

PHẦN IV. LẬP TRÌNH MẠNG AN TOÀN BẢO MẬT

CHƯƠNG VII

LẬP TRÌNH MẠNG AN TOÀN BẢO MẬT VỚI SSL

I. GIỚI THIỆU SSL VÀ MỘT SỐ KHÁI NIỆM

1. Giới thiệu SSL

SSL (Secure Socket Layer) là giao thức đa mục đích được thiết kế để tạo ra các giao tiếp giữa hai chương trình ứng dụng trên một cổng định trước (socket 443) nhằm mã hoá toàn bộ thông tin đi/đến, được sử dụng trong giao dịch điện tử như truyền số liệu thẻ tín dụng, mật khẩu, số bí mật cá nhân (PIN) trên Internet.

Trong các giao dịch điện tử trên mạng và trong các giao dịch thanh toán trực tuyến, thông tin/dữ liệu trên môi trường mạng Internet không an toàn thường được bảo đảm bởi cơ chế bảo mật thực hiện trên tầng vận tải có tên Lớp công bảo mật SSL (Secure Socket Layer) - một giải pháp kỹ thuật hiện nay được sử dụng khá phổ biến trong các hệ điều hành mạng máy tính trên Internet. Giao thức SSL được hình thành và phát triển đầu tiên năm 1994 bởi nhóm nghiên cứu Netscape dẫn dắt bởi Elgammal, và ngày nay đã trở thành chuẩn bảo mật thực hành trên mạng Internet. Phiên bản SSL hiện nay là 3.0 và vẫn đang tiếp tục được bổ sung và hoàn thiện. Tương tự như SSL, một giao thức khác có tên là Công nghệ truyền thông riêng tư PCT (Private Communication Technology) được đề xướng bởi Microsoft, hiện nay cũng được sử dụng rộng rãi trong các mạng máy tính chạy trên hệ điều hành Windows NT. Ngoài ra, một chuẩn của Nhóm đặc trách kỹ thuật Internet IETF (Internet Engineering Task Force) có tên là Bảo mật lớp giao vận TLS (Transport Layer Security) dựa trên SSL cũng được hình thành và xuất bản dưới khuôn khổ nghiên cứu của IETF Internet Draft được tích hợp và hỗ trợ trong sản phẩm của Netscape.

2. Khóa – Key

Định nghĩa khóa

Khóa (key) là một thông tin quan trọng dùng để mã hóa thông tin hoặc giải mã thông tin đã bị mã hóa. Có thể hiểu nôm na khóa giống như là mật khẩu(password).

Độ dài khóa – Key Length

Độ dài khóa được tính theo bit: 128 bits, 1024 bits hay 2048 bits,... Khóa càng dài thì càng khó phá. Chẳng hạn như khóa RSA 1024 bits đồng nghĩa với việc chọn 1 trong 2^{1024} khả năng.

Password và PassParse

Password và passparse gần giống nhau. Password không bao giờ hết hạn(expire). Passparse chỉ có hiệu lực trong một khoảng thời gian nhất định có thể là 5 năm, 10 năm hay chỉ là vài ba ngày. Sau thời gian đó, phải thay đổi lại mật khẩu mới. Nói chung, mọi thứ trong SSL như passparse, khóa, giấy chứng nhận, chữ kí số (sẽ nói sau), ... đều chỉ có thời hạn sử dụng nhất định. Passparse được dùng để mở (mã hóa/giải mã) khóa riêng.

