

## Securitatea bazelor de date – master anul 2

### Laborator 1

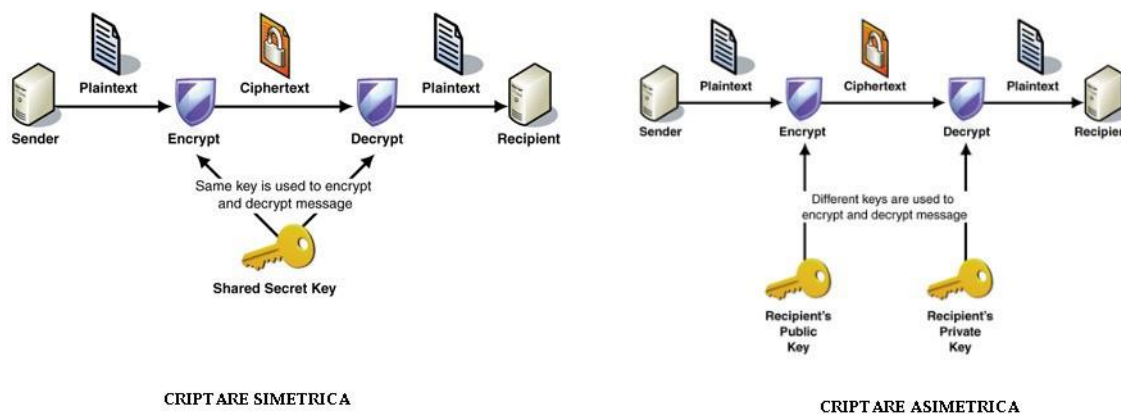
## Criptarea și decriptarea datelor într-o bază de date

Cuvinte cheie:

- |   |  |
|---|--|
| <ul style="list-style-type: none"><li>• criptare</li><li>• algoritm simetric/asimetric de criptare</li><li>• tehnica <i>padding</i></li><li>• tehnica <i>chaining</i> (înlănțuirii)</li></ul> | <ul style="list-style-type: none"><li>• pachetul DBMS_CCRYPTO</li><li>• TDE (<i>Transparent Data Encryption</i>)</li><li>• confidențialitate, integritate</li><li>• <i>hashing</i></li></ul> |
|---|--|

### 1. Introducere

- Criptarea datelor reprezintă un mod de a proteja datele curente și cele arhivate, de a le conferi confidențialitate.
- Exemple de date care necesită criptare? Parole, coduri PIN, numărul de securitate pe cardurile de credit etc.
- Elementele de bază într-un sistem de criptare sunt:
  - algoritmul de criptare – metoda prin care este modificată valoarea;
  - cheia de criptare, de a cărei siguranță depinde vulnerabilitatea datelor criptate.



Sursa: <http://msdn.microsoft.com/en-us/library/ff650720.aspx>

- Sistemul *Oracle* suportă **algoritmi de criptare**:
  - **simetrici** (care folosesc aceeași cheie pentru criptarea datelor și pentru decriptarea acestora) pentru criptarea datelor stocate;
  - **asimetrici** (în care receptorul generează 2 chei: o cheie privată ce va fi folosită de el pentru decriptare și o cheie publică pe care o trimite emitentului pentru a cripta mesajul) pentru autentificarea utilizatorilor bazei de date și pentru comunicarea între client și baza de date. Algoritmii de criptare asimetrici sunt parte a opțiunii contra-

cost Oracle Advanced Security.

Reținem că, pentru criptarea datelor stocate, sistemul Oracle utilizează algoritmi de criptare simetrici.

## 1.1 Recapitulare algoritmi simetrici de criptare

Reamintim, pe scurt, câțiva algoritmi simetrici de criptare, disponibili și în Oracle:

- **DES** (*Data Encryption Standard*)

Sistemul DES criptează un bloc de text clar de 64 biți într-un text criptat tot de 64 biți, utilizând 56 biți dintr-o cheie de 64 biți.

