

ATTENZIONE: Anche se ho impiegato cura ed impegno nella realizzazione di questo documento, consiglio vivamente chiunque voglia applicare i suggerimenti da me proposti, di eseguire tutte le opportune prove in un ambiente di test, prima di effettuare eventuali modifiche in produzione. Questa convinzione nasce dal fatto che nonostante mi sia documentato attraverso libri di testo, manuali e documenti in rete, esiste sempre la probabilità di eventuali errori nella stesura ed imprecisioni nella esposizione.

ATTENZIONE: Quanto scritto in questo documento può essere utilizzato in modo del tutto libero. Pregherei comunque tutti coloro che desiderano delle modificarlo, di comunicarlo al mio indirizzo di posta in modo tale da permettermi di aggiornare e/o correggere eventuali errori.

<http://www.oral.com> Aprile 2005

Di [Andrea Salzano](#)

Encrypt and Decrypt

Prima o poi nasce l'esigenza di dover criptare i dati all'interno della propria base dati. Oracle mette a disposizione un package PL/SQL utile a tale scopo: DBMS_OBFUSCATION_TOOLKIT. Questo permette ad un'applicazione di criptare, in Oracle 8i, i dati utilizzando l'algoritmo di codifica DES (Data Encryption Standard) attraverso la procedura DESEncrypt()/DESDecrypt e, a partire da Oracle 9i¹, l'algoritmo triplo DES con la procedura DES3Encrypt()/DES3Decrypt. In Oracle 10g esiste un nuovo package, DBMS_CRYPTO, estende e sostituisce DBMS_OBFUSCATION_TOOLKIT, anche se quest'ultimo comunque continua ad essere presente.

L'algoritmo DES è una chiave simmetrica, cioè la stessa chiave è utilizzata sia per codificare che per decodificare. DES codifica i dati a blocchi di 64-bit usando una chiave di 56-bit (l'algoritmo ignora i primi 8 bit). I dati codificati dal triplo DES (3DES) sono più difficili da violare usando un metodo di ricerca esaustiva: 2¹¹² o 2¹⁶⁸ tentativi piuttosto che 2⁵⁶ tentativi: il triplo DES non è vulnerabile quanto DES.

E' inoltre consigliato utilizzare l'utility "wrap" (installato da Oracle nella \$ORACLE_HOME/bin), per nascondere il codice PL/SQL al cui interno c'è il codice per la codifica dei dati. Questo previene la possibilità che gli utenti possano leggere la chiave utilizzata per criptare e decriptare i dati.

¹ Nel manuale di Oracle 8.1.7 non viene fatto riferimento alla funzione DES3Encrypt ed alla funzione DES3Decrypt. Anche se sono riuscito a crearle con successo su un Oracle 8.1.7.4, quando provo ad utilizzarle, ottengo il seguente errore:

```
SQL> insert into login values ('andrea',encrypt('andreapwd'));
insert into login values ('andrea',encrypt('andreapwd'))
*
```

```
ERROR at line 1:
ORA-28235: algorithm not available
ORA-06512: at "SYS.DBMS_OBFUSCATION_TOOLKIT_FFI", line 0
ORA-06512: at "SYS.DBMS_OBFUSCATION_TOOLKIT", line 123
ORA-06512: at "ANDREA.ENCRYPT", line 10
ORA-06512: at line 1
```

1. Cripting

La procedura di cripting DESEncrypt() ha tre argomenti:

```
PROCEDURE DESDECRYPT
```

Argument Name	Type	In/Out	Default?
INPUT_STRING	VARCHAR2	IN	
KEY_STRING	VARCHAR2	IN	
DECRYPTED_STRING	VARCHAR2	OUT	

Come si può vedere, ci sono 2 parametri di input: la stringa da codificare e la chiave da utilizzare per la sua stessa codifica, ed un solo parametro di output dove viene memorizzata la stringa criptata. Questa procedura è overloaded con una equivalente i cui parametri sono di tipo RAW, per criptare dati dello stesso tipo.

