

Föreläsning 7

DD2390

Internetprogrammering

6 hp

Innehåll

- Krypteringsöversikt (PKI)
- Java Secure Socket Extension (JSSE)

Säkerhetsproblem

1. Vem är det man kommunicerar med
 - Autentisering
2. Data kan avläsas
 - Interception
3. Data kan ändras
 - Dataintegritet

Public Key Infrastructure (PKI)

- Erbjuder ett sätt att lösa dessa problem
 1. Certifikat
 2. Kryptering
 3. Hash

Assymmetrisk kryptering

- Två nycklar, privat / publik
- Löser autentisering
- Beräkningsintensiv
 - olämplig för kryptering av större datamängder
- Används för kryptering av den symmetriska nyckeln
- Algoritmer
 - Rivest Shamir Adleman (RSA)
 - Diffie-Hellman (DH)

Symmetrisk kryptering

- En gemensam nyckel
- Snabbt => för större datamängder
- Vanliga algoritmer
 - Data Encryption Standard (DES)
 - Triple-strength DES (3DES)
 - Advanced Encryption Standard (AES)
 - Rivest Cipher 2 (RC2)
 - Rivest Cipher 4 (RC4).

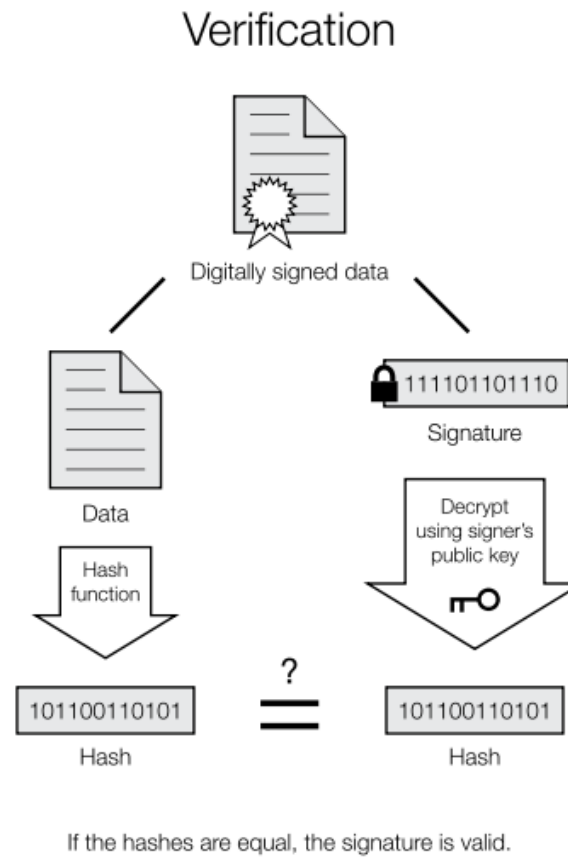
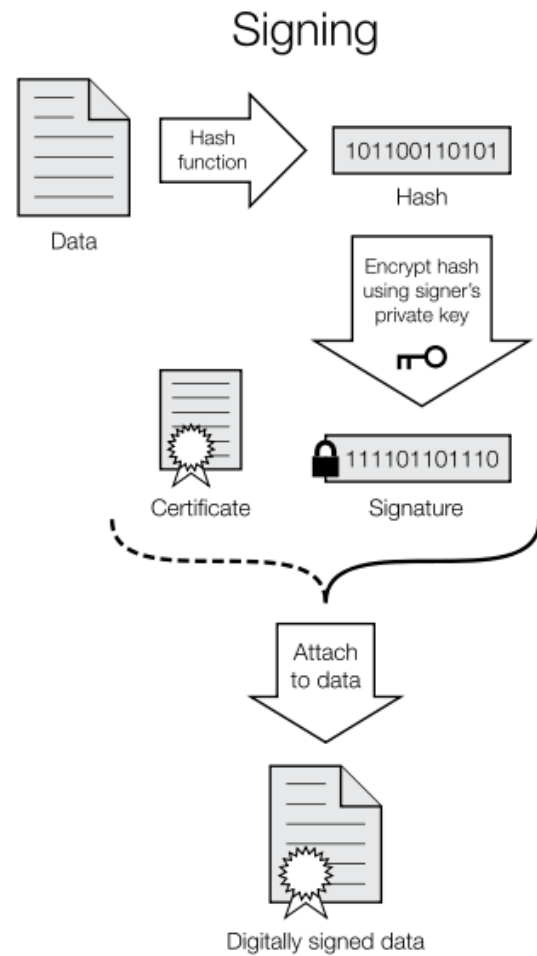
Hash Funktioner

