

The logo features the letters 'AMIS' in a bold, red, sans-serif font. A thick, red, curved swoosh starts from the bottom left of the 'A' and sweeps upwards and to the right, passing behind the 'M' and 'I' and ending above the 'S'.

**AMIS**

**COMMITTED TO ICT.  
INVOLVED IN PEOPLE.**

# Wie ik ben

- Andre van Winssen
- >20 jaar aan het werk met oracle software
- Oracle database support, DBA, security, OIAM, OU
- Principal Consultant Amis, BC
- OCP from 7.3-11g, OCM 10g, 11g
- CISA (auditing), CISSP (security), CEH (ethical hacking)



# TRAININGEN EN MASTERCLASSES



## 28 oktober 2011

**Oracle Database 10g/11g Advanced Security door Andre van Winssen**

**Data veilig? Het kan beter....**

**Leer over de complete set aan security opties in Oracle databases**

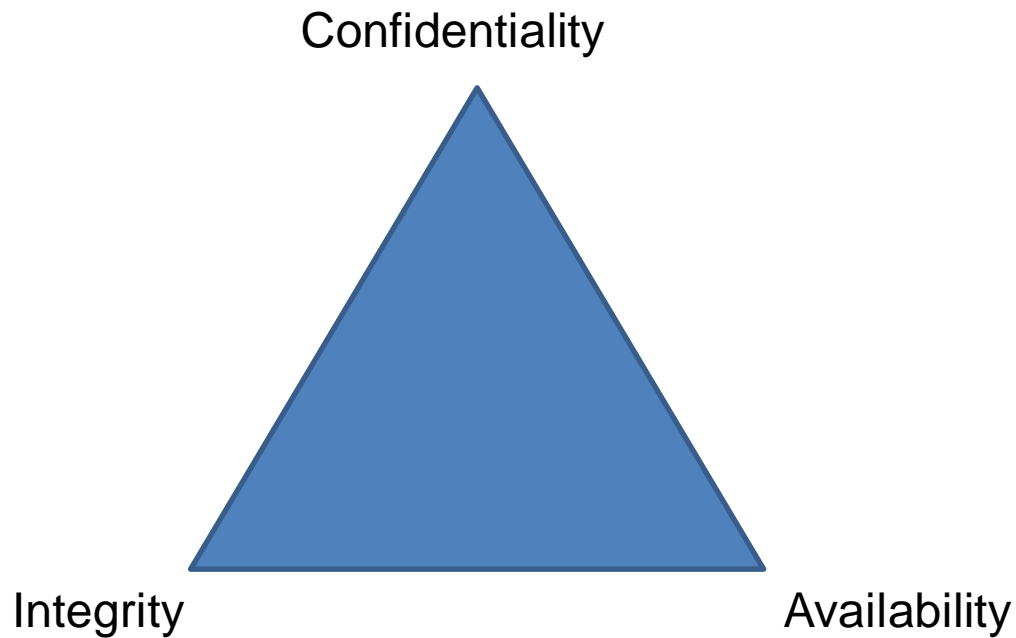
Advanced Security	Oracle Database Firewall	Applicatie Encryptie
Database Vault	Secure Application Roles	File encryptie (rman, DP)
Audit Vault	Oracle Label Security	Authenticatie en autorisatie
Data Masking	Virtual private databases	Auditing + FGA

Schrijf je in via [www.amis.nl](http://www.amis.nl)

