

## Vấn đề bảo mật trong Java

### 1. Các vấn đề cần giải quyết:

- Như chúng ta đã biết vấn đề bảo mật là một vấn đề quan trọng trong bất cứ một ứng dụng nào. Để viết một ứng dụng an toàn phải phụ thuộc vào mục đích của ứng dụng và môi trường mà ứng dụng này thực thi. để giải quyết được ta cần phải hiểu:

- Đòi hỏi xác thực: thông thường được xác thực dựa trên password hoặc một số thứ như là thẻ hoặc như khóa. Có nghĩa là xác thực xem client có thể đăng nhập dưới dạng mặc danh nào.
- Điều khiển truy nhập: xác nhận ai đang tham gia trong phiên làm việc để cấp quyền cho họ một cách hợp lý.
- Tính toàn vẹn dữ liệu: phải đảm bảo dữ liệu không bị thay đổi giữa những gì đã gửi và những gì đã nhận. điều này đặc biệt quan trọng nhất là khi thực thi trên môi trường mạng.
- Tính chắc chắn: nếu có trục trặc trong quá trình gửi dữ liệu và bạn không muốn dữ liệu này bị đánh cắp thì dữ liệu này cần phải được mã hóa.
- Không bị bác bỏ: mục đích của dịch vụ này là để chúng nhận một giao tác đặc biệt nào đó đã hoàn tất. dịch vụ không bác bỏ có trách nhiệm báo tin lại cho đối tượng đã gửi tin.

→ để có được ứng dụng đáp ứng được các đòi hỏi trên thì java có các bộ công cụ là:

Mã hóa với khóa đối xứng(Symmetric key encryption).

Mã hóa với khóa công khai(Public key encryption).

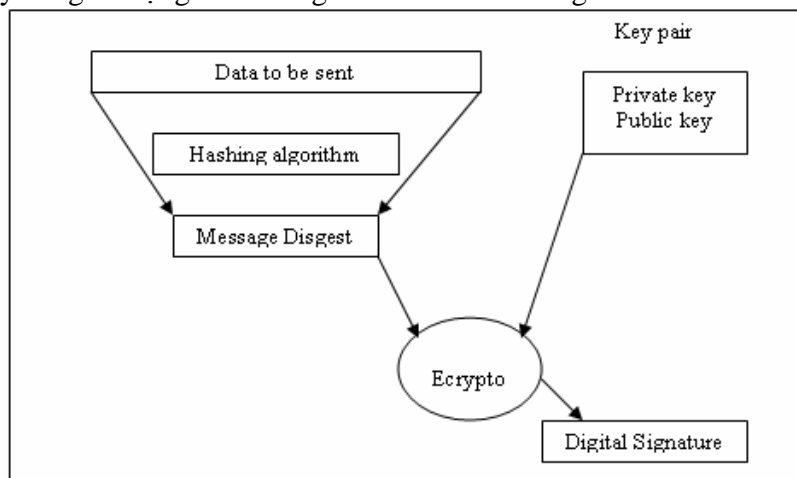
Phương pháp chữ ký số(Hashing/Digital signatures).

- Khóa đối xứng:

Đối với khóa đối xứng một thông điệp chỉ được đọc khi người nhận có khóa mã hóa mà bên gửi sẽ gửi cho bên nhận. câu trả lời là sử dụng khóa công khai hoặc không đối xứng để mã hóa. Với cách này ta có một cặp khóa:khóa bí mật/khóa công khai. Như vậy người gửi sẽ sử dụng khóa công khai để mã hóa dữ liệu và khóa bí mật sẽ được gửi trở lại. người nhận sẽ sử dụng khóa bí mật này để giải mã dữ liệu đã nhận được.

- Phương pháp chữ ký số:

ở đây cũng sử dụng khóa công khai như trên nhưng theo cách khác.