3. Thuật toán mã hóa

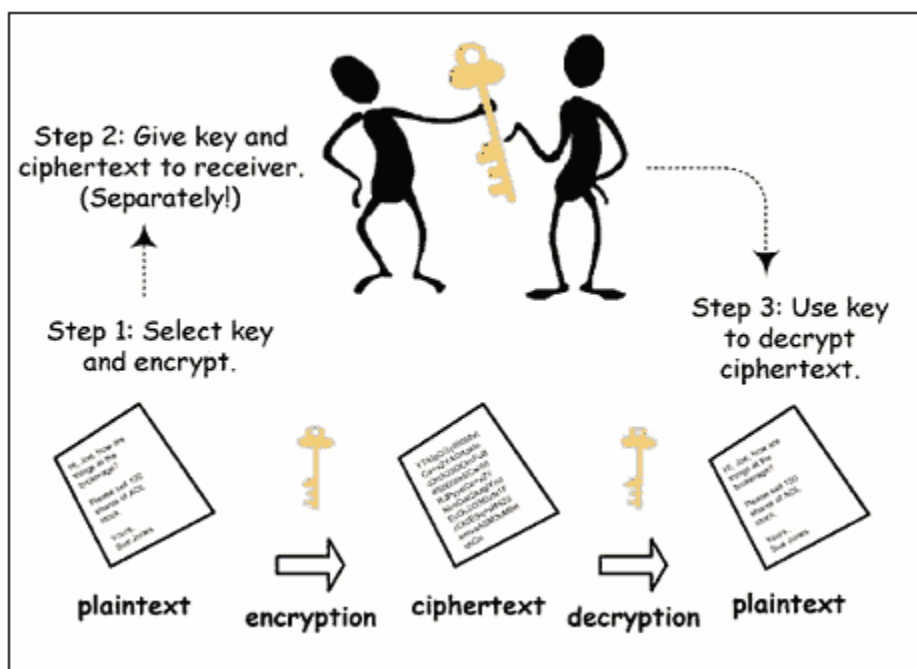
Mã hóa (encrypt) và giải mã (decrypt) thông tin dùng các hàm toán học đặt biệt. Được biết đến với cái tên là thuật toán mã hóa (cryptographic algorithm) và thường được gọi tắt là cipher. Các thuật toán mã hoá và xác thực của SSL được sử dụng bao gồm (phiên bản 3.0):

- (1) DES - Chuẩn mã hoá dữ liệu (ra đời năm 1977), phát minh và sử dụng của chính phủ Mỹ.
- (2) DSA - Thuật toán chữ ký điện tử, chuẩn xác thực điện tử, phát minh và sử dụng của chính phủ Mỹ.
- (3) KEA - Thuật toán trao đổi khoá, phát minh và sử dụng của chính phủ Mỹ.
- (4) MD5 - Thuật toán tạo giá trị "băm" (message digest), phát minh bởi Rivest.
- (5) RC2, RC4 - Mã hoá Rivest, phát triển bởi công ty RSA Data Security.
- (6) RSA - Thuật toán khoá công khai, cho mã hoá và xác thực, phát triển bởi Rivest, Shamir và Adleman.
- (7) RSA key exchange - Thuật toán trao đổi khoá cho SSL dựa trên thuật toán RSA.
- (8) SHA-1 - Thuật toán hàm băm an toàn, phát triển và sử dụng bởi chính phủ Mỹ.
- (9) SKIPJACK - Thuật toán khoá đối xứng phân loại được thực hiện trong phần cứng Fortezza, sử dụng bởi chính phủ Mỹ;
- (10) Triple-DES - Mã hoá DES ba lần.

Các phương pháp mã hóa

Có hai phương pháp mã hóa được sử dụng phổ biến hiện nay là mã hóa bằng khóa đối xứng và mã hóa dùng cặp khóa chung - khóa riêng..

Mã hóa bằng khóa đối xứng (symmetric-key)



