CSC12001 – An toàn bảo mật CSDL trong HTTT

Tháng 1/2020

DBMS Encryption

Tóm tắt nội dung bài thực hành:

Hướng dẫn phương pháp bảo mật ở mức độ mã hóa dữ liệu trong CSDL

Bộ môn **Hệ thống thông tin**
Khoa Công nghệ thông tin
ĐH Khoa học tự nhiên TP HCM

MỤC LỤC

1 Các kiểu dữ liệu hỗ trợ mã hóa 3

2 Các thuật toán băm 3

3 Các thuật toán mã hóa 3

4 Các phương pháp chuyển đổi khối mật mã 4

5 Padding 5

6 Cấu trúc hàm mã hóa/giải mã 6

1 Các kiểu dữ liệu hỗ trợ mã hóa

Tên	Mô tả
BLOB	Binary Large Object
CLOB	Character Large Object (bao gồm NCLOB)
PLS_INTEGER	Xác định loại thuật toán mã hóa (dùng với các kiểu dữ liệu BLOB, CLOB, và RAW)
RAW	Dữ liệu thô

Lưu ý: Chỉ hỗ trợ mã hóa trên các kiểu dữ liệu BLOB, CLOB, PLS_INTEGER, RAW. Đối với kiểu dữ liệu VARCHAR2, trước khi thực thi phương thức mã hóa, phải chuyển nó sang tập ký tự **AL32UTF8**, sau đó chuyển nó sang dạng dữ liệu thô **RAW**. Khi đó, có thể dùng gói DBMS_CRYPTO mã hóa dữ liệu.

2 Các thuật toán băm

1. HASH

Tên	Mô tả
HASH_MD4	Băm giá trị ban đầu thành giá trị băm 128 bit.
HASH_MD5	Băm giá trị ban đầu cho ra kết quả băm 128 bit, nhưng phức tạp hơn HASH_MD4.
HASH_SH1	SHA (Secure Hash Algorithm), tạo ra kết quả băm 160 bit.

2. MAC: là thuật toán băm, nhưng có thêm giá trị khóa để xác nhận kết quả băm

Tên	Mô tả
HMAC_MD5	Tương tự thuật toán băm MD5.
HMAC_SH1	Tương tự thuật toán băm SHA

3 Các thuật toán mã hóa

Tên	Mô tả
ENCRYPT_DES	DES-Data Encryption Standard. Block cipher. Sử dụng khóa có độ dài 56 bits

ENCRYPT_3DES_2KEY	Data Encryption Standard. Block cipher. Thực hiện trên khối 3 lần với 2 khóa. Chiều dài khóa hiệu quả là 112 bits.
ENCRYPT_3DES	Data Encryption Standard. Block cipher. Thực hiện trên khối 3 lần. Độ dài khóa 192 bit.
ENCRYPT_AES128	Advanced Encryption Standard. Block cipher. Sử dụng khóa có kích thước 128 bit.
ENCRYPT_AES192	Advanced Encryption Standard. Block cipher. Sử dụng khóa có kích thước 192 bit.
ENCRYPT_AES256	Advanced Encryption Standard. Block cipher. Sử dụng khóa có kích thước 256 bit.
ENCRYPT_RC4	Stream cipher. Sử dụng một khóa đơn nhất được tạo ra một cách bí mật và ngẫu nhiên cho mỗi phiên làm việc.

4 Các phương pháp chuyển đổi khối mật mã

Quá trình mã hóa sẽ chia dữ liệu thành các khối (block) với kích thước định sẵn(block-size) tùy theo thuật toán mã hóa. Các khối này sau khi mã hóa sẽ hợp lại theo các phương pháp khác nhau thành dữ liệu mã hóa.