La procedura DES3Encrypt(), ha invece 4 argomenti:

```
PROCEDURE DES3ENCRYPT
```

Argument Name	Type	In/Out	Default?
INPUT_STRING	VARCHAR2	IN	
KEY_STRING	VARCHAR2	IN	
ENCRYPTED_STRING	VARCHAR2	OUT	
WHICH	BINARY_INTEGER	IN	DEFAULT

Rispetto a DESEncrypt(), questa procedura ha un argomento in più: "which". Se impostato a "0" (zero), il valore di default, allora viene utilizzata una chiave di cripting a 128 bits (TwoKeyMode o 2-key); se invece è impostato ad 1, allora viene utilizzata una chiave di cripting a 192 bits (ThreeKeyMode o 3-key).

In figura 1 è riportato un esempio di funzione di cripting. Commentiamo il codice. a) E' importante dire che la stringa da criptare sia un multiplo di 8 bytes. In caso contrario, viene restituito il seguente errore:

```
SQL> insert into login values ('andrea',encrypt('anpwd'));
insert into login values ('andrea',encrypt('anpwd'))
*
ERROR at line 1:
ORA-28232: invalid input length for obfuscation toolkit
ORA-06512: at "SYS.DBMS_OBFUSCATION_TOOLKIT_FFI", line 0
ORA-06512: at "SYS.DBMS_OBFUSCATION_TOOLKIT", line 33
ORA-06512: at "ANDREA.ENCRYPT", line 10
ORA-06512: at line 1
```

```

create or replace function encrypt (
    input_string IN varchar2
)
return varchar2
is
    data_crypt varchar2(256);
    data_crypt_out varchar2(256);

begin
    data_crypt := rpad( input_string,
        (trunc(length(input_string)/8)+1)*8, chr(0) );

    dbms_obfuscation_toolkit.DESEncrypt(
        input_string => data_crypt,
        key_string    => 'AnDrEaAn',
        encrypted_string=> data_crypt_out);

    return data_crypt_out;
end;

```

- Figura 1 - Funzione di codifica -

Occorre quindi essere certi che la stringa da criptare sia un multiplo di 8 bytes, che è uno dei vincoli imposti. b) La funzione LENGTH restituisce la lunghezza in caratteri della stringa passata. Dividere tale valore per 8 vuol dire fornire in media quanti bits ci sono in un singolo carattere della stringa stessa. Dato un numero decimale nella forma n,m (dove n rappresenta la parte intera ed m i decimali), la funzione TRUNC, applicata ad un tipo NUMBER, restituisce la sola parte intera, che nel nostro caso è "n". In particolare, nel caso in cui la cifra intera è 0 (zero), allora TRUNC (0.m) che in realtà viene visualizzato come TRUNC (.m) restituisce 0 (zero). Facendo in questo modo, ovvero

$$(\text{TRUNC} (\text{LENGTH} (\text{input_string}) / 8) + 1) * 8$$

mi assicuro di ottenere un valore multiplo di 8, anche per quelle stringhe la cui lunghezza è minore di 8 bytes. c) La funzione RPAD ha lo scopo è quello di aggiungere n caratteri a destra (R sta per right) della stringa passata. La sua sintassi è:

RPAD (char1 , n [, char2])

Abbiamo detto che uno dei vincoli della procedura DESEncrypt() di cripting è quella che la lunghezza della stringa da criptare sia un multiplo di 8 bytes e questo lo si realizza proprio con la funzione RPAD. Infatti, nel nostro caso, si aggiungono spazi bianchi (CHR(0) rappresenta in ASCII proprio lo spazio bianco) alla stringa "input_string" fino ad una lunghezza pari a:

$$(\text{TRUNC}(\text{LENGTH}(\text{input_string}) / 8) + 1) * 8$$

Nel nostro caso:

```
char1 -> input_string
n      -> TRUNC ( LENGTH ( input_string ) / 8 ) + 1 ) * 8
char2 -> chr(0)
```

Per capire facciamo un esempio, supponendo che input_string sia il carattere "a":

INPUT	OUTPUT
a	
length ('a')	1
length ('a')/8	.125
Trunc(length('a')/8)	0
Trunc(length('a')/8)+1	1
(trunc(length('a')/8)+1)*8	8
rpad('a', (trunc(length('a')/8)+1)*8, chr(0))	a[][][][][][][]*

* il simbolo [] rappresenta uno spazio

Come si può quindi vedere, con questo algoritmo (che ho preso da uno dei forum sul sito di Tom Kyte (<http://asktom.oracle.com>), si riesce ad accettare in ingresso, una stringa da criptare di qualsiasi lunghezza. Ovviamente bisognerà tenere conto di questo algoritmo in fase di decodifica.

