# BÁO CÁO TÌM HIỂU MÃ HÓA



**NHÓM 22**

**Hoàng Thị Bích Vân - 141289**

**Võ Minh Duy - 1412101**

12/04/2019

An Toàn Và Bảo Mật Trong Hệ Thống Thông Tin

# Thông Tin Nhóm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | MSSV | Họ và Tên | SĐT | Email |
| 1 | 1412689 | Hoàng Thị Bích vân | 0388568780 | hoangthibichvan95@gmail.com |
|  |  |  |  |  |

# Bảng phân công

|  |  |  |  |
| --- | --- | --- | --- |
| Công việc | Người làm | Hoàn Thành | Giải thích |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Lịch Sử

Nhiều người cho rằng kỷ nguyên của mã hóa hiện đại được bắt đầu bởi Claude Shannon. Ông được xem là cha đẻ của mã hóa. Năm 1949 ông công bố bài báo Lý thuyết về truyền thông trong các hệ thống bảo mật (*Communication Theory of Secrecy Systems*) trên tập san Bell System Technical Journal - Tập san kỹ thuật của hệ thống Bell - và một thời gian ngắn sau đó, trong cuốn Mathematical Theory of Communication - Lý thuyết toán học trong truyền thông - cùng với tác giả Warren Weaver.

# Ưu Điểm

* Mã hóa cơ sở dữ liệu có thể che giấu dữ liệu khỏi những kẻ xâm nhập, thậm chí cả DBA nếu họ không được phép truy cập dữ liệu.
* Mã hóa cơ sở dữ liệu là phương pháp bảo vệ dữ liệu rất hiệu quả đối với những tấn công mức lưu trữ. Những kẻ tấn công có được dữ liệu nhưng không thể hiểu được dữ liệu.

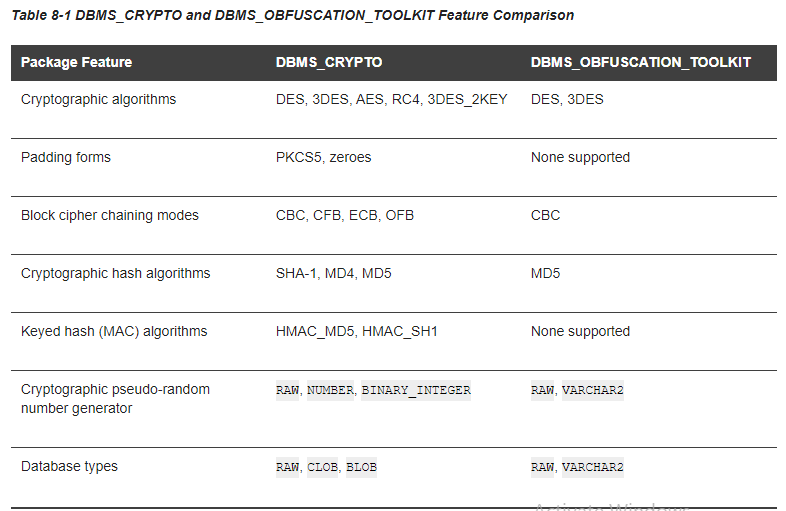
# Khuyết Điểm

* Mã hóa làm tăng số lượng xử lý khi truy xuất dữ liệu và không gian lưu trữ dữ liệu (tùy thuật toán mã hóa) Æ làm chậm hệ thống.
* Dữ liệu vô dụng trong trạng thái mã hóa. Mã hóa không chỉ che đậy dữ liệu khỏi người dùng mà còn khỏi cơ sở dữ liệu. Cơ sở dữ liệu không thể thực thi cácphương thức truy cập dữ liệu cơ bản, so sánh các giá trị dữ liệu trong trạng thái mã hóa một cách hiệu quả, và các phép tính trên giá trị mã hóa không thể thực hiện được.
* Mã hóa đòi hỏi phải có chính sách quản lý khóa thích hợp. Chính sách quản lý khóa có nhiệm vụ quản lý, lưu trữ, bảo vệ, luân phiên và sao lưu khóa mã hóa, đảm bảo rằng chỉ những người có quyền mới thấy được dữ liệu được mã hóa.
  + Nếu khóa bị lộ, dữ liệu sẽ bị giải mã bởi những đối tượng không được phép. Việc mã hóa dữ liệu trở nên vô nghĩa.
  + Nếu khóa mã hóa hóa bị mất, dữ liệu sẽ không thể được giải mã. Đặc biệt trong trường hợp khóa có sử thông tin bí mật của người dùng (ví dụ như mật khẩu), khi người dùng làm mất khóa thì dữ liệu sẽ không bao giờ được giải mã nếu không có thêm cơ chế bảo vệ khóa nào khác.