Applicatie Encryptie  
of  
Toegepaste versleuteling

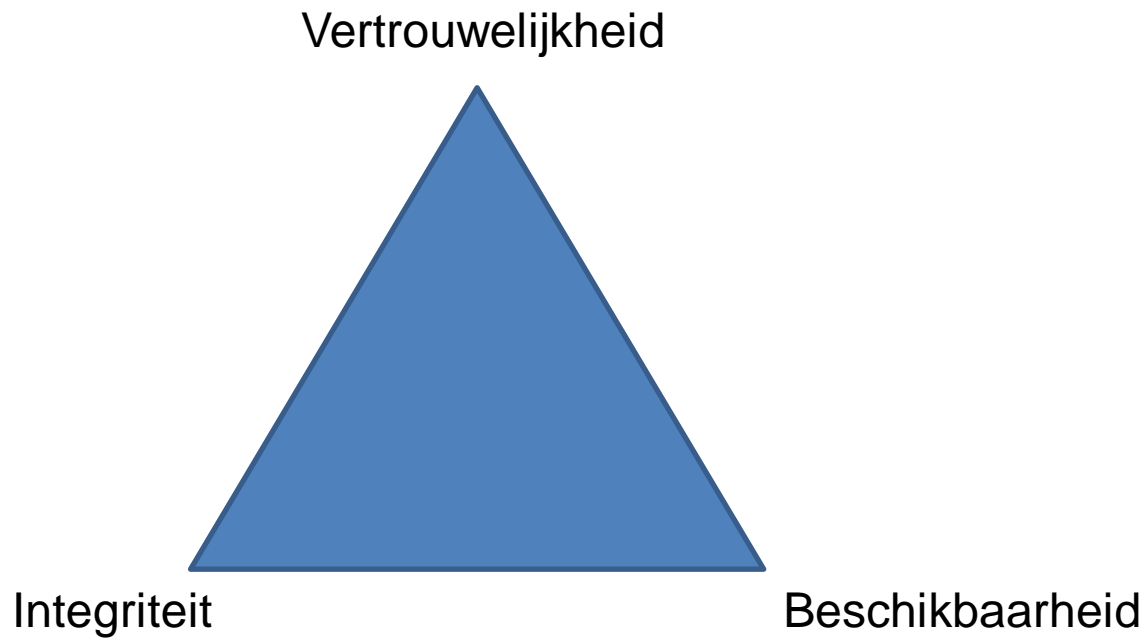
**OGH 4 oktober 2011**

## CIA Triad



[http://en.wikipedia.org/wiki/Information\\_security](http://en.wikipedia.org/wiki/Information_security)