d) Un altro vincolo (in realtà non è un vero e proprio vincolo quanto piuttosto il modo con cui funziona l'algoritmo di codifica), è che la chiave di cripting deve essere almeno di 64 bits (come detto, l'algoritmo DES, cripta a blocchi di 64 bits) ovvero 8 caratteri. Se infatti, come "key_string" scelgo un valore più piccolo di 8 bytes (1 byte = 1 carattere), ottengo il seguente errore:

```
SQL> insert into login values ('andrea',encrypt('anpwd'));
insert into login values ('andrea',encrypt('anpwd'))
```

*

```
ERROR at line 1:
ORA-28234: key length too short
ORA-06512: at "SYS.DBMS_OBFUSCATION_TOOLKIT_FFI", line 0
ORA-06512: at "SYS.DBMS_OBFUSCATION_TOOLKIT", line 33
ORA-06512: at "ANDREA.ENCRYPT", line 13
ORA-06512: at line 1
```

2. Decrypting

La funzione di DESDecrypt(), ha 3 argomenti:

PROCEDURE DESDECRYPT			
Argument Name	Type	In/Out	Default?
INPUT_STRING	VARCHAR2	IN	
KEY_STRING	VARCHAR2	IN	
DECRYPTED_STRING	VARCHAR2	OUT	

Come si può vedere, ci sono 2 parametri di input: la stringa da decryptare e la chiave da utilizzare per il cripting, ed un solo parametro di output dove viene memorizzata la stringa decryptata. Questa procedura è overloaded con una equivalente con parametri di tipo RAW, per criptare dati di tipo RAW.

La procedura DES3Decrypt(), ha invece 4 argomenti:

PROCEDURE DES3DECRYPT			
Argument Name	Type	In/Out	Default?
INPUT_STRING	VARCHAR2	IN	
KEY_STRING	VARCHAR2	IN	
DECRYPTED_STRING	VARCHAR2	OUT	
WHICH	BINARY_INTEGER	IN	DEFAULT

Anche per tale procedura vale il discorso fatto per DES3Encrypt(): rispetto a DESDecrypt() ha un argomento in più: "which". Se impostato a "0" (zero), il valore di default, allora viene utilizzato una chiave di cripting a 128 bits (TwoKeyMode o 2-key); se invece è impostato ad 1, allora viene utilizzato una chiave di cripting a 192 bits (ThreeKeyMode o 3-key). In figura 2 c'è un esempio di funzione di cripting.

```

create or replace function decrypt (
    input_string IN varchar2
)
return varchar2
is
    data_decrypt_out  varchar2(256);

begin
    dbms_obfuscation_toolkit.DESDecrypt(
        input_string => input_string,
        key_string   => 'AnDrEaAn',
        encrypted_string=> data_decrypt_out);

    return rtrim( data_decrypt_out, chr(0) );
end;

```

- Figura 2 - Funzione di decrypt -

Commentiamo il codice. In questo caso, la rilevanza principale risiede nel valore restituito dalla funzione. Prima di restituire la stringa decodificata, va tenuto conto che la sua lunghezza era stata resa multipla di 8. Per “denormalizzare” tale stringa, vanno semplicemente eliminati eventuali spazi bianchi presenti a destra (aggiunti in fase di codifica). Per fare questo si utilizza la funzione RTRIM, la cui sintassi è:

```
RTRIM ( char [, set ] )
```

La funzione RTRIM rimuove tutte le sequenze di caratteri passati come secondo argomento. Di default, l’insieme di caratteri da cancellare è lo spazio (chr(0)). Nel nostro caso:

```

char -> data_decrypt_out
set  -> chr(0)

```

Riprendendo l’esempio precedente, abbiamo che:

INPUT	OUTPUT
a[][][][][][][]*	
rtrim('a[][][][][][][]',chr(0))	a

* il simbolo [] rappresenta uno spazio

3. DBMS_OBFUSCATION_TOOLKIT

Diamo un accenno veloce ai moduli presenti all’interno del package DBMS_OBFUSCATION_TOOLKIT: di tutti quelli presenti, ce ne sono 4 a cui siamo veramente interessati. Ognuna di questa unità è “overloded” così che c’è una procedura ed una funzione per ogni unità di programma, più le duplicazioni

che permettono la gestione di differenti tipi di dato. Gli esempi trattati fanno riferimento sia a procedure, paragrafi 1) e 2) che a funzioni, paragrafo 5.3). In tutti i casi comunque viene fatto riferimento alle versioni con dati di tipo testo e non a dati di tipo RAW.