- HMAC (Hash Message Authentication Code)
 - Hash kod adderat till ändelsen av symmetrisk krypterat data
- Alogritmer
 - SHA (Secure Hash Algoritm)
 - MD5 (Message Digest 5)
- Digital signatur
 - Hash krypterat med avsändarens privata nyckel

Certifikat

- Certificate Authority (CA).
 - Symantec är idag störst med c:a 40% av marknaden.
 - Över 100 rotcertifikat finns installerade i de vanligaste webbläsarna
- X.509 är en standard för att specificera ett certifikats format (1988)
- Innehåller bl a
 - Issuer
 - Period of validity
 - Subject
 - Subject's publika nyckel
 - Signatur
 - krypterad med CA:s privata nyckel

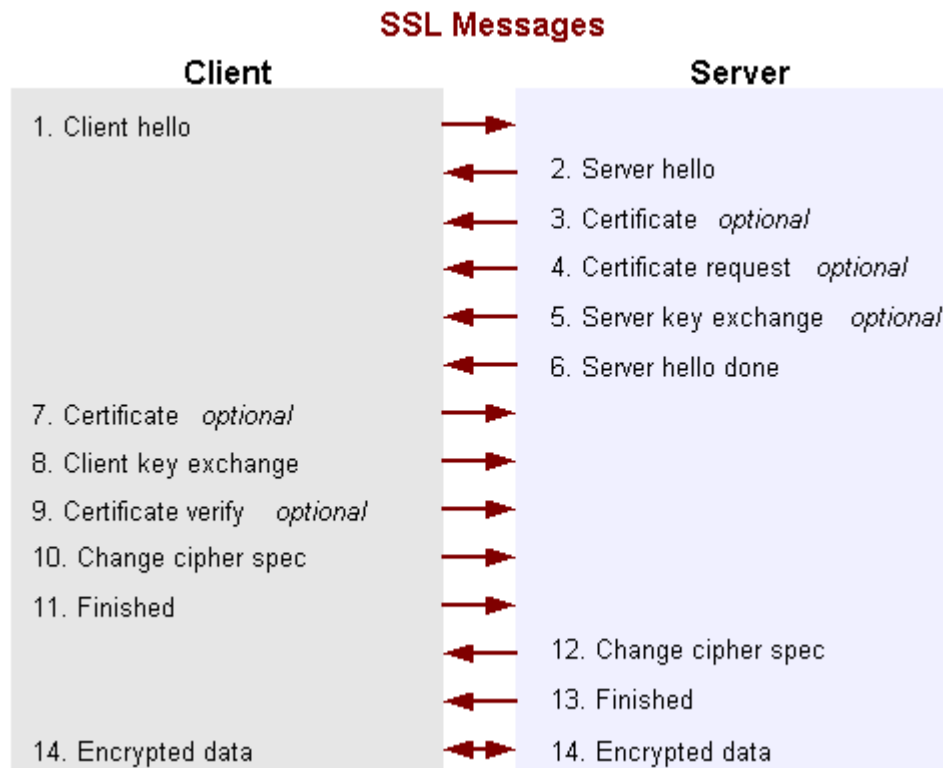
Digital signatur



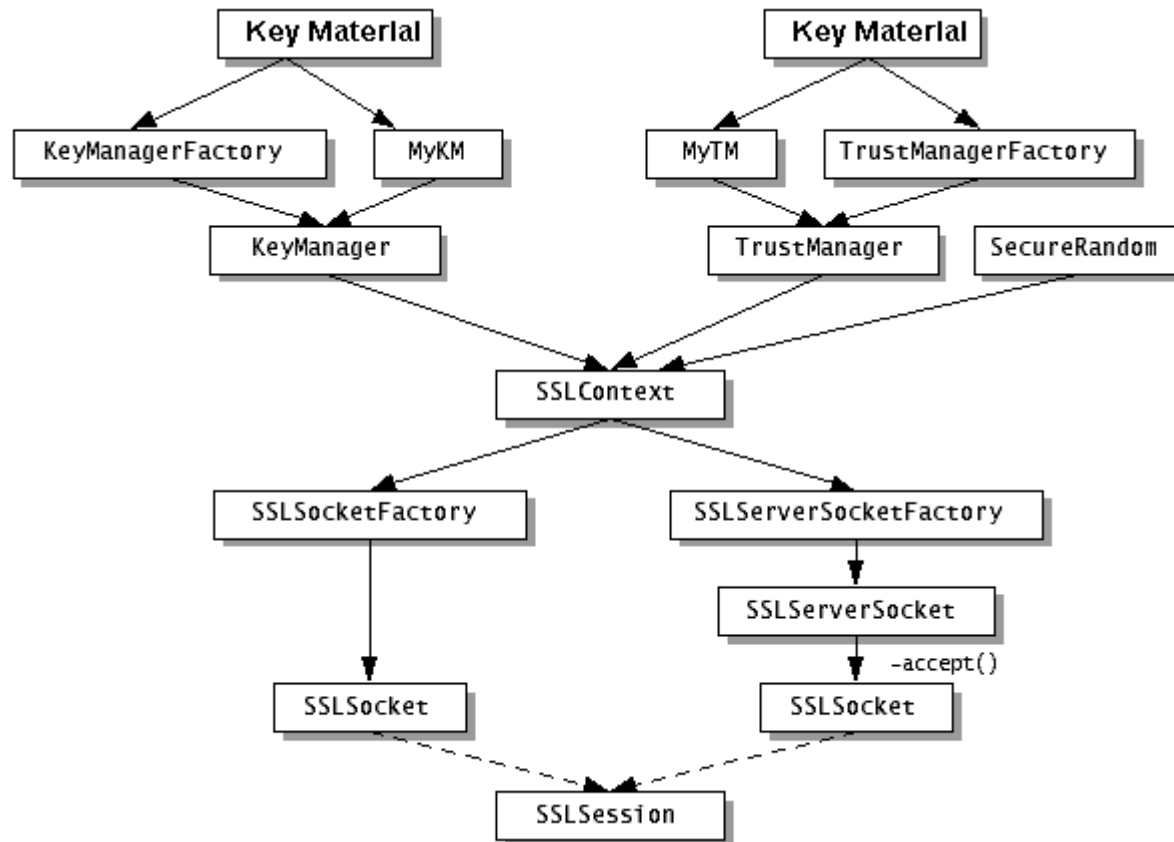
Transport Layer Security (TLS)

- Är ett protokoll
- Ligger mellan Transportlagret / Applikationslagret
- Specificerar ej krypteringsalgoritmer utan en metod att generellt inbädda PKI
- SSL 3.0 => TLS 1.2
- HTTPS använder port 443
- PKI i java => JSSE