# VIB Driehoek

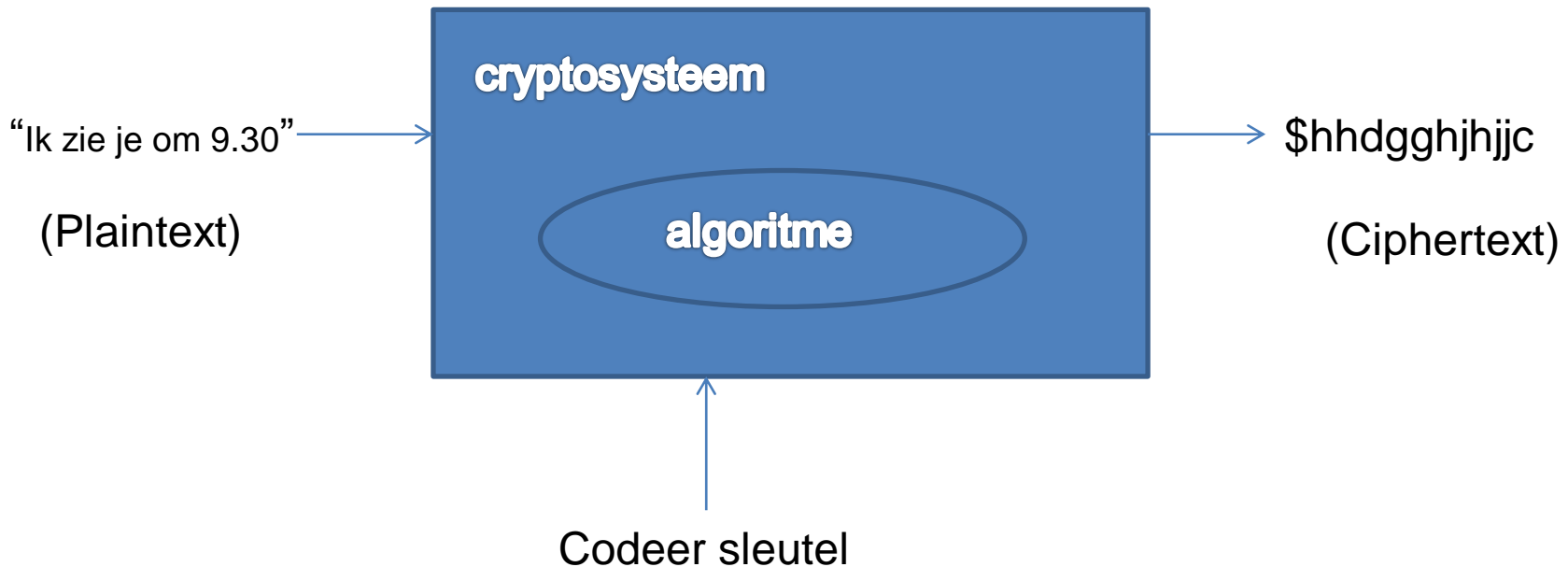


# Cryptografie

- Letterlijk: “verborgen schrijven”
- Plaintext alleen bedoeld voor zender en ontvanger
- Plaintext -> ciphertext -> plaintext
- In Oracle context
  - Codeer gegevens in ruste (disk)
  - Codeer gegevens in beweging (netwerk verkeer)
  - Codeer gegevens in werkgeheugen
- Cryptografie helpt vertrouwelijkheid te handhaven