Di seguito, la descrizione del package:

```
SQL> desc dbms_obfuscation_toolkit
PROCEDURE DES3DECRYPT
Argument Name                                Type                                In/Out Default?
-----
INPUT                                        RAW                                IN
KEY                                        RAW                                IN
DECRYPTED_DATA                              RAW                                OUT
WHICH                                       BINARY_INTEGER                    IN      DEFAULT
FUNCTION DES3DECRYPT RETURNS RAW
Argument Name                                Type                                In/Out Default?
-----
INPUT                                        RAW                                IN
KEY                                        RAW                                IN
WHICH                                       BINARY_INTEGER                    IN      DEFAULT
PROCEDURE DES3DECRYPT
Argument Name                                Type                                In/Out Default?
-----
INPUT_STRING                               VARCHAR2                           IN
KEY_STRING                                 VARCHAR2                           IN
DECRYPTED_STRING                           VARCHAR2                           OUT
WHICH                                       BINARY_INTEGER                    IN      DEFAULT
FUNCTION DES3DECRYPT RETURNS VARCHAR2
Argument Name                                Type                                In/Out Default?
-----
INPUT_STRING                               VARCHAR2                           IN
KEY_STRING                                 VARCHAR2                           IN
WHICH                                       BINARY_INTEGER                    IN      DEFAULT
PROCEDURE DES3ENCRYPT
Argument Name                                Type                                In/Out Default?
-----
INPUT                                        RAW                                IN
KEY                                        RAW                                IN
ENCRYPTED_DATA                              RAW                                OUT
WHICH                                       BINARY_INTEGER                    IN      DEFAULT
FUNCTION DES3ENCRYPT RETURNS RAW
Argument Name                                Type                                In/Out Default?
-----
INPUT                                        RAW                                IN
KEY                                        RAW                                IN
WHICH                                       BINARY_INTEGER                    IN      DEFAULT
PROCEDURE DES3ENCRYPT
Argument Name                                Type                                In/Out Default?
-----
INPUT_STRING                               VARCHAR2                           IN
KEY_STRING                                 VARCHAR2                           IN
ENCRYPTED_STRING                           VARCHAR2                           OUT
WHICH                                       BINARY_INTEGER                    IN      DEFAULT
```

```

FUNCTION DES3ENCRYPT RETURNS VARCHAR2
Argument Name          Type          In/Out Default?
-----
INPUT_STRING          VARCHAR2      IN
KEY_STRING            VARCHAR2      IN
WHICH                 BINARY_INTEGER IN      DEFAULT
PROCEDURE DESDECRYPT
Argument Name          Type          In/Out Default?
-----
INPUT                 RAW           IN
KEY                   RAW           IN
DECRYPTED_DATA        RAW           OUT
FUNCTION DESDECRYPT RETURNS RAW
Argument Name          Type          In/Out Default?
-----
INPUT                 RAW           IN
KEY                   RAW           IN
PROCEDURE DESDECRYPT
Argument Name          Type          In/Out Default?
-----
INPUT_STRING          VARCHAR2      IN
KEY_STRING            VARCHAR2      IN
DECRYPTED_STRING      VARCHAR2      OUT
FUNCTION DESDECRYPT RETURNS VARCHAR2
Argument Name          Type          In/Out Default?
-----
INPUT_STRING          VARCHAR2      IN
KEY_STRING            VARCHAR2      IN
PROCEDURE DESENCRYPT
Argument Name          Type          In/Out Default?
-----
INPUT                 RAW           IN
KEY                   RAW           IN
ENCRYPTED_DATA        RAW           OUT
FUNCTION DESENCRYPT RETURNS RAW
Argument Name          Type          In/Out Default?
-----
INPUT                 RAW           IN
KEY                   RAW           IN
PROCEDURE DESENCRYPT
Argument Name          Type          In/Out Default?
-----
INPUT_STRING          VARCHAR2      IN
KEY_STRING            VARCHAR2      IN
ENCRYPTED_STRING      VARCHAR2      OUT
FUNCTION DESENCRYPT RETURNS VARCHAR2
Argument Name          Type          In/Out Default?
-----
INPUT_STRING          VARCHAR2      IN
KEY_STRING            VARCHAR2      IN
PROCEDURE MD5
Argument Name          Type          In/Out Default?
-----
INPUT                 RAW           IN
CHECKSUM              RAW(16)       OUT