# Các nguyên tắc khi mã hóa dữ liệu:

1. Mã hóa không được sử dụng để điều khiển truy cập của người dùng.
2. Mã hóa không ngăn chặn được người quản trị có ý đồ xấu.
3. Mã hóa mọi thứ không làm cho dữ liệu an toàn hơn.

# Công cụ mã hóa của Oracle Database:



Oracle Database có 2 gói ứng dụng mã hóa:

* DBMS\_OBFUSCATION\_TOOLKIT: có từ thời khai quốc Oracle đến bản Oracle9i.
* DBMS\_CRYPTO: có từ bản 10g, hỗ trợ thêm nhiều thuật toán mã hóa và kiểu dữ liệu LOBS.

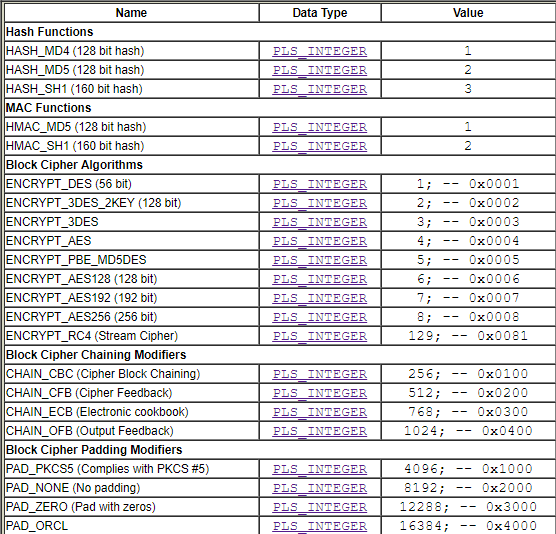
# Quyền sử dụng DBMS\_CRYPTO package

DBMS\_CRYPTO package thuộc sử dụng của SYS schema. Nên người dùng thông thường cần được cấp quyền EXECUTE

GRANT EXECUTE ON SYS.DBMS\_CRYPTO TO [USER | ROLE]

# Cách sử dụng DBMS\_CRYPTO package

* + - 1. **Các parameter được sử dụng trong chức năng mã hóa:**
* **Source** (src) : dữ liệu đầu vào cho mã hóa hoặc giải mã, thuộc một trong các dạng dữ liệu:
  + RAW: dữ liệu hay chuỗi nhị phân
  + BLOB (Binary Large Object): chứa dữ liệu nhị phân không cấu trúc (tập tin hình ảnh, âm thanh, video, documents, …), tối đa 128 terabytes
  + CLOB (Character Large Object): chưa dữ liệu dạng chuỗi kí tự, tối đa 128 terebytes.
* **Type** (typ): tổ hợp các thuật toán mã hóa, liên kết khối và padding sử dụng cho mã hóa hoặc giải mã.

[](http://psoug.org/reference/dbms_crypto.html)

* **Key**: khóa dùng để giải mã hoặc giải mã.
* **Initialization Vector** (iv) (không bắt buộc): một giá trị nhị phân ngẫu nhiên (muối) được sử dụng để mã hóa/giải mã khối dữ liệu đầu tiên trong chuỗi khối. IV và Key cần đi chung một cặp nếu sử dụng.
* **Destination Location** (dst): vị trí xuất dữ liệu mã hóa khi mã hóa dữ liệu dạng LOB.
  + - 1. **Chuyển đổi dữ liệu dạng RAW:**
* **UTL\_I18N.STRING\_TO\_RAW:**
* **UTL\_RAW.CAST\_TO\_RAW:**
* **UTL\_RAW.CAST\_FROM\_BINARY\_DOUBLE:**
* **UTL\_RAW.CAST\_FROM\_BINARY\_FLOAT:**
* **UTL\_RAW.CAST\_FROM\_BINARY\_INTEGER:**
* **UTL\_RAW.CAST\_FROM\_NUMBER:**
* **UTL\_I18N.RAW\_TO\_CHAR:**
* **UTL\_I18N.RAW\_TO\_NCHAR:**
* **UTL\_RAW.CAST\_TO\_BINARY\_DOUBLE:**
* **UTL\_RAW.CAST\_TO\_BINARY\_FLOAT:**
* **UTL\_RAW.CAST\_TO\_BINARY\_INTEGER:**
* **UTL\_RAW.CAST\_TO\_NUMBER:**
* **UTL\_RAW.CAST\_TO\_VARCHAR2:**
* **UTL\_RAW.CAST\_TO\_NVARCHAR2:**
  + - 1. **Sử dụng chức năng mã hóa của DBMS\_CRYPTO:**