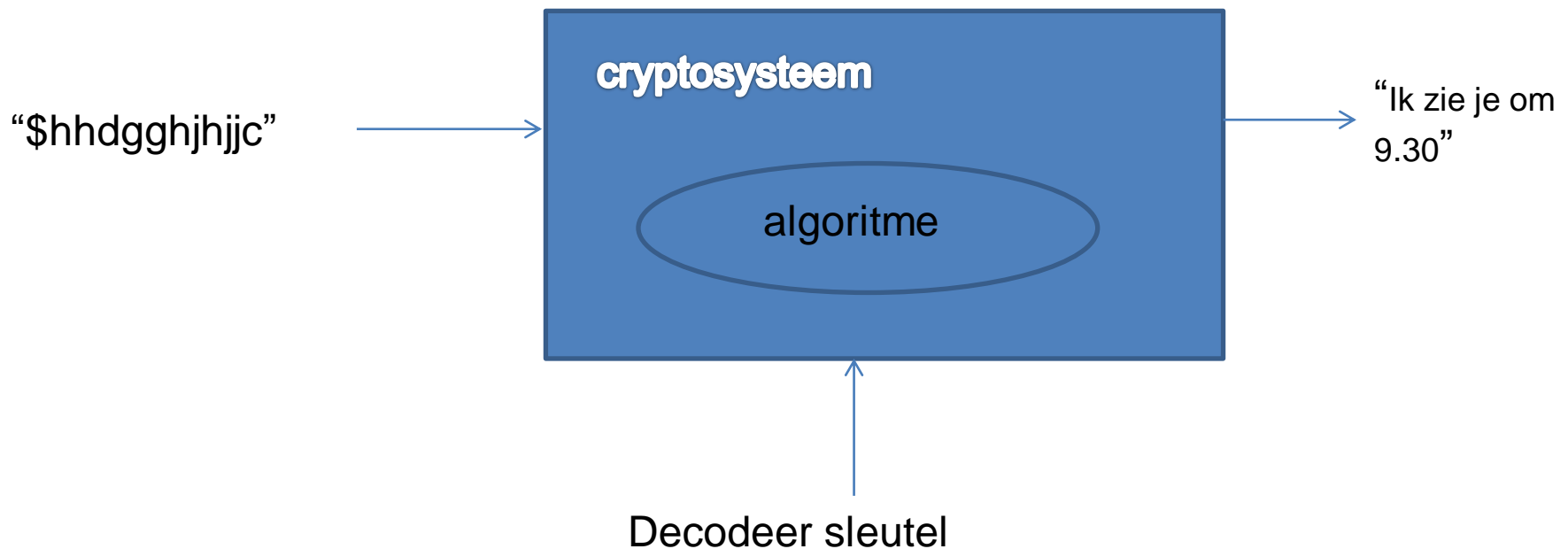
# Encryptie

- ENCRYPTIE: coderen (versleutelen) van gegevens op basis van een bepaald algoritme en een sleutel.



# Decryptie

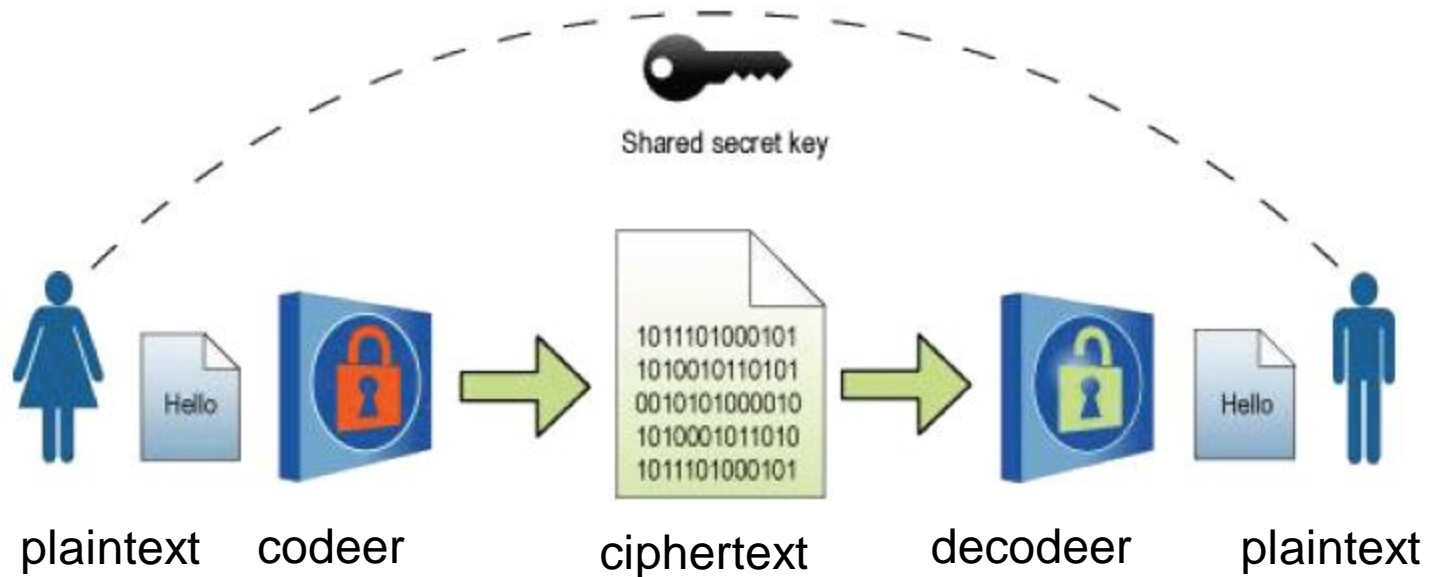
- DECRYPTIE: decoderen van versleutelde gegevens op basis van algoritme en decodeer sleutel



# Encryptie types

- Symmetrisch
  - Codeer sleutel en decodeer sleutel identiek
  - Noodzaak om sleutels **veilig** uit te wisselen
  - Snelle codering/decodering mogelijk
- Asymmetrisch
  - Codeer sleutel verschilt van decodeer sleutel
  - Langzamere codering/decodering
  - Niet geschikt voor grote hoeveelheden te versleutelen gegevens