- **3-DES** (*Triple Data Encryption Standard*)

Are la bază formula  $c = \text{DES}_{k3}(\text{DES}_{k2}^{-1}(\text{DES}_{k1}(m)))$  în varianta 3DES-edc

sau formula  $c = \text{DES}_{k3}(\text{DES}_{k2}(\text{DES}_{k1}(m)))$  în varianta 3DES-eee

unde  $k1, k2, k3$  sunt chei de 56 biți (folosind astfel împreună 168 biți dintr-o cheie de 192 biți solicitată),  $\text{DES}_k$  este criptarea DES cu cheie  $k$ ,  $\text{DES}_k^{-1}$  este decriptarea DES cu cheie  $k$ , iar  $m$  este blocul de 64 de biți original.

Dacă  $k1 = k2$  sau  $k2 = k3$  sau  $k1 = k2 = k3$ , varianta 3DES devine DES.

- **AES** (*Advanced Encryption Standard*)

Sistemul AES criptează un bloc de text clar de 128 biți într-un text criptat tot de 128 biți, utilizând o cheie de 128, 192 sau 256 biți.

Se observă că algoritmi de criptare enumerați anterior lucrează cu blocuri de dimensiune fixă, stabilită (64 biți=8 bytes la DES și 3-DES, respectiv 128 biți=16 bytes la AES). Un fragment de date în clar va fi segmentat în blocuri de dimensiunea cerută de algoritm și algoritmul va fi aplicat pe fiecare bloc astfel obținut.



## 1.2 Padding și chaining

- Cum tratăm cazul în care dimensiunea datelor în clar NU este multiplu de dimensiunea cerută a blocului?

→ Se utilizează **tehnica padding** de completare a ultimului segment din fragmentul de date în clar până la dimensiunea unui bloc.

- Ne amintim că la funcțiile pe șiruri de caractere am întâlnit funcțiile LPAD, RPAD.

```
SELECT LPAD('A',3,'#'), RPAD('B',3,'@') FROM DUAL;
```

 LPAD('A',3,'#')	 RPAD('B',3,'@')
##A	B@@

- În vederea criptării, se poate opta pentru *padding* cu zero-uri sau pentru schema de *padding* PKCS#5.

- Fie  $dim\_bloc$  dimensiunea în bytes a blocului cerută de algoritm

$dim\_date$  dimensiunea totală în bytes a fragmentului de date în clar.

Schema de padding PKCS#5 calculează pentru ultimul segment din fragment diferența:

$$d = \text{dim\_bloc} - (\text{dim\_date} \bmod \text{dim\_bloc})$$

și completează fiecare octet lipsă cu valoarea hexa 0x0d.

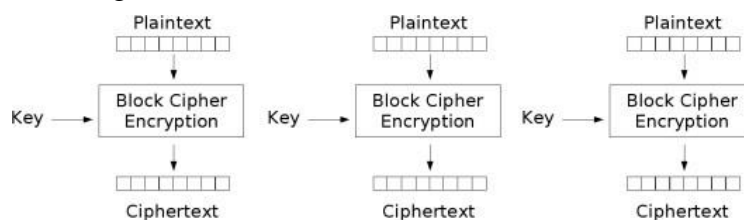
**Exemplu:**  $\text{dim\_bloc} = 8, \text{dim\_date} = 100 \Rightarrow d = 8 - (100 \bmod 8) = 8 - 4 = 4$

La ultimul segment, cel de 4 bytes, se face padding cu 0x04040404 (adică 00000100 00000100 00000100 00000100)

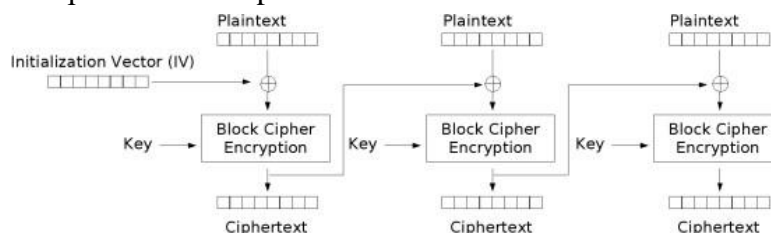
*Reținem că padding-ul se aplică înaintea criptării și este înlăturat după decriptare.*

- Cum tratăm cazul când fragmentul de date în clar constă din mai multe blocuri de criptat?  
 → Se utilizează **tehnica chaining (înlănțuirii)**, care stabilește dacă pentru un bloc criptarea este independentă sau dependentă de criptarea blocurilor anterioare din fragmentul în clar.
- În *Oracle* sunt disponibile variantele următoare de *chaining*:

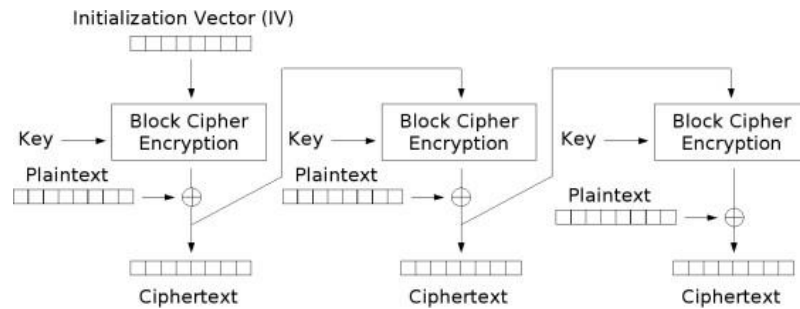
- *Electronic Code Book (CHAIN\_ECB)* – fiecare bloc este criptat independent de celelalte blocuri din fragment. Dezavantajul este că se pot identifica șabloane repetitive în fragment;



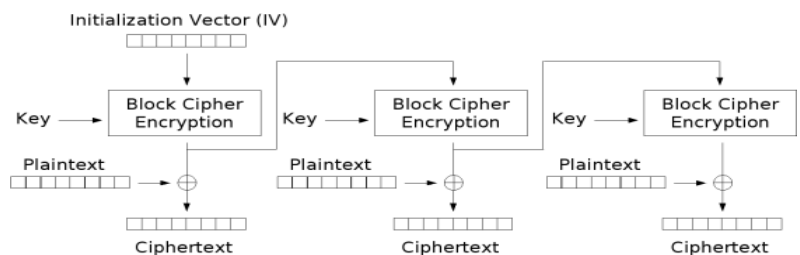
- *Cipher Block Chaining (CHAIN\_CBC)* – pentru fiecare bloc, înaintea criptării, este aplicat XOR cu un vector. Pentru primul bloc din secvență se folosește un vector de inițializare, iar pentru un bloc din restul secvenței se folosește ca vector de biți rezultatul criptării blocului precedent.



Sursa: [http://en.wikipedia.org/wiki/Block\\_cipher\\_modes\\_of\\_operation](http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation)

➤ *Cipher Feedback (CHAIN\_CFB) :*

Sursa: [http://en.wikipedia.org/wiki/Block\\_cipher\\_modes\\_of\\_operation](http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation)

➤ *Output Feedback (CHAIN\_OFB) :*

Sursa: [http://en.wikipedia.org/wiki/Block\\_cipher\\_modes\\_of\\_operation](http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation)

## 2. Criptarea datelor prin cod PL/SQL

- Avem la dispoziție pachetele:
  - DBMS\_CRYPTO (introdus odată cu versiunea Oracle 10g)
  - Anterior versiunii Oracle 10g: DBMS\_OBFUSCATION\_TOOLKIT (*deprecated*)

**Observație:** Executați următoarea comandă fiind logați ca *SYS AS SYSDBA* pentru a da utilizatorului *nume\_user* drept de execuție pe pachetul DBMS\_CRYPTO:

```
GRANT EXECUTE ON dbms_crypto TO nume_user;
```

În absența acestui drept, veți primi eroarea:

 Error(12,19): PLS-00201: identifier 'DBMS\_CRYPTO' must be declared

### 2.1 Sintaxa

- **CRIPTARE:**

```
dbms_crypto.encrypt(
  fragment_clar IN RAW,
  mod_operare IN PLS_INTEGER,
  cheie IN RAW,
  vector_inializare IN RAW DEFAULT NULL)
RETURN RAW;
```

- **DECRIPTARE:**

```
dbms_crypto.decrypt(
  fragment_clar IN RAW,
  mod_operare IN PLS_INTEGER,
  cheie IN RAW,
```

```
vector_inicializare IN RAW DEFAULT NULL)
RETURN RAW;
```

unde:

**Mod\_operare = Cod\_algoritm + Cod\_padding + Cod\_chaining**

DBMS_CRYPTO.ENCRYPT_DES	DBMS_CRYPTO.PAD_PCKSS	DBMS_CRYPTO.CHAIN_CBC
DBMS_CRYPTO.ENCRYPT_3DES_2KEY	DBMS_CRYPTO.PAD_ZERO	DBMS_CRYPTO.CHAIN_CFB
DBMS_CRYPTO.ENCRYPT_3DES	DBMS_CRYPTO.PAD_NONE	DBMS_CRYPTO.CHAIN_ECB
DBMS_CRYPTO.ENCRYPT_AES128		DBMS_CRYPTO.CHAIN_OFB
DBMS_CRYPTO.ENCRYPT_AES192		
DBMS_CRYPTO.ENCRYPT_AES256		
DBMS_CRYPTO.ENCRYPT_RC4		

## 2.2 Alte funcții utile

- Conversie VARCHAR2 → RAW:

```
utl_i18n.string_to_raw(
    data IN VARCHAR2 CHARACTER SET ANY_CS,
    dst_charset IN VARCHAR2 DEFAULT NULL)
RETURN RAW;
unde dst_charset = 'AL32UTF8'
```

Alternativ se poate utiliza, dacă în baza de date este setat *character set*-ul la *AL32UTF8*:

```
utl_raw.cast_to_raw(sirul IN VARCHAR2) RETURN RAW;
```

- Conversie RAW → VARCHAR2 cu caractere:

```
utl_i18n.raw_to_char(
    data IN RAW,
    src_charset IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
unde dst_charset = 'AL32UTF8'
```

- Conversie RAW ↔ VARCHAR2 cu hexa:

```
RAWTOHEX ( data IN RAW) RETURN VARCHAR2;
HEXTORAW (data IN VARCHAR2) RETURN RAW;
```

## 3. Aspecte privind managementul cheilor de criptare a datelor

- Este dificil pentru utilizatorii bazei de date să genereze manual chei eficiente de criptare, de lungimea solicitată de algoritmii de criptare.
- În ceea ce privește furnizarea manuală a cheii de criptare sub forma unui șir de caractere (convertit apoi în RAW), lungimea șirului se calculează astfel:

$$L_{sir} = Lungime\_cheie\_in\_biti / 8$$

**Exemplu:** Pentru ENCRYPT\_AES128, cheia este de 128 biți => șirul va avea lungimea  

$$L_{sir} = 128/8 = 16$$

Furnizarea cheii '1234567890123456' va fi acceptată întrucât are 16 caractere, în timp ce cheia '1234' va ridica excepția “*key length too short*” Analog, pentru restul algoritmilor, pe baza tabelului:

Constant	Effective key length
ENCRYPT_DES	56
ENCRYPT_3DES	156
ENCRYPT_AES128	128
ENCRYPT_AES192	192
ENCRYPT_AES256	256

- Alternativa o reprezintă **generarea automată a cheilor** de dimensiunea dorită:  
`cheie RAW (nr_bytes);`  
`cheie:= DBMS_CRYPTO.randombytes (nr_bytes);`
- Funcția *randombytes* implementează algoritmul *Pseudo-Random Number Generator*.
- Odată obținute, cheile secrete trebuie păstrate în siguranță, întrucât divulgarea lor poate compromite securitatea datelor criptate.
- Opțiuni:
  - | ---- o cheie la nivelul bazei de date |--- stocată în baza de date (într-un tabel special)
  - | |--- stocată într-un fișier extern bazei de date
  - |
  - | ---- o cheie la nivel de înregistrare --- stocată în baza de date (într-un tabel special)
  - |
  - |-----o combinație între cele anterioare – există o cheie master la nivelul bazei de date și câte o cheie la nivel de înregistrare. Atât la criptare cât și la decriptare se folosește o *cheie hibridă* = *cheia master XOR cheie înregistrare* (funcția PL/SQL *UTL\_RAW.bit\_xor*)

Reținem că opțiunea unei chei hibride este cea mai eficientă dintre opțiunile enumerate;

- dacă se fură baza de date cu totul, datele nu vor putea fi decriptate în cazul stocării cheii master în sistemul de fișiere;
- dacă se divulgă cheia master și o cheie de înregistrare, celelalte înregistrări rămân protejate.