Tên	Mô tả
CHAIN_ECB	Electronic Codebook. Các khối mã hóa hoàn toàn độc lập với nhau, được sắp xếp tuần tự theo thứ tự ban đầu (trước khi mã hóa) để tạo thành dữ liệu mã hóa. ECB không che giấu được các mẫu dữ liệu.
CHAIN_CBC	Cipher Block Chaining. Khối Plaintext kế tiếp được XOR với khối ciphertext trước đó trước khi nó được mã hóa. Vector khởi tạo (IV) được xem như khối plaintext đầu tiên.
CHAIN_CFB	Cipher-Feedback. Plaintext không được mã hóa bằng chính thuật toán đang xét mà được mã hóa bằng cách XOR với một chuỗi được tạo ra bằng thuật toán mã hóa. Kết quả này làm đầu vào cho khối kế tiếp. Phương pháp CFB biến Block Cipher thành Stream Cipher.
CHAIN_OFB	Output-Feedback. Tương tự như CFB, plaintext cũng được XOR với một chuỗi được tạo ra bằng thuật toán mã hóa, nhưng đầu vào cho khối kế tiếp là chuỗi mã hóa được tạo ra bằng thuật toán mã hóa. Phương pháp OFB cũng biến Block Cipher thành Stream Cipher

5 Padding

Các khối mã hóa phải đúng kích thước khối do thuật toán qui định, nếu không, khối phải được đệm thêm (pad) cho đúng kích thước khối. Đây là quá trình làm tăng kích thước dữ liệu sau khi mã hóa.

Tên	Mô tả
PAD_PKCS5	Cung cấp cơ chế đệm thêm tuân theo chuẩn PKCS #5: Password-Based Cryptography Standard: thêm vào n số, chỉ n byte còn thiếu của khối cuối cùng. Nếu khối cuối cùng đủ, vẫn đệm thêm toàn bộ khối.
PAD_NONE	Cung cấp lựa chọn chỉ định khỏi đệm. Đối tượng gọi phải đảm bảo rằng kích thước khối phải đúng, nếu không, package sẽ báo lỗi.
PAD_ZERO	Cung cấp cơ chế đệm thêm số 0 vào khối cuối cùng cho các byte còn thiếu.

6 Cấu trúc hàm mã hóa/giải mã

Encrypt – Decrypt Function: hàm mã hóa dữ liệu kiểu RAW

```
DBMS_CRYPTO.ENCRYPT(
   src IN RAW,
   typ IN PLS_INTEGER,
   key IN RAW,
   iv  IN RAW          DEFAULT NULL)
 RETURN RAW;
```

```
DBMS_CRYPTO.DECRYPT(
   src IN RAW,
   typ IN PLS_INTEGER,
   key IN RAW,
   iv  IN RAW DEFAULT NULL)
 RETURN RAW;
```

Tham số	Mô tả
src	Dữ liệu đầu vào, kiểu RAW.
typ	Thuật toán mã hóa – giải mã sẽ sử dụng, là các số nguyên đã được đinh nghĩa (PLS_INTEGER), bao gồm thuật toán mã hóa, phương pháp chuyển đổi khối mật mã và kiểu đệm thêm.
key	Khóa dùng để mã hóa – giải mã, kiểu dữ liệu thô RAW.
iv	Vector khởi tạo cho các khối mã hóa, kiểu RAW. Tùy chọn (có thể không có). Mặc định là NULL