# Symmetrische Encryptie



# Symmetrische Encryptie

- Stream cipher
  - XOR operatie op elke bit van originele text
    - met behulp van bijv een 128bits key
  - Gecodeerde tekst even lang als originele
  - XOR :  $1+0=1$  ,  $0+1=1$  ,  $1+1=0$  ,  $0+0=0$
  - Ondersteunde Stream Cipher Algorithms
    - ENCRYPT\_RC4 CONSTANT PLS\_INTEGER := 129; -- 0x0081

# Symmetrische Encryptie

- Block cipher
  - Originele tekst opgeknipt in blokken
  - XOR'd met cipher blokken
  - XOR :  $1+0=1$  ,  $0+1=1$  ,  $1+1=0$  ,  $0+0=0$

# Symmetrische Encryptie

## Verschillende Block Cipher algoritmen

- DES Data Encryption Standard, Blockcipher
- 3DES blok 3 x gecodeerd
- 3DES\_2KEY blok 3 x gecodeerd met 2 sleutels
- AES Advanced Encryption Standard
- PBE\_MD5DES
- AES128 Advanced Encryption Standard 128bit key
- AES192 Advanced Encryption Standard 192bit key
- AES256 Advanced Encryption Standard 256bit key

# Asymmetrische Encryptie

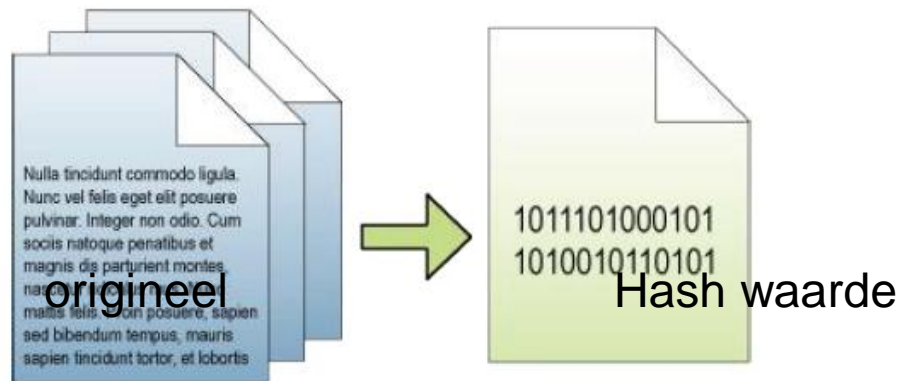
- RC4 stream cipher, gebruikt geheime sleutel die uniek is voor iedere sessie

# Symmetrische Encryptie

- **Ondersteunde Block Cipher Chaining smaken**
  - ECB Electronic Code Book – elk blok onafhankelijk van andere gecodeerd
  - CBC Cipher Block Chaining – elk blok gecodeerd met ciphertext van vorige blok
  - CFB Cipher Feedback Block – codering van blokken data kleiner dan blok grootte
  - OFB Output Feedback –  $n$  bits van vorige output blok gebruikt in te coderen blok
- **Ondersteunde Block Cipher Padding smaken**
  - PKCS5 aanvullen volgens PKCS #5 standaard (password based cryptography)
  - NONE niet aanvullen, programma moet zorgen dat blok grootte correct is
  - ZERO aanvullen met 0'en

# Hashing

- Wiskundige bewerking om willekeurige string van bits om te zetten in unieke string van bits van vaste lengte
- Eénrichting alleen, originele waarde kan niet worden teruggerekend
- Kandidaatwaardes kunnen wel gevoerd worden aan algoritme om te kijken of zelfde hash waarde wordt geproduceerd
  - Zo ja dan heb je een hit!