```

```

FUNCTION MD5 RETURNS RAW(16)
Argument Name                Type                In/Out Default?
-----
INPUT                        RAW                IN
PROCEDURE MD5
Argument Name                Type                In/Out Default?
-----
INPUT_STRING                 VARCHAR2           IN
CHECKSUM_STRING              VARCHAR2           OUT
FUNCTION MD5 RETURNS VARCHAR2(16)
Argument Name                Type                In/Out Default?
-----
INPUT_STRING                 VARCHAR2           IN

```

L'output sopra mostrato è stato preso da Oracle 8i. La documentazione tuttavia riporta solo le procedure DESEncrypt e DESDecrypt. La documentazione Oracle 9i invece descrive anche le procedure DES3Encrypt e DES3Decrypt, mentre quella di Oracle 10g riporta e spiega tutti moduli sopra elencati. Consiglio quindi di consultare la documentazione ufficiale al seguente link http://download-west.oracle.com/docs/cd/B14117_01/appdev.101/b10802/d_obtool.htm#1002213, per tutte le informazioni necessarie.

4. Offuscare il codice

Oracle offre la possibilità di nascondere il proprio codice PL/SQL, attraverso un'utility da linea di comando. In fase di installazione del software, nella ORACLE_HOME/bin, viene scaricato il binario "wrap". Questo accetta due parametri: iname ed oname. Il primo specifica il file che contiene il codice da offuscare, mentre il secondo il file che conterrà il codice offuscato. Di seguito un esempio:

```
oracle> wrap iname=codice_chiaro.sql oname=codice_nascosto.sql
```

```
PL/SQL Wrapper: Release 9.2.0.6.0- Production on Thu Apr 07 14:03:42
2005
```

```
Copyright (c) Oracle Corporation 1993, 2001. All Rights Reserved.
```

```
processing codice_chiaro.sql to codice_nascosto.sql
```

e quindi

```
oracle> sqlplus andrea/andpwd
```

```
SQL> @codice_nascosto.sql
```

Procedendo in questo modo, si nasconde il codice in cui è inserita la chiave di cripting e decrypting.

5. Conclusioni

Le due funzioni riportate nei paragrafi 1) e 2), utilizzano un semplice metodo per criptare i dati. Oracle mette a disposizione una funzione di hashing che potrebbe essere utilizzata ad esempio per generare la chiave utilizzata dalla funzione ENCRYPT. In realtà, nelle versioni precedenti di Oracle, dove non esisteva ancora il package DBMS_OBFUSCATION_TOOLKIT, veniva utilizzata proprio tale funzione per criptare i dati. Un esempio è riportato in 5.1).

Le due funzioni ENCRYPT e DECRYPT devono essere prese solo come riferimento per la codifica dei dati: il loro utilizzo dovrebbe essere all'interno di stored procedure per la gestione dei campi password. Ovviamente esiste un'infinità di modi in cui possono essere utilizzate.

Per la gestione del campo password di una base dati, si potrebbe ad esempio utilizzare la seguente tecnica: per l'inserimento, si crea un trigger (in 5.2)) che in modo del tutto trasparente codifica il campo password; mentre per determinare se un utente può accedere o meno, si sfrutta un'API (in 5.3)).

Supponiamo, per semplicità di cose, che la tabella utilizzata sia composta da soli due campi: lo username, che è anche PRIMARY KEY (non ha senso avere due utenti con la stessa utenza di accesso), e la password con vincolo di NOT NULL (la password è vincolante per la sicurezza di accesso. Inoltre non è da escludere che utenti diversi abbiano la stessa password: per tale motivo non esiste sul campo password il vincolo di univocità):

```
SQL> create table tb_login (  
2  username varchar2(15) primary key,  
3  password varchar2(15) not null  
4  );
```

Table created.