- Người gửi sẽ cho dữ liệu qua một hàm băm để phát sinh một mã khóa gọi là mã băm, tiếp tục mã hóa nó với khóa bí mật rồi gửi mã băm đã mã hóa đi cùng với khóa công khai. Bên phía người nhận sẽ sử dụng khóa công khai nhận được để giải mã và tìm được mã băm, và tiếp tục tìm mã băm trên phần dữ liệu đã nhận được. Nếu mã băm vừa tìm được là giống nhau thì người nhận biết chắc rằng dữ liệu gửi đi đã đáp ứng được các vấn đề giải quyết ở trên.

## 2. Java Cryptography Extension (JCE)

- JCE được cung cấp như là một phần mở rộng của java.JCE 1.2, nó cung cấp một môi trường làm việc và thực thi việc mã hóa, phát sinh khóa và chấp nhận khóa để bổ sung thêm giao tiếp và triển khai số hóa thông điệp, chữ ký.

JCE cung cấp phương thức mã hóa đối xứng thông qua việc sử dụng khóa bí mật, một khóa được chia sẻ bởi người gửi và người nhận để mã hóa cũng như để giải mã dữ liệu.

JCE có gói chính là javax.crypto và nó bao gồm hai gói phụ javax.crypto.spec và javax.crypto.interfaces. đây là hai gói chính mà chương trình hay sử dụng trong việc chấp nhận khóa và chứng thực thông điệp.

Java cung cấp nhiều giải thuật mã hóa dữ liệu như: DES,3DES,AES,RSA...và được chia thành 3 loại mã hóa.

→Mã hóa 1 chiều gồm các giải thuật SHA(Secure Hash Algorithm), MD5(Message Digest)... thường được gọi là hashing hay one-way encryption. MD5 ở đây theo đúng nghĩa của nó thì không phải là mã hóa nhưng trong java cũng có MD5 nên cũng có thể gọi nó là thuật toán mã hóa. người ta thường dùng cái này để mã hóa Password. Đặc điểm của thuật toán này là nó không thể giải mã được.

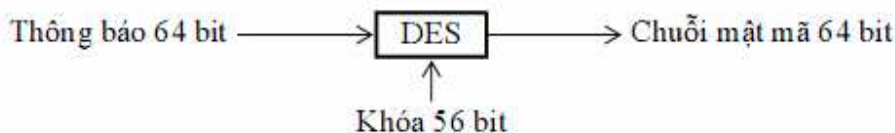
→Mã hóa đối xứng(symmetric): gồm các giải thuật DES(Data Ecrypto Standard), TripleDes(Triple Data Ecrypto Standard)... ở đây có nghĩa là cả hai bên nhận và bên gửi phải sử dụng chung một mã giống nhau. Trong java gọi là SecretKey.

→Mã hóa không đối xứng(asymmetric): gồm các giải thuật RSA... ở đây ta mã hóa bằng một khóa chung (public key) và có thể giải mã bằng khóa private key của người nhận. cho nên nếu có người thứ ba biết public key cũng không thể giải mã được dữ liệu mà anh ta đã nhận được.

### Giải thuật DES:

- DES được dùng như một phương thức mã hóa dữ liệu dùng khóa riêng (private-key). Và có rất nhiều khóa để dùng, mỗi Message sẽ có một khóa mới được chọn ngẫu nhiên. Đối với thuật toán này thì khóa riêng (private key) của đầu gửi(sender) và đầu nhận(reciver) phải giống nhau.

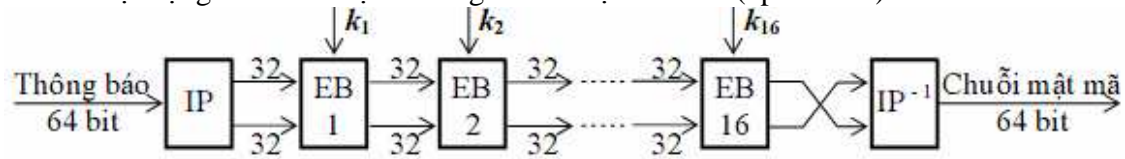
- Giải thuật DES áp dụng khóa 56 bit cho mỗi block dữ liệu 64-bit. Tức là thông báo được “bẻ” ra thành từng khối 64 bit và mỗi khối đại diện cho một ký tự.