# Hashing algoritmen

- MD4
  - Produceert 128 bits hash waarde (16 bytes)
- MD5
  - Idem als MD4 maar sterker algoritme (RFC1321, Ron Rivest)
- SHA-1, National Security Agency (1995)
  - Produceert 160 bit hash waarde (20 bytes)
- SHA-2, National Security Agency (2001)
- SHA-3, nog in ontwikkeling

# Message Authentication Code MAC

- Zelfde idee als hashing maar met behulp van een codeer sleutel, dus wel heen maar niet terug
- Ondersteunde functies
  - HMAC\_MD5 als MD5 maar met codeer sleutel
  - HMAC\_SHA1 als SHA-1 maar met codeer sleutel

# DBMS\_CRYPTO

- Meegeleverd PLSQL package
- Vervangt DBMS\_OBFUSCATION\_TOOLKIT
- Belangrijkste sup programma's:
  - Encrypt
  - Decrypt
  - Hash
  - MAC
  - Randombytes om sleutels mee te genereren.
- Executie privilege standaard meegegeven aan:
  - APEX\_040000, OWBSYS, SYSMAN

# DBMS\_CRYPTO

## voorbeeld van encryptie

FUNCTION ENCRYPT RETURNS RAW

Argument Name	Type	In/Out	Default?
SRC	RAW	IN	
TYP	BINARY_INTEGER	IN	
KEY	RAW	IN	
IV	RAW	IN	DEFAULT

```
FUNCTION encrypt(origineel IN VARCHAR2 , sleutel in varchar2)
RETURN VARCHAR2 AS
    orig_raw RAW(4000);
BEGIN

    orig_raw := SYS.DBMS_CRYPTO.ENCRYPT(
        SRC=>SYS.UTL_I18N.STRING_TO_RAW(origineel, 'AL32UTF8'),
-- DES_CBC_PKCS5 = ENCRYPT_DES + CHAIN_CBC+ PAD_PKCS5
        TYP=> SYS.DBMS_CRYPTO.DES_CBC_PKCS5,
        KEY=> utl_raw.cast_to_raw(sleutel));

    RETURN RAWTOHEX(orig_raw);
END;
/
```

# DBMS\_CRYPTO

## voorbeeld van decryptie

```
FUNCTION DECRYPT (RETURNS RAW
```

Argument Name	Type	In/Out	Default?
SRC	RAW	IN	
TYP	BINARY_INTEGER	IN	
KEY	RAW	IN	
IV	RAW	IN	DEFAULT

```
FUNCTION decrypt(codeer_text IN VARCHAR2 , sleutel in varchar2)
RETURN VARCHAR2 AS
    RAWSTRING RAW(4000);
BEGIN
    RAWSTRING := SYS.DBMS_CRYPTO.DECRYPT(
        SRC=>HEXTORAW(encrvalue),
        TYP=> SYS.DBMS_CRYPTO.DES_CBC_PKCS5,
        KEY=> utl_raw.cast_to_raw(sleutel));

    RETURN SYS.UTL_I18N.RAW_TO_CHAR(RAWSTRING, 'AL32UTF8');
END;
/
```

# DBMS\_CRYPTO

## voorbeeld van hashing

```
FUNCTION HASH RETURNS RAW
```

Argument Name	Type	In/Out	Default?
SRC	RAW	IN	
TYP	BINARY_INTEGER	IN	

```
FUNCTION HASH(p_text IN VARCHAR2 , HASHTYPE in binary_integer)
RETURN VARCHAR2 AS
    RAWSTRING RAW(4000);
BEGIN
    RAWSTRING := SYS.DBMS_CRYPTO.HASH(
        SRC=>utl_raw.cast_to_raw(p_text),
        TYP=> hashtype);
    RETURN SYS.UTL_I18N.RAW_TO_CHAR(RAWSTRING, 'AL32UTF8');
END;
/
```

```
-- in DBMS_CRYPTO gedefinieerde constanten voor Hash functions
HASH_MD4 CONSTANT PLS_INTEGER := 1;
HASH_MD5 CONSTANT PLS_INTEGER := 2;
HASH_SH1 CONSTANT PLS_INTEGER := 3;
```