5.1. Funzione di HASH

Prima di Oracle 8, dove non esisteva il package di offuscamento dei dati, si utilizzava la funzione di hash GET_HASH_VALUE, presente nel package DBMS_UTILITY, per generare valori codificati. Tale funzione di Hash ha la seguente forma:

```
DBMS_UTILITY.GET_HASH_VALUE (  
  name      VARCHAR2,  
  base      NUMBER,  
  hash_size NUMBER)  
RETURN NUMBER;
```

Il significato dei parametri di input è il seguente:

name - Stringa di cui fare l'hash
base - Valore base dal quale iniziare per il valore di hash restituito
hash_size - Dimensione desiderata dell'hash table

Questa funzione restituisce un valore di hash basato su una stringa di input. Per esempio, per ottenere su una stringa un valore di hash che dovrebbe essere tra 1000 e 3047, usa 1000 come valore di base (parametro base) e 2048 come valore di hash_size. L'algoritmo di generazione del valore di hash funziona meglio se si utilizza una potenza di due per il parametro hash_size.

Partendo da questo, possiamo costruire la funzione di cripting "hsencrypt":

```
CREATE OR REPLACE FUNCTION hsencrypt (  
    p_username IN VARCHAR2,  
    p_password IN VARCHAR2  
)  
RETURN VARCHAR2  
IS  
BEGIN  
  
RETURN LTRIM (  
    TO_CHAR (  
        DBMS_UTILITY.GET_HASH_VALUE (  
            UPPER(p_username) || '/' || UPPER(p_password),  
            1000000000,  
            POWER(2,30)),  
        RPAD('X',29,'X') || 'X'  
    );  
END;
```

GET_HASH_VALUE, restituisce un intero (HASH) basato su username e password, all'interno dell'intervallo 1-1073741824 (estremi compresi), e vi aggiunge 1000000000 (per renderlo più grande). Viene poi calcolato e restituito il valore esadecimale dell'HASH precedente ottenuto attraverso la funzione TO_CHAR².

5.2. Trigger

Come accennato precedentemente, il trigger è pensato in modo tale da svincolare chi programma di preoccuparsi di scrivere codice adhoc per la codifica dei dati. In particolare in questo modo si evitano statement sql in cui sia presente la funzione di codifica. Esistono diversi modi per codificare i dati. Si potrebbe ad esempio pensare di creare separatamente la funzione di codifica e di utilizzarla

² La funzione TO_CHAR ammette due parametri: il primo è il numero, il secondo è il formato di output. Il formato XXXX, restituisce il valore esadecimale del numero di digits specificati.

poi nel trigger. Nel mio caso invece ho voluto creare l'algoritmo di cripting, all'interno del trigger stesso, utilizzando la funzione di hash vista in 5.1).

```
create or replace trigger tr_encrypt
before insert on tb_login
for each row
declare
    data_crypt varchar2(256);
    data_crypt_out varchar2(256);
    hash_key_string varchar2(256);

begin
    data_crypt := rpad( :new.password,
        (trunc(length(:new.password)/8)+1)*8, chr(0) );

    hash_key_string := TO_CHAR (
        DBMS_UTILITY.GET_HASH_VALUE (
            UPPER(:new.username)||'/'||UPPER(:new.password),
            1000000000, POWER(2,30)));

    dbms_obfuscation_toolkit.DESEncrypt(
        input_string => data_crypt,
        key_string    => hash_key_string,
        encrypted_string=> data_crypt_out);

    :new.password:= data_crypt_out;
end;
/
```

In questo caso, sarà la funzione di HASH a generare la chiave da fornire al package DBMS_OBFUSCATION_TOOLKIT. L'idea è quella di usare una chiave diversa per ogni username. Ovviamente il codice del trigger va offuscato come indicato nel paragrafo 4).

5.3. API

La funzione che segue determina se un utente è presente o meno nella base dati. Vediamo dapprima il codice e poi discutiamo quanto scritto.

```
Create or replace function api_decrypt (
    p_username IN varchar2,
    p_password IN varchar2
)
RETURN VARCHAR2
AS

    cursor c_pwd (USER VARCHAR2) is
```

```

select password from tb_login
where username=p_username;

normalized_password varchar2(256);
hash_key_string varchar2(256);
cripted_password varchar2(256);
stored_password login.password%type;

responce varchar2(2);
BEGIN

-- Normalizzazione della password in multipli di 8 bytes
normalized_password := rpad( p_password,
    (trunc(length(p_password)/8)+1)*8, chr(0));

-- Generazione della chiave di hash
hash_key_string := TO_CHAR (
    DBMS_UTILITY.GET_HASH_VALUE (
        UPPER(p_username)||'/'||UPPER(p_password),
        1000000000, POWER(2,30)));

-- Codifica della password
cripted_password:=dbms_obfuscation_toolkit.DESEncrypt(
    input_string => normalized_password,
    key_string   => hash_key_string);

-- Selezione della password nel db
open c_pwd(p_username);
fetch c_pwd into stored_password;

-- Confronto delle password
if stored_password=cripted_password then
    responce := 'OK';
else
    responce := 'KO';
end if;

RETURN(responce);
END;