**Hình 15.1** Mô hình DES

Khóa k được chọn trong chuỗi 56 bit và từ đó trích ra 16 khóa con  $k_1, k_2, \dots, k_{16}$ . Mỗi khóa con  $k_i$  là một chuỗi 48 bit của khóa k 56 bit. Mỗi khóa con này được sử dụng trong

một khối mã hóa để biến đổi thành 64 bit đầu vào thành 64 bit đầu ra. Quá trình mã hóa có thể hoạt động ở vài chế độ và bao gồm 16 lượt thao tác (operations).



**Hình 15.2** Hình ảnh 16 vòng mã hóa của DES

ở hình trên DES có sử dụng thuật toán hoán vị IP (Initial Permutation) ở đầu và hoán vị ngược  $IP^{-1}$  ở cuối.

- Mặc dù quá trình này đã là rất phức tạp nhưng người ta vẫn có thể dùng thuật toán 3DES, thuật toán này được phát triển từ thuật toán DES, nó có tính bảo mật cao hơn DES rất nhiều chính vì vậy độ phức tạp của nó gấp 3 lần thuật toán DES. Hiện nay thuật toán DES đã có những phương thức để crack nhưng đối với thuật toán 3DES thì khả năng đưa ra các phương thức crack là rất khó. 3DES dùng khóa có chiều dài 168 bit.

- Trong tương lai, DES sẽ không được xem là chuẩn nữa. Cisco có kế hoạch hỗ trợ thuật toán AES (Advance Encryption Standard) vào cuối năm 2001.

### Giải thuật MD5:

- MD5 là một hàm băm để mã hóa với giá trị băm là 128 bit. MD5 được sử dụng rộng rãi trong thế giới phần mềm để đảm bảo rằng tập tin tải về không bị hỏng. người sử dụng có thể so sánh giữa thông số kiểm tra phần mềm bằng MD5 được công bố với thông số kiểm tra phần mềm tải về bằng MD5.

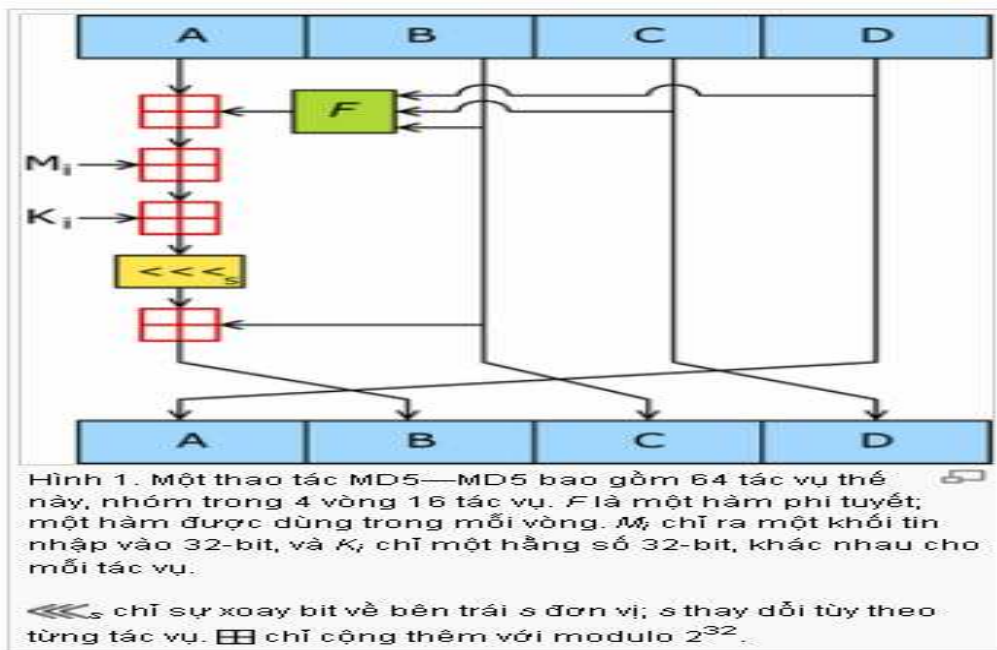
- MD5 thường được sử dụng để mã hóa mật khẩu. và mục đích của việc mã hóa là biến đổi một chuỗi mật khẩu thành một chuỗi khác sao cho từ đoạn mã đó không thể nào lần ra mật khẩu. có nghĩa là không thể giải mã hoặc có thể giải mã nhưng mất rất nhiều thời gian vô tận. sau đây là giải thuật:

- MD5 biến đổi một thông điệp có chiều dài bất kì thành một khối có kích thước cố định 128 bits. Thông điệp đưa vào sẽ được cắt thành các khối 512 bits. Thông điệp được đưa vào bộ đệm để chiều dài của nó sẽ chia hết cho 512. Bộ đệm hoạt động như sau:

- Trước tiên nó sẽ chèn bit 1 vào cuối thông điệp.
- Tiếp đó là hàng loạt bit Zero cho tới khi chiều dài của nó nhỏ hơn bội số của 512 một khoảng 64 bit.
- Phần còn lại sẽ được lấp đầy bởi một số nguyên 64 bit biểu diễn chiều dài ban đầu của thông điệp.

- Thuật toán chính của MD5 hoạt động trên một bộ 128 bit. Chia nhỏ nó ra thành 4 từ 32 bit, kí hiệu là A, B, C và D. Các giá trị này là các hằng số cố định.

Sau đó thuật toán chính sẽ luân phiên hoạt động trên các khối 512 bit. Mỗi khối sẽ phối hợp với một bộ. Quá trình xử lý một khối thông điệp bao gồm 4 bước tương tự nhau, gọi là vòng ("round"). Mỗi vòng lại gồm 16 quá trình tương tự nhau dựa trên hàm một chiều F, phép cộng module và phép xoay trái...



- Hàm băm MD5 (còn được gọi là hàm tóm tắt thông điệp - message digests) sẽ trả về một chuỗi số thập lục phân gồm 32 số liên tiếp. Dưới đây là các ví dụ mô tả các kết quả thu được sau khi băm.

MD5("MD5") = 7f138a09169b250e9dcb378140907378

Thậm chí chỉ cần một thay đổi nhỏ cũng làm thay đổi hoàn toàn kết quả trả về:

MD5("MD5 ") = 9f5fb83531d4ceeb734e06cb6a529972

Ngay cả một chuỗi rỗng cũng cho ra một kết quả phức tạp:

MD5("") = d41d8cd98f00b204e9800998ecf8427e

### 3. Các lớp Engine:

- JCE định nghĩa các lớp Engine như sau:

JCE engine class	Chức năng
Javax.crypto.Cipher	Cung cấp các hàm để viết thành mật mã cho việc mã hóa và giải mã
Javax.crypto.KeyAgreement	Cung cấp các hàm cho giao thức chuyển đổi mã.
Javax.crypto.KeyGenertor	Cung cấp các hàm cho việc tạo khóa đối xứng.
Javax.crypto.Mac	Cung cấp các hàm cho thuật toán MAC
Javax.crypto.SecretkeyFactory	Tạo các khóa bí mật
....	

#### 3.1 Lớp Cipher:

- Lớp javax.crypto.Cipher là một lớp căn bản, lớp này thường xuyên triển khai việc tính toán mã số được sử dụng cho việc mã hóa và giải mã.

- các phương thức cơ bản của lớp Cipher:

getInstance()

- Lớp này có được khi ta gọi phương thức getInstance(). Phương thức này nhận đối số là một chuỗi đại diện cho một transformation. Một transformation là một chuỗi mô tả phép toán(hoặc một phép toán). Một transformation luôn bao gồm tên của thuật toán mã hóa, theo sau đó có thể là một chế độ phản hồi và khung đệm.