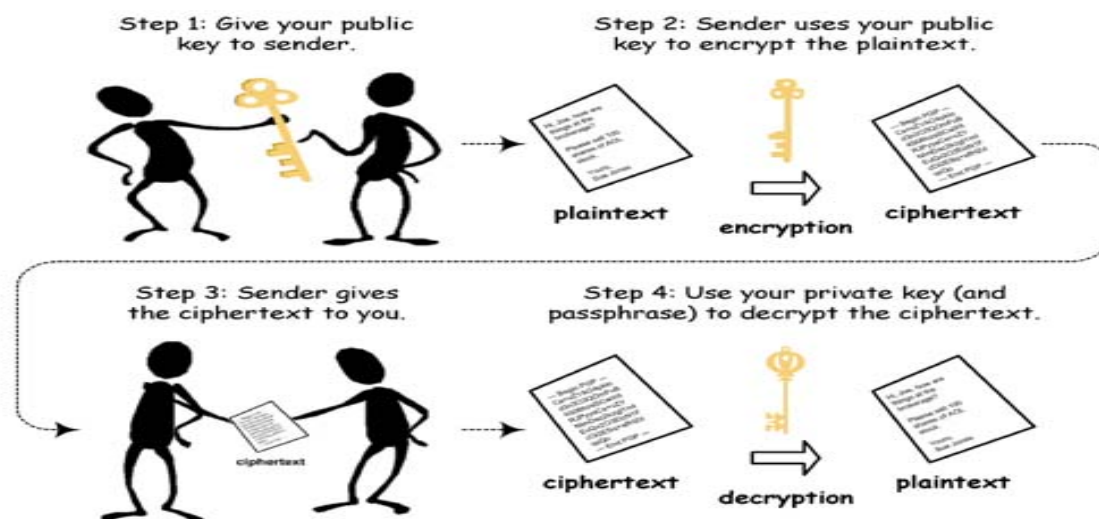
Hình 7.1. Mã hoá bằng khoá đối xứng

Khóa dùng để mã hóa cũng là khóa dùng để giải mã.

Một khe hở trong mã hóa đối xứng là bạn phải chuyển khóa cho người nhận để họ có thể giải mã. Việc chuyển khóa không được mã hóa qua mạng là một điều cực kì mạo hiểm. Nhỡ như khóa này rơi vào tay người khác thế là họ có thể giải mã được thông tin mà đã chuyển đi. Phương pháp mã hóa bằng khóa chung - khóa riêng ra đời nhằm giải quyết vấn đề này.

Thay vì chỉ có một khóa duy nhất dùng chung cho mã hóa và giải mã, sẽ có một cặp khóa gồm khóa chung chỉ dùng để mã hóa và khóa riêng chỉ dùng để giải mã. Khi người A muốn gửi thông điệp cho người B thì người B cần biết khóa chung của người A. (Khóa này được người A công bố công khai). Người B mã hóa các thông tin gửi đến người A bằng khóa chung của người A. Chỉ có người A mới có khóa riêng để giải mã các thông tin này. Nhỡ như thông tin này có rơi vào tay người khác thì họ cũng không thể giải mã được vì chỉ có người A mới có khóa riêng dành cho việc giải mã đúng thông điệp trên.

Mã hóa dùng cặp khóa chung – khóa riêng



Hình 7.2. Mã hoá công khai

4. Cơ chế làm việc của SSL – SSL Protocol

Điểm cơ bản của SSL là được thiết kế độc lập với tầng ứng dụng để đảm bảo tính bí mật, an toàn và chống giả mạo luồng thông tin qua Internet giữa hai ứng dụng bất kỳ, thí dụ như webserver và các trình duyệt (browser), do đó được sử dụng rộng rãi trong nhiều ứng dụng khác nhau trên môi trường Internet. Toàn bộ cơ chế hoạt động và hệ thống thuật toán mã hoá sử dụng trong SSL được phổ biến công khai, trừ khóa chia sẻ tạm thời được sinh ra tại thời điểm trao đổi giữa hai ứng dụng là tạo ngẫu nhiên và bí mật đối với người quan sát trên mạng máy tính. Ngoài ra, giao thức SSL còn đòi hỏi ứng dụng chủ phải được chứng thực bởi một đối tượng lớp thứ ba (CA) thông qua chứng chỉ điện tử (digital certificate) dựa trên mật mã công khai (thí dụ RSA).

