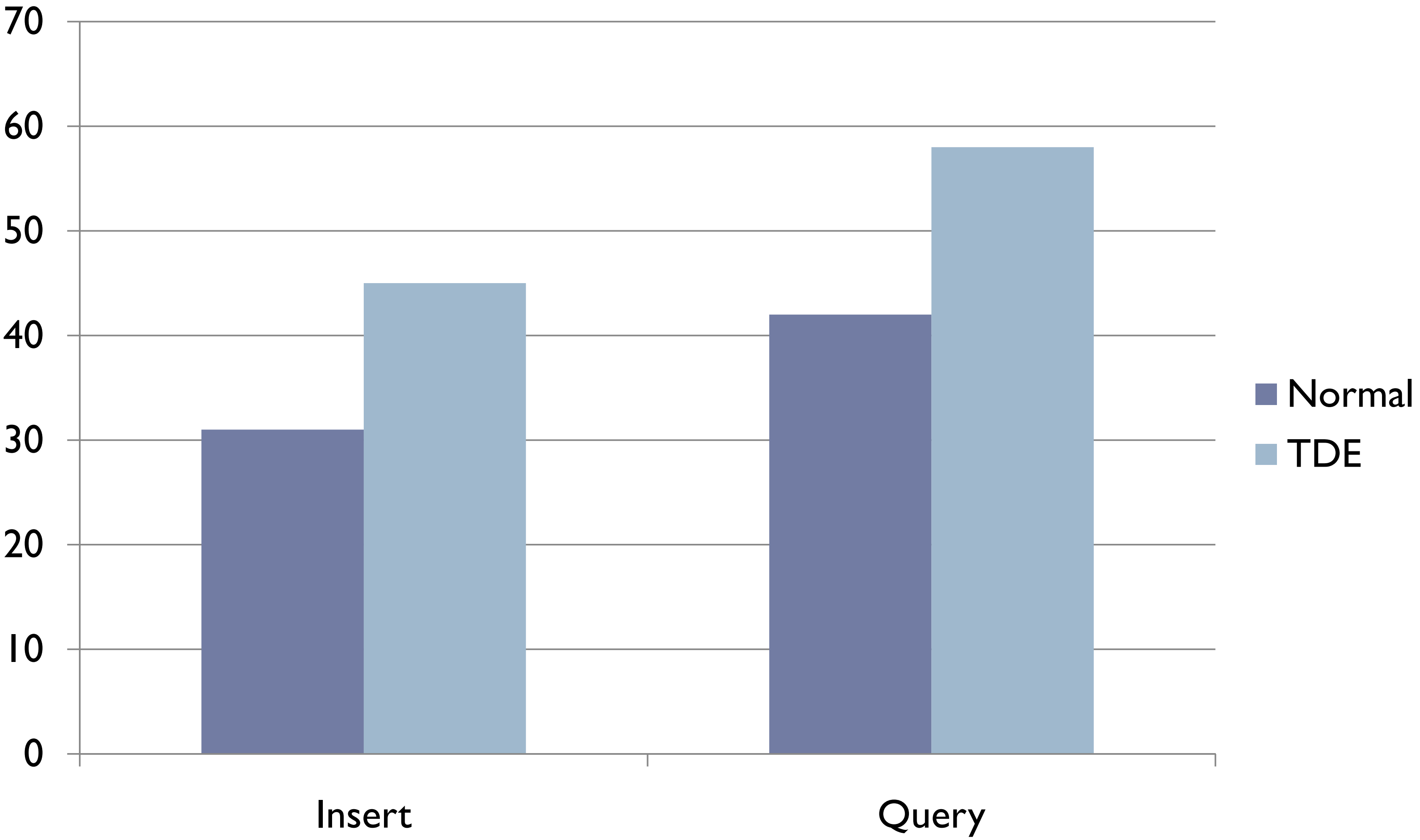
- SSL
- TDE

Changing Summary (for TDE only)

- Creates and opens the wallet
  - ALTER SYSTEM SET ENCRYPTION KEY AUTHENTICATED BY "myPassword";

- Tables changes:
  - ALTER TABLE CC MODIFY (CC_NUMBER ENCRYPT);

Performance Comparison



Conclusion

- Why?
- When?

Q&A

References

- [1] Oracle Press : Advanced Security Administrator, pp.55–85, 2005.

Thank You!