- Một đối tượng Cipher nhận được từ phương thức getInstance() phải được thiết lập cho chế độ là mã hóa/ giải mã. Chế độ này được định nghĩa là một hằng số nguyên trong lớp Cipher và có thể tham chiếu tới hai chế độ này theo tên là : ENCRYPT\_MODE hoặc DECRYPT\_MODE.

-Giải thuật được thực hiện dựa trên các khối dữ liệu có kích thước được định sẵn.

Ví dụ để tạo đối tượng Cipher:

```
Cipher cip=Cipher.getInstance("DES"); //DES: Data Ecrypto Stantard
```

Đối tượng được khởi tạo bằng cách gọi phương thức init() như sau:

```
Cip.init(Cipher.ENCRYPT_MODE,sk);
```

Dữ liệu có thể mã hóa hoặc giải mã phụ thuộc vào việc gọi phương thức doFinal() hay update(). Và việc thực hiện nhiều bước sẽ có ích nếu như ta không biết độ dài thực tế câu chuỗi dữ liệu hoặc là dữ liệu lớn có thể lưu trữ trong bộ nhớ cùng một lúc.

### 3.2 Lớp KeyGenertor:

Lớp javax.crypto.Keypair.Genertor dùng để phát sinh cặp khóa public hay private. ở đây JCE cung cấp đối tượng KeyGenertor dùng để phát sinh khóa bí mật cho giải thuật đối xứng. đối tượng được tạo ra khi gọi phương thức getInstance().

Ví dụ:

```
KeyGenertor kg= KeyGenertor. getInstance("DES/ECB/SKCS5Padding");
```

Giải thuật mã hóa DES ở chế độ ECB có SKCS5Padding.

Phương thức getInstance() sẽ lấy tham số là tên của thuật toán mã hóa đối xứng mà khóa bí mật sẽ được tạo ra. Nếu ta chỉ định tên của thuật toán, hệ thống sẽ tìm và chọn bộ khóa đầu tiên có giá trị. Còn nếu ta cung cấp cả tên thuật toán và provider thì hệ thống sẽ tìm bộ tạo khóa theo yêu cầu và sẽ chỉ ra ngoại lệ nếu không tìm thấy.

### 3.3 SecretKey:

Được dùng để tạo một SecretKey từ một mảng byte mà không cần dựa trên SecretKeyFactory(SecretKeyFactory dùng để tạo khóa bí mật, thường được sử dụng trong trường hợp khóa đối xứng). Lớp này có ích khi các khóa bí mật có thể biểu diễn thành một mảng byte và không có tham số khóa như trường hợp khóa DES hoặc Triple DES.

Ví dụ:

```
KeyGenertor kg= KeyGenertor. getInstance("DES");
```

```
SecretKey sk=kg.generate();
```

### 3.4 Lớp keyAgrment:

- lớp này cung cấp chức năng của giao thức chấp nhận khóa. Các khóa liên quan trong việc thiết lập một khóa có thể chia sẻ được tạo bởi một trong số các bộ tạo khóa(KeyPairGenertor hay KeyGenertor). Trước tiên phải khởi tạo bằng phương thức getInstance() với đối số là tên của giải thuật sinh khóa.

### 3.5 Lớp MessageDigest:

- Lớp này dùng để phát sinh chữ ký và phát sinh thông điệp số. Sẽ tạo ra một đối tượng với chuỗi MD5 là tham số cho phương thức getInstance().