Sau đây ta xem xét một cách khái quát cơ chế hoạt động của SSL để phân tích cấp độ an toàn của nó và các khả năng áp dụng trong các ứng dụng nhạy cảm, đặc biệt là các ứng dụng về thương mại và thanh toán điện tử.

Giao thức SSL dựa trên hai nhóm con giao thức là giao thức "bắt tay" (handshake protocol) và giao thức "bản ghi" (record protocol). Giao thức bắt tay xác định các tham số giao dịch giữa hai đối tượng có nhu cầu trao đổi thông tin hoặc dữ liệu, còn giao thức bản ghi xác định khuôn dạng cho tiến hành mã hoá và truyền tin hai chiều giữa hai đối tượng đó. Khi hai ứng dụng máy tính, thí dụ giữa một trình duyệt web và máy chủ web, làm việc với nhau, máy chủ và máy khách sẽ trao đổi "lời chào" (hello) dưới dạng các thông điệp cho nhau với xuất phát đầu tiên chủ động từ máy chủ, đồng thời xác định các chuẩn về thuật toán mã hoá và nén số liệu có thể được áp dụng giữa hai ứng dụng. Ngoài ra, các ứng dụng còn trao đổi "số nhận dạng/khoá theo phiên" (session

ID, session key) duy nhất cho lần làm việc đó. Sau đó ứng dụng khách (trình duyệt) yêu cầu có chứng chỉ điện tử (digital certificate) xác thực của ứng dụng chủ (web server).

Chứng chỉ điện tử thường được xác nhận rộng rãi bởi một cơ quan trung gian (Thẩm quyền xác nhận CA - Certificate Authority) như RSA Data Security hay VeriSign Inc., một dạng tổ chức độc lập, trung lập và có uy tín. Các tổ chức này cung cấp dịch vụ "xác nhận" số nhận dạng của một công ty và phát hành chứng chỉ duy nhất cho công ty đó như là bằng chứng nhận dạng (identity) cho các giao dịch trên mạng, ở đây là các máy chủ webserver.

Sau khi kiểm tra chứng chỉ điện tử của máy chủ (sử dụng thuật toán mật mã công khai, như RSA tại trình máy trạm), ứng dụng máy trạm sử dụng các thông tin trong chứng chỉ điện tử để mã hoá thông điệp gửi lại máy chủ mà chỉ có máy chủ đó có thể giải mã. Trên cơ sở đó, hai ứng dụng trao đổi khoá chính (master key) - khoá bí mật hay khoá đối xứng - để làm cơ sở cho việc mã hoá luồng thông tin/dữ liệu qua lại giữa hai ứng dụng chủ khách. Toàn bộ cấp độ bảo mật và an toàn của thông tin/dữ liệu phụ thuộc vào một số tham số:

- (i) Số nhận dạng theo phiên làm việc ngẫu nhiên.
- (ii) Cấp độ bảo mật của các thuật toán bảo mật áp dụng cho SSL.
- (iii) Độ dài của khoá chính (key length) sử dụng cho lược đồ mã hoá thông tin.

5. Bảo mật của giao thức SSL

Mức độ bảo mật của SSL như trên mô tả phụ thuộc chính vào độ dài khoá hay phụ thuộc vào việc sử dụng phiên bản mã hoá 40 bits và 128bits. Phương pháp mã hoá 40 bits được sử dụng rộng rãi không hạn chế ngoài nước Mỹ và phiên bản mã hoá 128 bits chỉ được sử dụng trong nước Mỹ và Canada. Theo luật pháp Mỹ, các mật mã "mạnh" được phân loại vào nhóm "vũ khí" (weapon) và do đó khi sử dụng ngoài Mỹ (coi như là xuất khẩu vũ khí) phải được phép của chính phủ Mỹ hay phải được cấp giấy phép của Bộ Quốc phòng Mỹ (DoD). Đây là một lợi điểm cho quá trình thực hiện các dịch vụ thương mại và thanh toán điện tử trong Mỹ và các nước đồng minh phương Tây và là điểm bất lợi cho việc sử dụng các sản phẩm cần có cơ chế bảo mật và an toàn trong giao dịch điện tử nói chung và thương mại điện tử nói riêng trong các nước khác.