Ví dụ: sử dụng bộ thuật toán mã hóa DES_CBC_PKCS5

```
DECLARE
    input_string     VARCHAR2(16) := 'tigertigertigert';
    raw_input        RAW(128) :=
UTL_RAW.CAST_TO_RAW(CONVERT(input_string,'AL32UTF8','US7ASCII'));
    key_string       VARCHAR2(8)  := 'scottsco';
    raw_key          RAW(128) :=
UTL_RAW.CAST_TO_RAW(CONVERT(key_string,'AL32UTF8','US7ASCII'));
    encrypted_raw    RAW(2048);
    encrypted_string VARCHAR2(2048);
    decrypted_raw    RAW(2048);
    decrypted_string VARCHAR2(2048);

-- 1. Begin testing Encryption
BEGIN
    dbms_output.put_line('> Input String : ' ||
    CONVERT(UTL_RAW.CAST_TO_VARCHAR2(raw_input),'US7ASCII','AL32UTF8'));
    dbms_output.put_line('> ========= BEGIN TEST Encrypt =========');

-- encrypt and decrypt functions in DBMS_CRYPTO package work on the RAW data
type only
    encrypted_raw := dbms_crypto.Encrypt(src => raw_input,
                                         typ => DBMS_CRYPTO.DES_CBC_PKCS5,
                                         key => raw_key);
```

```
        dbms_output.put_line('> Encrypted hex value  : ' ||
rawtohex(UTL_RAW.CAST_TO_RAW(encrypted_raw)));

        decrypted_raw := dbms_crypto.Decrypt(src => encrypted_raw,
                                             typ => DBMS_CRYPTO.DES_CBC_PKCS5,
                                             key => raw_key);
    decrypted_string :=
    CONVERT(UTL_RAW.CAST_TO_VARCHAR2(decrypted_raw),'US7ASCII','AL32UTF8');
dbms_output.put_line('> Decrypted string output : ' || decrypted_string);
if  input_string = decrypted_string THEN
    dbms_output.put_line('> String DES Encyption and Decryption successful');
END if;
dbms_output.put_line('');

--Shows how to create a 160-bit hash using SHA-1 algorithm
dbms_output.put_line('> ========= BEGIN TEST Hash =========');
    encrypted_raw := dbms_crypto.Hash(src => raw_input,
                                      typ => DBMS_CRYPTO.HASH_SH1);
dbms_output.put_line('> Hash value of input string: ' ||
        rawtohex(UTL_RAW.CAST_TO_RAW(encrypted_raw)));

-- Demonstrates how MAC, a key-dependent one-way hash, can be computed using
MD5 algorithm
dbms_output.put_line('> ========= BEGIN TEST Mac =========');
    encrypted_raw := dbms_crypto.Mac(src => raw_input,
                                     typ => DBMS_CRYPTO.HMAC_MD5,
                                     key => raw_key);
dbms_output.put_line('> Message Authentication Code  : ' ||
        rawtohex(UTL_RAW.CAST_TO_RAW(encrypted_raw)));
dbms_output.put_line('');
dbms_output.put_line('> End of DBMS_CRYPTO tests  ');
END;
```

Ví dụ: sử dụng thuật toán mã hóa ENCRYPT_AES256 với CHAIN_CBC và PAD_PKCS

```
declare

   input_string       VARCHAR2 (200) := 'Secret Message';

   output_string      VARCHAR2 (200);

   encrypted_raw      RAW (2000);      -- stores encrypted binary text

   decrypted_raw      RAW (2000);      -- stores decrypted binary text

   num_key_bytes      NUMBER := 256/8;  -- key length 256 bits (32 bytes)

   key_bytes_raw      RAW (32);         -- stores 256-bit encryption key

   encryption_type    PLS_INTEGER :=    -- total encryption type

                            DBMS_CRYPTO.ENCRYPT_AES256

                          + DBMS_CRYPTO.CHAIN_CBC

                          + DBMS_CRYPTO.PAD_PKCS5;
```

```
begin
   DBMS_OUTPUT.PUT_LINE ('Original string: ' || input_string);
   key_bytes_raw := DBMS_CRYPTO.RANDOMBYTES (num_key_bytes);
   encrypted_raw := DBMS_CRYPTO.ENCRYPT
      (  src => UTL_I18N.STRING_TO_RAW (input_string, 'AL32UTF8'),
         typ => encryption_type,
         key => key_bytes_raw
      );

    -- The encrypted value in the encrypted_raw variable can be used here
   decrypted_raw := DBMS_CRYPTO.DECRYPT
      (
         src => encrypted_raw,
         typ => encryption_type,
         key => key_bytes_raw
      );
   output_string := UTL_I18N.RAW_TO_CHAR (decrypted_raw, 'AL32UTF8');
   DBMS_OUTPUT.PUT_LINE ('Decrypted string: ' || output_string);
end;
```