Ví dụ: MessageDigest md=MessageDigest.Instance("MD5");



```

// TODO code application logic here
try
{
    //hàm mật mã cho việc mã hóa dữ liệu
    Cipher Cip=Cipher.getInstance("DES");
    //hàm tạo khóa đối xứng
    KeyGenerator KG=KeyGenerator.getInstance("DES");
    // hàm tạo khóa bí mật
    SecretKey SK=KG.generateKey();
    Cip.init(Cipher.ENCRYPT_MODE, SK);

    FileInputStream fis=new FileInputStream("data.txt");
    BufferedInputStream bis=new BufferedInputStream(fis);
        int len=bis.available();
        byte []buff=new byte[len];
        byte []enc=new byte[len];
        while(bis.available()!=0)
        {
            len=bis.read(buff);
            int bytecount=Cip.update(buff,0,len,enc);
        }
        bis.close();
        fis.close();
    Cip.doFinal();

    // ghi dữ liệu sau khi đã mã hóa ra file output.txt
    FileOutputStream fos=new FileOutputStream("output.txt");
        fos.write(enc);
    fos.close();
    // ghi khóa ra file key.txt
    FileOutputStream fosk= new FileOutputStream("key.txt");
        fosk.write(SK.getEncoded());
        String s1;
        s1=SK.getFormat();
    fosk.close();
}
catch(Exception ex)
{
    System.out.print("Loi khong ma hoa duoc du lieu");
}
}
}

```

→ Đây là chương trình giải mã với dữ liệu nằm trong file output.txt đã được mã hóa như trên:

```

import java.security.*;
import java.io.*;
import javax.crypto.*;
import javax.crypto.spec.*;

public class GiaiMa{
    public static void main(String[] args) {
        // TODO code application logic here
        try
        {
            // đọc khóa bí mật được mã hóa trong file key.txt
            FileInputStream fisk=new FileInputStream("key.txt");
            BufferedInputStream bisk=new BufferedInputStream(fisk);
            int len1=bisk.available();
            byte []enkey=new byte[len1];
            fisk.read(enkey);
            fisk.close();

            // giải mã khóa bí mật
            SecretKeyFactory skf=SecretKeyFactory.getInstance("DES");
            DESKeySpec dks=new DESKeySpec(enkey);
            SecretKey sk=skf.generateSecret(dks);

            // tạo Cipher để gọi hàm giải mã
            Cipher Cip=Cipher.getInstance("DES");
            Cip.init(Cipher.DECRYPT_MODE,sk);

            // giải mã dữ liệu nằm trong file output.txt
            FileInputStream fis=new FileInputStream("output.txt");
            BufferedInputStream bis=new BufferedInputStream(fis);
            int len=bis.available();
            byte []enc=new byte[len];
            byte []buff=new byte[len];

            while(bis.available() != 0)
            {
                len=bis.read(buff);
                int bytecount=Cip.update(buff,0,len,enc);
            }
            bis.close();
            fis.close();
            //Cip.init(Cipher.DECRYPT_MODE,sk);
            //Cip.doFinal();

            // ghi dữ liệu sau khi mã hóa ra file output2.txt
            FileOutputStream fos=new FileOutputStream("output2.txt");

```

```

        fos.write(enc);
        fos.close();
    }
    catch(Exception e)
    {
        System.out.print("Loi khong giai ma duoc du lieu");
    }
}
}

```

Đây là chương trình mã hóa password sử dụng giải thuật MD5:

```

import java.io.*;
import javax.crypto.*;
import java.security.*;
import java.math.BigInteger;

public class MD5{
    public static void main(String[] args) {
        String text=null;
        //text=args[0];
        if (args.length==0)
            text="MD5";
        else
            text=args[0];

        try
        {
            MessageDigest md5 = MessageDigest.getInstance("MD5");
            md5.update(text.getBytes());
            BigInteger dis = new BigInteger(1, md5.digest());
            //byte []dis=md5.digest();
            text = dis.toString(16);
            System.out.println("Chuoi sau khi duoc ma hoa bang giai thuat MD5:");
            System.out.print(text);
        }
        catch(NoSuchAlgorithmException e)
        {
            System.out.print("Loi khong ma hoa duoc bang giai thuat MD5");
        }
    }
}

```