Các phương thức tấn công (hay bẻ khoá) của các thuật toán bảo mật thường dùng dựa trên phương pháp "tấn công vét cạn" (brute-force attack) bằng cách thử-sai miền không gian các giá trị có thể của khoá. Số phép thử-sai tăng lên khi độ dài khoá tăng và dẫn đến vượt quá khả năng và công suất tính toán, kể cả các siêu máy tính hiện đại nhất. Thí dụ, với độ dài khoá là 40 bits, thì số phép thử sẽ là $2^{40}=1,099,511,627,776$ tổ hợp. Tuy nhiên độ dài khoá lớn kéo theo tốc độ tính toán giảm (theo luật thừa nghịch đảo) và dẫn đến khó có khả năng áp dụng trong thực tiễn. Một khi khoá bị phá, toàn bộ thông tin giao dịch trên mạng sẽ bị kiểm soát toàn bộ. Tuy nhiên do độ dài khoá lớn (thí dụ 128 bits, 256 bits), số phép thử-sai trở nên "không thể thực hiện" vì phải

mất hàng năm hoặc thậm chí hàng nghìn năm với công suất và năng lực tính toán của máy tính mạnh nhất hiện nay.

Ngày từ năm 1995, bản mã hoá 40 bits đã bị phá bởi sử dụng thuật toán vét cạn. Ngoài ra, một số thuật toán bảo mật (như DES 56 bits, RC4, MD4,...) hiện nay cũng bị coi là không an toàn khi áp dụng một số phương pháp và thuật toán tấn công đặc biệt. Đã có một số đề nghị thay đổi trong luật pháp Mỹ nhằm cho phép sử dụng rộng rãi các phần mềm mã hoá sử dụng mã hoá 56 bits song hiện nay vẫn chưa được chấp thuận.

II. LẬP TRÌNH MẠNG AN TOÀN BẢO MẬT VỚI SSL

1. Thư viện java hỗ trợ lập trình SSL

Java Language	Java Language									
Tools & Tool APIs	java	javac	javadoc	apt	jar	javap	JPDA	JConsole	Java VisualVM	
	Security	Int'l	RMI	IDL	Deploy	Monitoring	Troubleshoot	Scripting	JVM TI	
Java Web (Deployment)	Java Web App Development/Distribution						Java Web Start		Applet (Plug-In)	
User Interface Toolkits	AWT				Swing			Java 2D		
	Accessibility		Drag n Drop		Input Methods		Image I/O	Print Service		Sound
Integration Libraries	IDL	JDBC™		JNDI™		RMI	RMI-IIOP		Scripting	
Other Base Libraries	Beans		Intl Support		I/O	JMX	JNI		Math	
	Networking		Override Mechanism		Security	Serialization	Extension Mechanism		XML JAXP	
lang and util Base Libraries	lang and util		Collections	Concurrency Utilities		JAR		Logging	Management	
	Preferences API		Ref Objects	Reflection		Regular Expressions		Versioning	Zip	Instrument
Java Virtual Machine	Java Hotspot™ Client VM					Java Hotspot™ Server VM				
Platforms	Solaris™			Linux		Windows			Other	