Encrypt – Decrypt Procedures: các hàm mã hóa cho dữ liệu lớn LOB

```
DBMS_CRYPTO.ENCRYPT(
   dst IN OUT NOCOPY BLOB,
   src IN            BLOB,
   typ IN            PLS_INTEGER,
   key IN            RAW,
   iv  IN            RAW          DEFAULT NULL);

DBMS_CRYPTO.ENCRYPT(
   dst IN OUT NOCOPY BLOB,
   src IN            CLOB         CHARACTER SET ANY_CS,
   typ IN            PLS_INTEGER,
   key IN            RAW,
   iv  IN            RAW          DEFAULT NULL);
```

```
DBMS_CRYPTO.DECRYPT(
   dst IN OUT NOCOPY BLOB,
   src IN            BLOB,
   typ IN            PLS_INTEGER,
   key IN            RAW,
   iv  IN            RAW          DEFAULT NULL);

DBMS_CRYPT.DECRYPT(
   dst IN OUT NOCOPY CLOB         CHARACTER SET ANY_CS,
   src IN            BLOB,
   typ IN            PLS_INTEGER,
   key IN            RAW,
   iv  IN            RAW          DEFAULT NULL);
```

Tham số	Mô tả
dst	Chứa dữ liệu đầu ra kiểu BLOB, giá trị của biến sẽ bị ghi đè nếu biến đã có giá trị.
src	Dữ liệu đầu vào, có kiểu BLOB hay CLOB CHARACTER SET ANY_CS.
typ	Thuật toán mã hóa – giải mã sẽ sử dụng, là các số nguyên đã được đinh nghĩa (PLS_INTEGER), bao gồm thuật toán mã hóa – giải mã, phương pháp chuyển đổi khối mật mã và kiểu đệm thêm.
key	Khóa dùng để mã hóa – giải mã, kiểu dữ liệu thô LOB.
iv	Vector khởi tạo do người dùng chọn, kiểu LOB. Tham số này là tùy chọn, có thể không có. Mặc định là NULL.