Các chức năng trong gói DBMS\_CRYPTO bao gồm:

* **DBMS\_CRYPTO.ENCRYPT**: Sử dụng để mã hóa *plain text* thành *cypher text*

dbms\_crypto.encrypt(

(dst IN OUT BLOB),

src IN RAW (LOB),

typ IN PLS\_INTERGER,

key IN RAW,

iv IN RAW DEFAULT NULL)

RETURN RAW;

* **DBMS\_CRYPTO.DECRYPT**: Sử dụng để giải mã *cypher text*  thành *plain text*

dbms\_crypto.decrypt(

(dst IN OUT BLOB),

src IN RAW (LOB),

typ IN PLS\_INTERGER,

key IN RAW,

iv IN RAW DEFAULT NULL)

RETURN RAW;

* **DBMS\_CRYPTO.HASH**: Chức năng băm dữ liệu

dbms\_crypto.hash (

src IN RAW,

typ IN PLS\_INTERGER)

RETURN RAW;

* **DBMS\_CRYPTO.MAC**: MAC ( Message Authentication Code ) cũng là một chức năng băm dữ liệu nhưng có bổ sung khóa. MAC được sử dụng để kiểm tra tính toán vẹn của dữ liệu.

dbms\_crypto.mac (

src IN RAW,

typ IN PLS\_INTEGER,

key IN RAW)

RETURN RAW;

* **DBMS\_CRYPTO.RANDOMBYTES**: Tạo ra một chuỗi dữ liệu dạng RAW ngẫu nhiên

dbms\_crypto.randombytes (

number\_bytes PLS\_INTEGER)

RETURN RAW;

* **DBMS\_CRYPTO.RANDOMINTEGER**: Tạo ra một số tự nhiên ngẫu nhiên trong khoảng của BINARY\_INTEGER

dbms\_crypto.randominteger RETURN BINARY\_INTEGER;

* **DBMS\_CRYPTO.RANDOMNUMBER**: Tạo ra một số tự nhiên ngẫu nhiên trong khoảng [0..2128-1]

dbms\_crypto.randomnumber RETURN NUMBER;

# Một số ứng dụng minh họ

# Ứng dụng demo

Sử dụng ngữ cảnh đồ án cuối kì

Thực hiện mã hóa

/\*Encrypt data procedure

Parameter : IN raw data, raw key

OUT raw encrypted data

Call: encrypt\_data( in\_data => raw\_data,

in\_key => raw\_key

)

\*/

create or replace FUNCTION encrypt\_data

(

in\_data IN RAW,

in\_key IN RAW DEFAULT NULL

)

RETURN RAW

IS

encryption\_type PLS\_INTEGER := dbms\_crypto.encrypt\_AES128

+ dbms\_crypto.chain\_cbc

+ dbms\_crypto.pad\_pkcs5;

m\_key RAW(128);

encrypted\_data RAW(2000);

BEGIN

--Check if user provide a key

m\_key := in\_key;

if m\_key = NULL then

m\_key := dbms\_crypto.randombytes(128);

end if;

encrypted\_data := dbms\_crypto.encrypt

( src => in\_data,

typ => encryption\_type,

key => m\_key

);

return encrypted\_data;

END;

--Trigger fire when UPDATE or INSERT into table PHIEUBAU

create or replace TRIGGER alter\_phieubau\_trigger

BEFORE UPDATE OR INSERT ON PHIEUBAU

FOR EACH ROW

declare

encrypted\_data RAW(2000);

BEGIN

encrypted\_data := encrypt\_data(in\_data => UTL\_I18N.STRING\_TO\_RAW(:new.UCV1,'AL32UTF8'));

:new.UCV1 := UTL\_RAW.CAST\_TO\_VARCHAR2(encrypted\_data);

encrypted\_data := encrypt\_data(in\_data => UTL\_I18N.STRING\_TO\_RAW(:new.UCV2,'AL32UTF8'));

:new.UCV2 := UTL\_RAW.CAST\_TO\_VARCHAR2(encrypted\_data);

encrypted\_data := encrypt\_data(in\_data => UTL\_I18N.STRING\_TO\_RAW(:new.UCV3,'AL32UTF8'));

:new.UCV3 := UTL\_RAW.CAST\_TO\_VARCHAR2(encrypted\_data);

END;

# Demo

1. input: plaintext - mã hóa
2. output: plaintext - giải mã