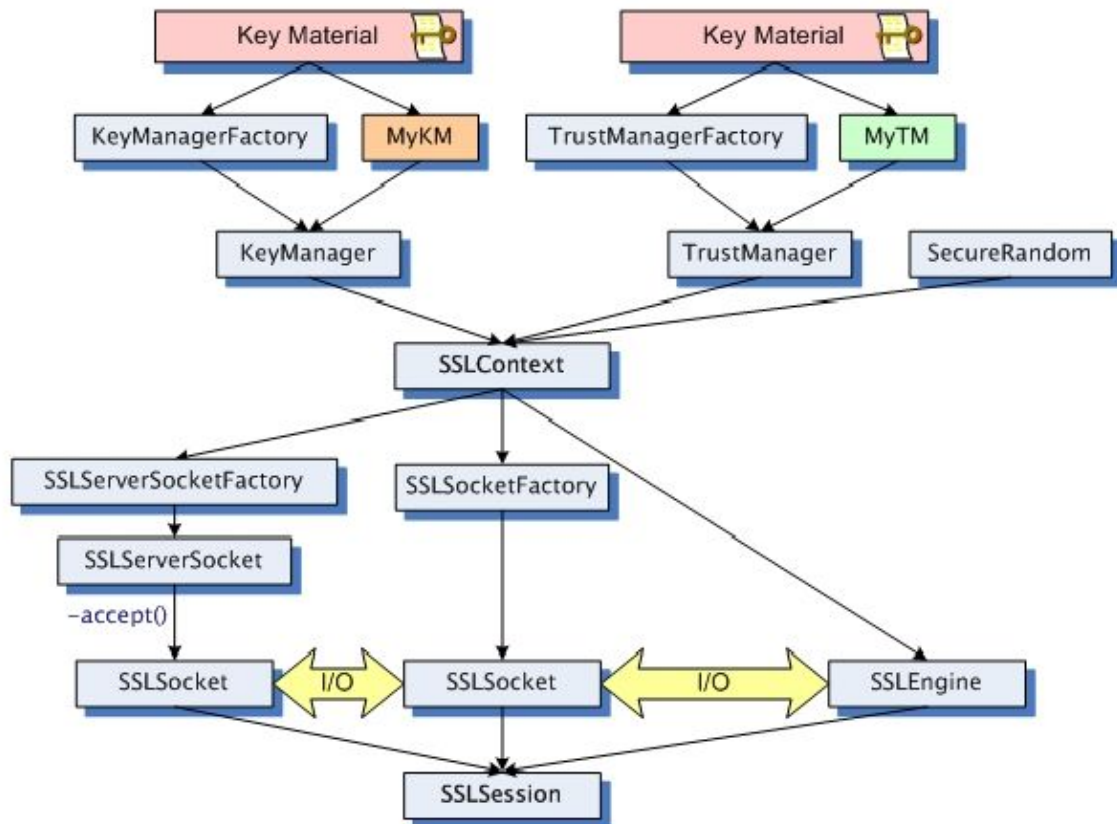
Hình 7.3. Kiến trúc JDK

Java™ security bao gồm tập hợp rất nhiều APIs, công cụ, và cài đặt của các thuật toán bảo mật thông dụng (commonly-used security algorithms), các cơ chế (mechanisms) và các giao thức (protocols). Java security APIs được sử dụng rộng rãi. Bao gồm mã hóa (cryptography), hạ tầng khóa chung (public key infrastructure), trao đổi bảo mật (secure communication), xác thực (authentication), và điều khiển truy cập (access control). Bao gồm rất nhiều lớp thư viện như Java Authentication and Authorization Service (JAAS), Java Cryptography Extension (JCE), Java Secure Socket Extension (JSSE)... Tuy nhiên trong báo cáo này chỉ tập trung vào JSSE. Và cụ thể hơn là JSSE hỗ trợ SSL. Các gói thư viện hỗ trợ lập trình với SSL:

- Gói javax.net.ssl (JSSE)
- Gói javax.rmi.ssl (SSL/TLS-based RMI Socket Factories)

1.1. Lớp SSL

Để truyền thông an toàn, cả 2 phía của kết nối đều phải sử dụng SSL. Trong java, các lớp điểm cuối của kết nối là `SSLSocket` và `SSLEngine`. Hình 7.4. cho thấy các lớp chính được sử dụng để tạo ra `SSLSocket`/`SSLEngines`.