#### 4. Transparent Data Encryption (TDE)

- Este o facilitate oferită începând cu versiunea *Oracle 10g*, care permite declararea unor coloane criptate la nivelul unui tabel al bazei de date.
- La inserarea datelor în coloanele declarate criptate, în mod automat *Oracle* criptează datele și le stochează criptate în baza de date. Orice operație SELECT va decripta automat datele din bază. Reținem ca **Transparent Data Encryption** nu face diferențiere între utilizatori, oferindu-le tuturor valoarea decriptată a datelor.

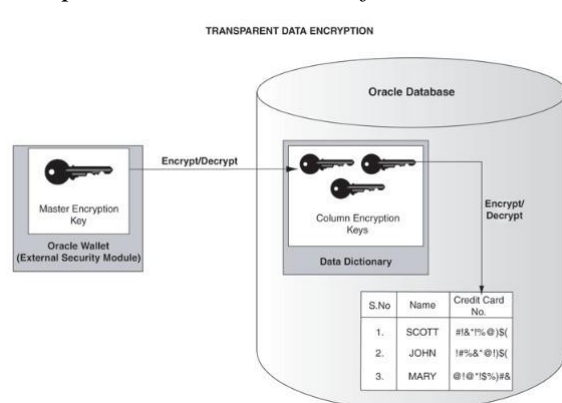
- Nu orice coloană poate fi declarată 'criptată'; coloanele din cheia externă (*foreign key*) NU pot fi criptate TDE.

**Exemplu:** Fie tabelul *CONT* (*id\_cont#*, *serie\_card*, *posesor*, *sold*) pentru care dorim să declarăm criptate coloanele *serie\_card* și *sold*:

```
ALTER TABLE cont MODIFY (serie_card ENCRYPT USING 'AES128');
```

```
ALTER TABLE cont MODIFY (sold ENCRYPT USING 'AES128');
```

- Pentru toate coloanele criptate dintr-un tabel *T* se folosește o aceeași cheie privată *Key\_T*. Dacă avem mai multe tabele *T1*, *T2*, ..., *Tn* care conțin fiecare diverse coloane criptate, rezultă *n* chei private *Key\_T1*, *Key\_T2*, ..., *Key\_Tn*.
- Fiecare cheie privată *Key\_Tj*, *j* = 1, ..., *n*, este criptată la rândul ei cu o cheie *master* *Key\_Master* și rezultatul criptării ei este stocat în dicționarul datelor.
- Cheia master este stocată extern bazei de date într-un *wallet*. Astfel, *Transparent Data Encryption* previne decriptarea datelor în cazul furtului bazei de date.



Sursa: [http://docs.oracle.com/cd/B28359\\_01/network.111/b28530/asotrans.htm](http://docs.oracle.com/cd/B28359_01/network.111/b28530/asotrans.htm)

- Pașii:

<b><i>La criptare automată</i></b>	<b><i>La decriptare automată</i></b>
Obținerea cheii master <i>Key_Master</i> din <i>wallet</i> -ul extern;	Obținerea cheii master <i>Key_Master</i> din <i>wallet</i> -ul extern;
Decriptarea cheii private <i>Key_Tk</i> folosind cheia <i>master</i> ;	Decriptarea cheii private <i>Key_Tk</i> folosind cheia <i>master</i> ;
Criptarea datelor de inserat folosind cheia privată <i>Key_Tk</i> ;	Decriptarea datelor folosind cheia privată <i>Key_Tk</i> ;
Stocarea datelor criptate în coloanele tabelului.	Returnarea rezultatului.

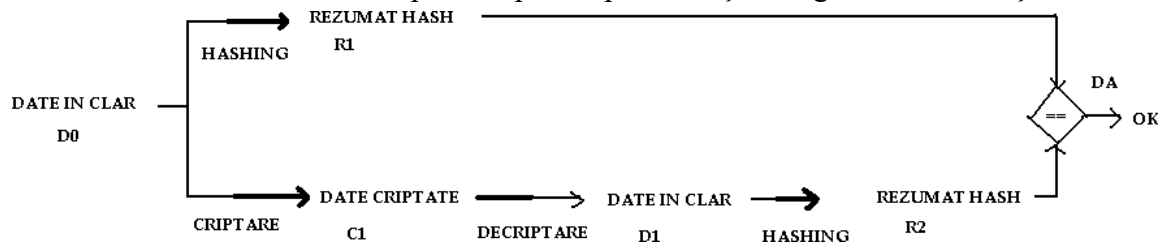
Mai multe detalii – în documentația *Oracle*.