Ví dụ:

```
-- Create a table for BLOB column.
create table table_lob (id number, loc blob);

-- insert 3 empty lobs for src/enc/dec
insert into table_lob values (1, EMPTY_BLOB());
insert into table_lob values (2, EMPTY_BLOB());
insert into table_lob values (3, EMPTY_BLOB());

set echo on
set serveroutput on

declare
    srcdata    RAW(1000);
    srcblob    BLOB;
    encrypblob BLOB;
    encrypraw  RAW(1000);
    encrawlen  BINARY_INTEGER;
    decrypblob BLOB;
    decrypraw  RAW(1000);
    decrawlen  BINARY_INTEGER;

    leng       INTEGER;

begin
     -- RAW input data 16 bytes
    srcdata := hextoraw('6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D');

    dbms_output.put_line('---');
    dbms_output.put_line('input is ' || srcdata);
    dbms_output.put_line('---');

    -- select empty lob locators for src/enc/dec
    select loc into srcblob from table_lob where id = 1;
    select loc into encrypblob from table_lob where id = 2;
    select loc into decrypblob from table_lob where id = 3;

    dbms_output.put_line('Created Empty LOBS');
    dbms_output.put_line('---');

    leng := DBMS_LOB.GETLENGTH(srcblob);
    IF leng IS NULL THEN
```

```
        dbms_output.put_line('Source BLOB Len NULL ');
    ELSE
        dbms_output.put_line('Source BLOB Len ' || leng);
    END IF;

    leng := DBMS_LOB.GETLENGTH(encrypblob);
    IF leng IS NULL THEN
        dbms_output.put_line('Encrypt BLOB Len NULL ');
    ELSE
        dbms_output.put_line('Encrypt BLOB Len ' || leng);
    END IF;

    leng := DBMS_LOB.GETLENGTH(decrypblob);
    IF leng IS NULL THEN
        dbms_output.put_line('Decrypt  BLOB Len NULL ');
    ELSE
        dbms_output.put_line('Decrypt BLOB Len ' || leng);
    END IF;

    -- write source raw data into blob
    DBMS_LOB.OPEN (srcblob, DBMS_LOB.lob_readwrite);
    DBMS_LOB.WRITEAPPEND (srcblob, 16, srcdata);
    DBMS_LOB.CLOSE (srcblob);

    dbms_output.put_line('Source raw data written to source blob');
    dbms_output.put_line('---');

    leng := DBMS_LOB.GETLENGTH(srcblob);
    IF leng IS NULL THEN
        dbms_output.put_line('source BLOB Len NULL ');
    ELSE
        dbms_output.put_line('Source BLOB Len ' || leng);
    END IF;

    /*
    * Procedure Encrypt
    * Arguments: srcblob -> Source BLOB
    *            encrypblob -> Output BLOB for encrypted data
    *            DBMS_CRYPTO.AES_CBC_PKCS5 -> Algo : AES
    *                                         Chaining : CBC
```

```
    *                                                Padding : PKCS5
    *             256 bit key for AES passed as RAW
    *                 ->
hextoraw('000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F')
  *       IV (Initialization Vector) for AES algo passed as RAW
    *                 -> hextoraw('00000000000000000000000000000000')
    */

    DBMS_CRYPTO.Encrypt(encrypblob,
                srcblob,
                DBMS_CRYPTO.AES_CBC_PKCS5,
                hextoraw
('000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F'),
                hextoraw('00000000000000000000000000000000'));


    dbms_output.put_line('Encryption Done');
    dbms_output.put_line('---');

    leng := DBMS_LOB.GETLENGTH(encrypblob);
    IF leng IS NULL THEN
        dbms_output.put_line('Encrypt BLOB Len NULL');
    ELSE
        dbms_output.put_line('Encrypt BLOB Len ' || leng);
    END IF;

    -- Read encrypblob to a raw
    encrawlen := 999;

    DBMS_LOB.OPEN (encrypblob, DBMS_LOB.lob_readwrite);
    DBMS_LOB.READ (encrypblob, encrawlen, 1, encrypraw);
    DBMS_LOB.CLOSE (encrypblob);

    dbms_output.put_line('Read encrypt blob to a raw');
    dbms_output.put_line('---');

    dbms_output.put_line('Encrypted data is (256 bit key) ' || encrypraw);
    dbms_output.put_line('---');

    /*
    * Procedure Decrypt
    * Arguments: encrypblob -> Encrypted BLOB to decrypt
```

```
    *             decrypblob -> Output BLOB for decrypted data in RAW
    *             DBMS_CRYPTO.AES_CBC_PKCS5 -> Algo : AES
    *                                          Chaining : CBC
    *                                          Padding : PKCS5
    *             256 bit key for AES passed as RAW (same as used during
Encrypt)
    *                 ->
hextoraw('000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F')
    *             IV (Initialization Vector) for AES algo passed as RAW (same
as
                  used during Encrypt)
    *                 -> hextoraw('00000000000000000000000000000000')
    */

    DBMS_CRYPTO.Decrypt(decrypblob,
                encrypblob,
                DBMS_CRYPTO.AES_CBC_PKCS5,
                hextoraw
('000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F'),
                hextoraw('00000000000000000000000000000000'));

    leng := DBMS_LOB.GETLENGTH(decrypblob);
    IF leng IS NULL THEN
        dbms_output.put_line('Decrypt BLOB Len NULL');
    ELSE
        dbms_output.put_line('Decrypt BLOB Len ' || leng);
    END IF;

    -- Read decrypblob to a raw
    decrawlen := 999;

    DBMS_LOB.OPEN (decrypblob, DBMS_LOB.lob_readwrite);
    DBMS_LOB.READ (decrypblob, decrawlen, 1, decrypraw);
    DBMS_LOB.CLOSE (decrypblob);

    dbms_output.put_line('Decrypted data is (256 bit key) ' || decrypraw);
    dbms_output.put_line('---');

    DBMS_LOB.OPEN (srcblob, DBMS_LOB.lob_readwrite);
```

```
    DBMS_LOB.TRIM (srcblob, 0);
    DBMS_LOB.CLOSE (srcblob);

    DBMS_LOB.OPEN (encrypblob, DBMS_LOB.lob_readwrite);
    DBMS_LOB.TRIM (encrypblob, 0);
    DBMS_LOB.CLOSE (encrypblob);

    DBMS_LOB.OPEN (decrypblob, DBMS_LOB.lob_readwrite);
    DBMS_LOB.TRIM (decrypblob, 0);
    DBMS_LOB.CLOSE (decrypblob);
end;
```