Handshake



Klasser



Server.java (1/2)

```
import javax.net.ssl.*;
import java.io.*;

public class Server{
    public static void main(String[] args){
        SSLServerSocketFactory ssf =
            (SSLServerSocketFactory)SSLServerSocketFactory.getDefault();
        System.out.println("Stöder:");
        for(int i = 0; i < ssf.getSupportedCipherSuites().length; i++)
            System.out.println(ssf.getSupportedCipherSuites()[i]);
        SSLServerSocket ss = null;
        try{
            ss = (SSLServerSocket)ssf.createServerSocket(1234);
            String[] cipher = {"SSL_DH_anon_WITH_RC4_128_MD5"};
            ss.setEnabledCipherSuites(cipher);
```

Server.java (2/2)

```
System.out.println("Vald:");
for(int i = 0; i < ss.getEnabledCipherSuites().length; i++)
    System.out.println(ss.getEnabledCipherSuites()[i]);
    SSLSocket s = (SSLSocket)ss.accept();
BufferedReader infil =
    new BufferedReader(new InputStreamReader(s.getInputStream()));
String rad = null;
while( (rad=infil.readLine()) != null)
    System.out.println(rad);
infil.close();
}
catch(IOException e){
    System.out.println(e.getMessage());
}
}
}
```

Client.java (1/2)

```
import java.io.*;
import java.net.*;
import javax.net.ssl.*;

public class Client{
    public static void main(String[] args){
        SSLSocketFactory sf = (SSLSocketFactory)SSLSocketFactory.getDefault();
        for(int i = 0; i < sf.getSupportedCipherSuites().length; i++)
            System.out.println(sf.getSupportedCipherSuites()[i]);
        HTTPSURLConnection.setDefaultSSLSocketFactory(sf);
        SSLSocket s = null;
        try{
            s = (SSLSocket)sf.createSocket("my.nada.kth.se",1234);
        }
        catch(MalformedURLException e){
            System.out.println(e.getMessage());
        }
        catch(IOException e){
            System.out.println(e.getMessage());
        }
    }
}
```

Client.java (2/2)

```
for(int i = 0; i < s.getSupportedCipherSuites().length; i++)
    System.out.println(s.getSupportedCipherSuites()[i]);
String[] cipher = {"SSL_DH_anon_WITH_RC4_128_MD5"};
s.setEnabledCipherSuites(cipher);
for(int i = 0; i < s.getEnabledCipherSuites().length; i++)
    System.out.println(s.getEnabledCipherSuites()[i]);

PrintWriter utfil = null;
try{
    utfil = new PrintWriter(s.getOutputStream());
}
catch(IOException e){
    System.out.println(e.getMessage());
}
utfil.println("Hej");
utfil.close();
}
}
```


keytool

- `#!/bin/sh`
- `rm $home/.keystore`
- `keytool -genkey -keyalg "RSA" -storepass rootroot -validity 365 -alias SSLCertificate`
- `keytool -list -storepass rootroot`
- `keytool -export -alias SSLCertificate -storepass rootroot -file server.cer`

Certifikat 1/2

```
try{

    InputStream infil = new FileInputStream("server.cer");
    CertificateFactory cf = CertificateFactory.getInstance("X.509");
    X509Certificate cert = (X509Certificate)cf.generateCertificate(infil);
    infil.close();
}
catch(CertificateException e){
    System.out.println(e.getMessage());
}
catch(IOException e){
    System.out.println(e.getMessage());
}

KeyStore ks = null;
try{
    ks = KeyStore.getInstance("JKS", "SUN");
}
catch(KeyStoreException e){
    System.out.println(e.getMessage());
}
catch(NoSuchProviderException e){
    System.out.println(e.getMessage());
}
```

Certifikat 2 / 2

```
InputStream is = null;
try{
    is = new FileInputStream(new File("./keystore"));
}
catch(FileNotFoundException e){
    System.out.println(e.getMessage());
}
try{
    ks.load(is,"rootroot".toCharArray());
}
catch(IOException e){
    System.out.println(e.getMessage());
}
catch(NoSuchAlgorithmException e){
    System.out.println(e.getMessage());
}
catch(CertificateException e){
    System.out.println(e.getMessage());
}
```