Hình 7.4. Các lớp java SSL

2. Ví dụ về sử dụng các lớp SSL

Chương trình ví dụ có mã lệnh cho phép server và client có thể xác thực nhau. Muốn vậy thì client phải có chứng chỉ của server (thực tế là một tập (chain) chứng chỉ). Trường hợp ví dụ **chứng chỉ của server là chứng chỉ tự ký** (self-certificate). Sau đó khi chạy chương trình thì trở tới nó.

Sau khi đánh lệnh trên thì sẽ hiện ra các thông tin để điền vào như mật khẩu, tên cá nhân, tên tổ chức, thành phố,...

Server source code (EchoServer.java)

```
import javax.net.ssl.SSLServerSocket;

import javax.net.ssl.SSLServerSocketFactory;
```

```

import javax.net.ssl.SSLSocket;

import java.io.BufferedReader;

import java.io.InputStream;

import java.io.InputStreamReader;

public class EchoServer {

    public static void main(String[] arstring) {

        try {

            SSLServerSocketFactory sslserversocketfactory =

                (SSLServerSocketFactory)

SSLServerSocketFactory.getDefault();

            SSLServerSocket sslserversocket =

                (SSLServerSocket)

sslserversocketfactory.createServerSocket(9999);

            SSLSocket sslsocket = (SSLSocket) sslserversocket.accept();

            InputStream inputstream = sslsocket.getInputStream();

            InputStreamReader inputstreamreader = new

InputStreamReader(inputstream);

            BufferedReader bufferedreader = new

BufferedReader(inputstreamreader);

            String string = null;

            while ((string = bufferedreader.readLine()) != null) {

                System.out.println(string);

                System.out.flush();

            }

        } catch (Exception exception) {

            exception.printStackTrace();

        }

    }

}

```



```

Client source code (EchoClient.java)
import javax.net.ssl.SSLSocket;

import javax.net.ssl.SSLSocketFactory;

import java.io.*;

public class EchoClient {

    public static void main(String[] arstring) {

        try {

            SSLSocketFactory sslsocketfactory = (SSLSocketFactory)
SSLSocketFactory.getDefault();

            SSLSocket sslsocket = (SSLSocket)
sslsocketfactory.createSocket("localhost", 9999);

            InputStream inputStream = System.in;

            InputStreamReader inputstreamreader = new
InputStreamReader(inputStream);

            BufferedReader bufferedreader = new
BufferedReader(inputstreamreader);

            OutputStream outputStream = sslsocket.getOutputStream();

            OutputStreamWriter outputstreamwriter = new
OutputStreamWriter(outputStream);

            BufferedWriter bufferedwriter = new
BufferedReader(outputstreamwriter);

            String string = null;

            while ((string = bufferedreader.readLine()) != null) {

                bufferedwriter.write(string + '\n');

                bufferedwriter.flush();

            }

        } catch (Exception exception) {

            exception.printStackTrace();

        }

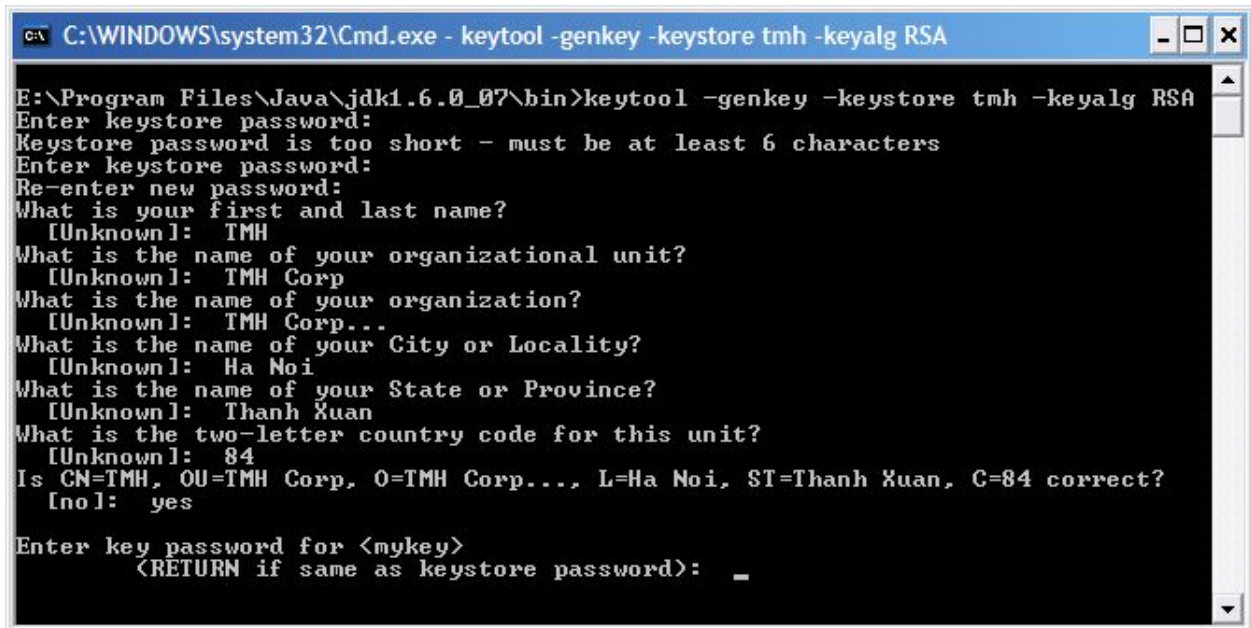
    }

}