## 5. Asigurarea integrității datelor criptate

*Criptarea datelor le asigura confidențialitatea, dar nu le garantează și integritatea. Astfel, datele criptate pot fi modificate.*

- Pentru a preveni acest pericol, în afară de criptarea datelor originale, se utilizează tehnica de *hashing*, de rezumare a datelor originale. *Hashing*-ul are două proprietăți importante:

- nu permite descifrarea valorii originale;
- este determinist, adică aplicat repetitiv pe aceleași date generează același rezultat.



- Oracle permite algoritmi de *hashing*: MD5 și SHA-1.
- Sintaxa:  

```

DBMS_CRYPTO.Hash (
    sir_original IN RAW,
    mod_operare IN PLS_INTEGER)
RETURN RAW;

```

unde  $mod\_operare \in \{DBMS\_CRYPTO.HASH\_MD5, DBMS\_CRYPTO.HASH\_SH1\}$



## 6. Exerciții

1. Scrieți procedura *CRIPTARE1* care criptează un șir primit ca parametru folosind algoritmul DES, cheia '12345678', *padding* cu zero-uri și metoda de chaining ECB. Apelați procedura pentru șirul de caractere 'Text în clar' dintr-un bloc PL/SQL anonim.

2. Scrieți procedura *DECRYPTARE1*, care decriptează un șir primit ca parametru folosind algoritmul DES, cheia '12345678', *padding* cu zero-uri și metoda de chaining ECB. Apelați procedura în același bloc PL/SQL anonim de la exercițiul 1.

3. Datele de salarizare (*id\_employee* și *salary*) ale tabelului *EMPLOYEES* vor fi criptate (AES- 128, *PAD\_PKCS5*, *CHAIN\_CBC*) și stocate în tabelul *EMPLOYEES\_CRIPT* astfel:

- înregistrările impare(1, 3,...) vor fi criptate cu cheia *CHEIE\_IMPAR* și
- înregistrările pare(2, 4,...) vor fi criptate cu cheia *CHEIE\_PAR*.

Cele doua chei vor fi generate automat și stocate în baza de date.

Să se creeze secvența *SECV\_IDCHEIE* și tabelul *TABEL\_CHEI* (*idcheie#*, *cheie*, *tabel*).

Să se creeze procedura *CRIPTARE\_PAR\_IMPAR* fără parametri. În cadrul procedurii să se genereze în mod automat 2 chei private *CHEIE\_IMPAR* și *CHEIE\_PAR* pe câte 16 bytes fiecare. Cheile se vor stoca în tabelul *TABEL\_CHEI*, cu cheie primară din secvența *SECV\_IDCHEIE*.

4. Încercați să modificați (*UPDATE*) valoarea criptată a salariului primului angajat (ca număr de ordine) din tabelul *EMPLOYEES\_CRIPT*. Setati-i salariul la valoarea 0x1F4 (adică 500 în decimal). Actualizarea a reușit?

5. Să se creeze procedura *DECRYPTARE\_PAR\_IMPAR* fără parametri, perechea celei de la exercițiul 4. În decriptare se folosesc cheile salvate în *TABEL\_CHEI*, în aceeași ordine (impar, par).

Datele decriptate se stochează în tabelul *EMPLOYEES\_DECRYPT*.

Comparați salariile primului angajat din tabelele *EMPLOYEES* și *EMPLOYEES\_DECRYPT*.

6. Creați o funcție *REZUM\_MD5* ce returnează rezumatul *hash* (MD5) pentru înregistrarea din tabelul *EMPLOYEES* corespunzătoare angajatului cu *employee\_id* egal cu 104. Stocați rezultatul într-o variabilă *bind rezumat1*. Actualizați salariul acestui angajat acordându-i un spor de 20%.

Creați un nou rezumat *hash* al acestei înregistrări și stocați rezultatul în variabila *bind rezumat2*. Ce observați?