```

Innanzitutto viene normalizzata la password in modo da rendere la sua lunghezza un multiplo di 8 bytes. Poi viene calcolata la chiave di codifica attraverso la funzione di hash, GET_HASH_VALUE. Con questi due valori, si utilizza la funzione DESEncrypt³ per codificare la password. Una volta selezionata la password storicizzata nel db, viene confrontata con quella

³ Come si può vedere in questo caso utilizzo DESEncrypt come funzione e non come procedura.

codificata: se esiste una corrispondenza tra i due valori, allora l'API chiamata restituisce OK, altrimenti un KO.

5.4. Oracle 10g

Oracle 10g mette a disposizione un nuovo package: DBMS_CRYPTO (http://download-west.oracle.com/docs/cd/B14117_01/appdev.101/b10802/d_crypto.htm#1005074) che vuol sostituire DBMS_OBFUSCATION_TOOLKIT. Questa procedura estende il supporto agli algoritmi di codifica: tra i tanti c'è AES (Advanced Encryption Standard). E' inoltre esteso il supporto ai tipi di dati supportati.

5.5. Errori

Esistono altri errori oltre quelli incontrati, legati al package DBMS_OBFUSCATION_TOOLKIT. Di seguito riporto la lista estratta dal manuale degli errori (http://download-west.oracle.com/docs/cd/B10501_01/server.920/a96525/e19400.htm#981286):

ORA-28231 no data passed to obfuscation toolkit

Cause: A NULL value was passed to a function or procedure.

Action: Make sure that the data passed is not empty.

ORA-28232 invalid input length for obfuscation toolkit

Cause: Length of data submitted for encryption or decryption is not a multiple of 8 bytes.

Action: Make sure that the length of the data to be encrypted or decrypted is a multiple of 8 bytes.

ORA-28233 double encryption not supported

Cause: The obfuscation toolkit does not support the encryption of already-encrypted data.

Action: Do not attempt to encrypt already-encrypted data.

ORA-28234 key length too short

Cause: The key specified is too short for the algorithm. DES requires a key of at least 8 bytes. Triple DES requires a key of at least 16 bytes in two-key mode and 24 bytes in three-key mode.

Action: Specify a longer key.

ORA-28235 algorithm not available

Cause: The desired encryption algorithm is not available.

Action: Run the installer to install the needed algorithm in Oracle Advanced Security.

ORA-28236 invalid Triple DES mode

Cause: An unknown value was specified for the mode in which triple DES encryption is to run.

Action: Specify a valid value. See the package declaration for a list of valid values.

ORA-28237 seed length too short

Cause: The seed specified for the key generation routine must be at least 80 characters.

Action: Specify a longer seed.

ORA-28238 no seed provided

Cause: A NULL value was passed in as the seed to be used in generating a key.

Action: Provide a non-NULL value for the seed.

ORA-28239 no key provided

Cause: A NULL value was passed in as an encryption or decryption key.

Action: Provide a non-NULL value for the key.

6. Riferimenti

- ✓ Esempio da cui ho creato le funzioni di cripting e decriptino:

http://asktom.oracle.com/pls/ask/f?p=4950:8:::::F4950_P8_DISPLAYID:791026226790

- ✓ Utile link sul cripting ed il decriptino

http://www.quest-pipelines.com/newsletter-v4/0703_C.htm

- ✓ Esempio di funzione di hash

http://asktom.oracle.com/pls/ask/f?p=4950:8:::::F4950_P8_DISPLAYID:95412348059

- ✓ Oracle 10g

http://otn.oracle.com/pls/db10g/portal.portal_demo3?selected=1

- ✓ Oracle 10g: DBMS_CRYPT

http://download-west.oracle.com/docs/cd/B14117_01/appdev.101/b10802/d_crypto.htm#1005074

✓ Manuale degli errori di Oracle 9i

http://download-west.oracle.com/docs/cd/B10501_01/server.920/a96525/toc.htm