```

}

Sau khi dịch chạy chương trình , được kết quả sau:



```
C:\WINDOWS\system32\Cmd.exe - keytool -genkey -keystore tmh -keyalg RSA

E:\Program Files\Java\jdk1.6.0_07\bin>keytool -genkey -keystore tmh -keyalg RSA
Enter keystore password:
Keystore password is too short - must be at least 6 characters
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: TMH
What is the name of your organizational unit?
[Unknown]: TMH Corp
What is the name of your organization?
[Unknown]: TMH Corp...
What is the name of your City or Locality?
[Unknown]: Ha Noi
What is the name of your State or Province?
[Unknown]: Thanh Xuan
What is the two-letter country code for this unit?
[Unknown]: 84
Is CN=TMH, OU=TMH Corp, O=TMH Corp..., L=Ha Noi, ST=Thanh Xuan, C=84 correct?
[no]: yes

Enter key password for <mykey>
(RETURN if same as keystore password): _
```

Giải thích:

- genkey: Lệnh tạo key
- keystore mySrvKeystore: Tên key là mySrvKeystore
- keyalg RSA: Thuật toán dùng để mã hóa là RSA

Về phần ứng dụng qua giao diện dòng lệnh thì sử dụng chương trình mẫu giống như ở trên. Chạy như sau:

Tạo chứng nhận:

```
keytool -genkey -keystore mySrvKeystore -keyalg RSA
```

Mật khẩu sẽ điền là 123456

Sau khi tạo xong chứng chỉ thì copy file key vào trong thư mục chứa file

Phía server thì chứng chỉ được lưu trong keyStore

Chạy chương trình:

```
java -Djavax.net.ssl.keyStore=mySrvKeystore -Djavax.net.ssl.keyStorePassword=123456
EchoServer
```

Phía Client thì chúng chỉ được lưu trong trustStore

Chạy chương trình:

```
java -Djavax.net.ssl.trustStore=mySrvKeystore -Djavax.net.ssl.trustStorePassword=123456  
EchoClient
```

III. KẾT LUẬN

Chương này bước đầu đề cập đến vấn đề lập trình mạng an toàn bảo mật mà chủ yếu với SSL, là giao thức được sử dụng rộng rãi nhất cho việc cài đặt mã hoá trong Web. Với cách tiếp cận này, sinh viên có thể tự nghiên cứu khai thác các kỹ thuật lập trình mạng an toàn bảo mật khác nhau, khai thác các hỗ trợ khác nhau của các môi trường Java, .NET...(bảo mật trong java, phương thức SOCKS, JCA, JCE...).