# DBMS\_CRYPTO

## voorbeeld van MAC

```
FUNCTION MAC RETURNS RAW
```

Argument Name	Type	In/Out	Default?
SRC	RAW	IN	
TYP	BINARY_INTEGER	IN	
KEY	RAW	IN	

```
FUNCTION MAC (p_text IN VARCHAR2 , HASHTYPE in binary_integer , SLEUTEL IN  
VARCHAR2)
```

```
RETURN VARCHAR2 AS
```

```
    RAWSTRING RAW(4000);
```

```
BEGIN
```

```
    RAWSTRING := SYS.DBMS_CRYPTO.MAC (  
        SRC=>utl_raw.cast_to_raw(p_text),  
        TYP=> SYS.DBMS_CRYPTO.HMAC_MD5,  
        KEY=> utl_raw.cast_to_raw(sleutel));
```

```
    RETURN SYS.UTL_I18N.RAW_TO_CHAR(RAWSTRING, 'AL32UTF8');
```

```
END;
```

```
-- in DBMS_CRYPTO gedefinieerde constanten voor MAC Functions
```

```
HMAC_MD5 CONSTANT PLS_INTEGER := 1;
```

```
HMAC_SH1 CONSTANT PLS_INTEGER := 2;
```

# DBMS\_CRYPTO

## voorbeeld van sleutel generatie

```
FUNCTION RANDOMBYTES RETURNS RAW
```

Argument Name	Type	In/Out	Default?
NUMBER_BYTES	BINARY_INTEGER	IN	

```
FUNCTION RANDOMINTEGER RETURNS BINARY_INTEGER
```

```
FUNCTION RANDOMNUMBER RETURNS NUMBER
```

```
raw_key := dbms_crypto.randombytes (number_bytes => 24);
```

NUMBER\_BYTES: aantal te genereren pseudo-willekeurige bytes

- DBMS\_CRYPTO.RANDOMBYTES volgt de RSA Security Inc. x9.31 Pseudo-Random Number Generator (PRNG) standaard en is goedgekeurd door de Federal Information Processing Standard (FIPS)-140-2 Annex C.
- DBMS\_OBFUSCATION\_TOOLKIT.getkey voldoet ook hieraan
- DBMS\_RANDOM niet!

# Sleutelbeheer

- In de database
- In een OS bestand
- Door de gebruiker beheerd

# Sleutelbeheer in de database

## TIPS:

- Bewaar sleutels in ander schema dan te coderen data
- Gebruik “wrap” voor versleutelen PLSQL package
  - wrap iname=mysecretcode.sql
- Gebruik verschillende sleutels per rij
  - PK van te versleutelen tabel is FK naar key tabel (1:1)
- Pas extra transformaties toe
  - Bijv: Gebruik PK waarde als onderdeel van sleutel

# Sleutelbeheer in OS bestand

- Gebruik UTL\_FILE om key op te halen
- Doe codeer en decodeer actie helemaal buiten database om

# Sleutelbeheer

## door gebruiker beheerd

Probleem gebieden:

- Gebruiker vergeet de sleutel
- Gebruikt niet veilige manier van opslag
  - Post-its remain undetected by intrusion prevention systems
- Ingevoerde sleutel gaat niet gecodeerd over het netwerk

# Alternatieven voor zelf toegepaste versleuteling

- Oracle Database Vault
  - Beschermt tegen alziend oog van de DBA
- Transparent Data Encryption
  - Gebruik van wallet of HSM (Hardware Security Module)
  - Masterkey (de-)codeert de sleutels voor data codering
    - Maakt hercoderen met andere sleutel mogelijk zonder alle data te hoeven decoderen + coderen
- Bestands encryptie van backups
  - RMAN backup encryption
  - Oracle Secure backup
  - Encrypted filesystem (EFS)

# Q&A

COMMITTED TO ICT.  
INVOLVED IN PEOPLE.