Internetprogrammering
DD1386

Föreläsning 18

Innehåll

- Repetition från Gabbes föreläsningen
- Java Secure Socket Extension (JSSE)
- Skapa certifikat med keytool
- En hello world server baserd på HTTPS

Repetition

Problem:
1. Autentisering: Är den jag kommunicerar med ”rätt” person?
2. Avlyssning: Är det någon som tjuvlyssnar på de hemliga saker vi pratar om?
3. Data integritet: Den information jag fått är det exakt samma som skickats?

RSA

- Asymetrisk chiffer
- Två nycklar E och D.
- Om E väljs som Publik nyckel så D är då den privata nyckeln och vise versa.
- E och D väljs enligt följande:

1.Välj två (stora) primtal, p1 och p2

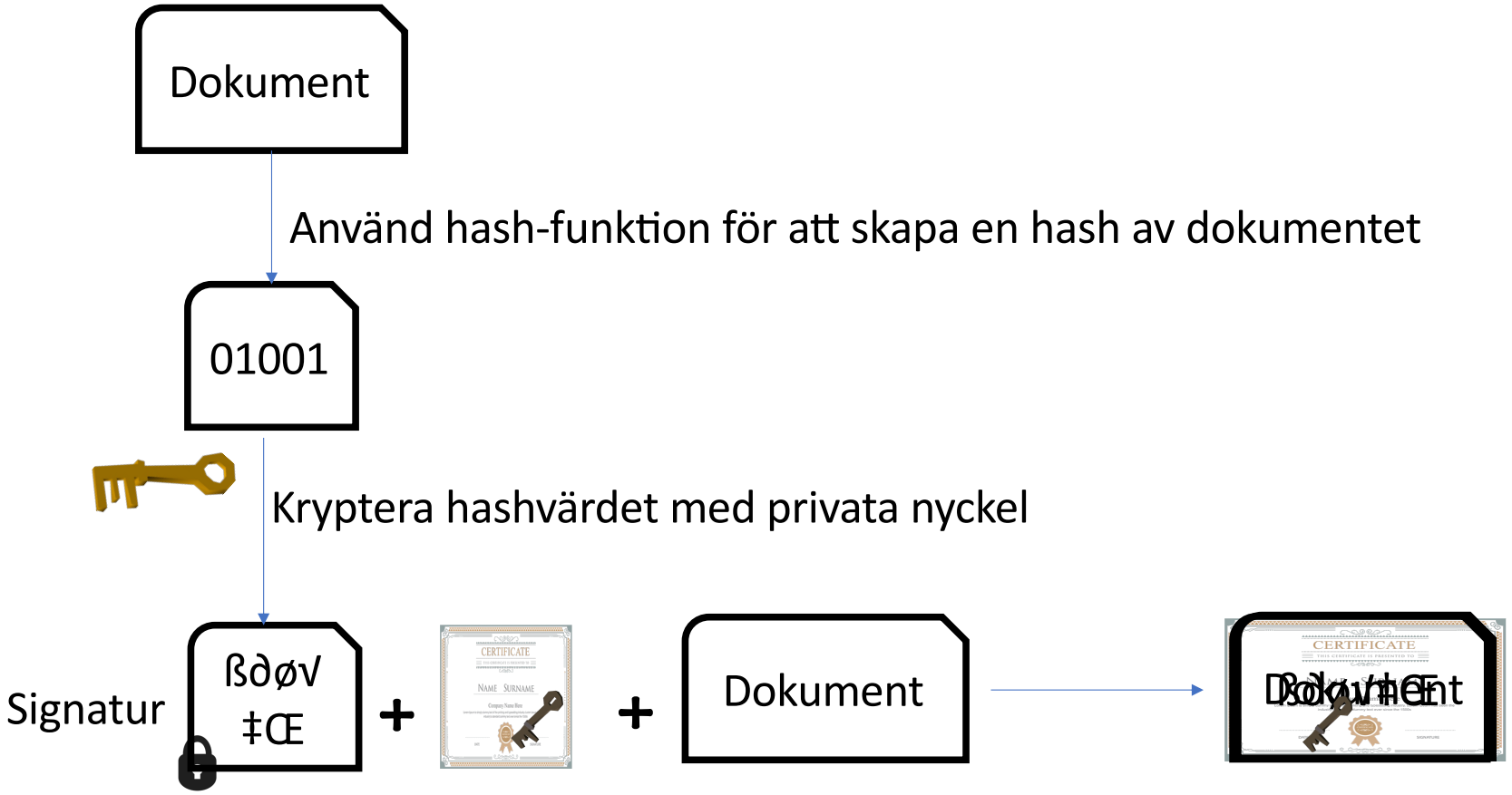
- n = p1* p2
- g = (p1-1)*(p2-1)
- Välj ett primtal, E, så att talet g är delbart med E.
- Välj ett primtal, D, så att resten av divisionen (D*E)/g blir talet 1.

Nu kan vi kryptera meddelandet M med nyckeln E och dekryptera med nyckeln D:

$Crypt(M) = C(M^E)\ \%\ n$

$Decrypt(Crypt(M)) = (C(M)^D)\%n = M$

Signering


Dokument
Använd hash-funktion för att skapa en hash av dokumentet
01001
Kryptera hashvärdet med privata nyckel
Signatur
ßðøV
‡Œ
+
+
Dokument
CERTIFICATE

Verifiering

Dokument

ßðøV
‡Œ

Signatur

Skapa hash av dokumentet

Dekryptera med publika nyckel

01001

?
=

01001

Själv-signerad cer?fikat(1/2)

1. Ta bort eventuellt äldre keystore filer:
`rm $HOME/.keystore`

`$HOME`: miljövariabeln för din hemkatalog på min dator `/Users/vahid`

Själv-signerad certifikat (2/2)

2. Skapa en ny keystore fil (nyckelpar och cer0fikat):

```
keytool -genkeypair -keyalg "RSA"
                    -storepass rootroot
                    -validity 365
                    -alias SSLCertificate
```

Keytool

• Se innehållet av keystore:

```
keytool -list -storepass rootroot
```

• Om du behöver exportera certifikatet och publika nyckeln till exempelvis klienter som behöver använda dina jar-signerade javafiler:

```
keytool -export -alias SSLCertificate -storepass
rootroot -file server.cer
```

Helloworld.java (1/4)

1. Ladda upp filen keystore:

```
KeyStore ks = KeyStore.getInstance("JKS", "SUN");
InputStream is = new FileInputStream( new
                     File("/home/vahid/.keystore"));
ks.load(is,"rootroot".toCharArray());
```

Helloworld.java (2/4)

2. Skapa KeyManagerFactory:

```
KeyManagerFactory kmf = KeyManagerFactory.getInstance(
                KeyManagerFactory.getDefaultAlgorithm());
kmf.init(ks,"rootroot".toCharArray());
KeyManager km[] = kmf.getKeyManagers();
```

Helloworld.java (3/4)

3. Sätt upp SSL-Context :

```
SSLContext sslContext = SSLContext.getInstance("TLS");
sslContext.init(km, null, null);
SSLServerSocketFactory ssf = (SSLServerSocketFactory)
                    sslContext.getServerSocketFactory();
SSLServerSocket ss = (SSLServerSocket)
                     ssf.createServerSocket(1234);
```

Helloworld.java (4/4)

4. Vänta på request och sedan utföra tjänsten:

```
SSLSocket s =(SSLSocket)ss.accept();
int i=0;
while (true){
   PrintStream response =
             new PrintStream(s.getOutputStream());
   response.println("HTTP/1.0 200 OK");
   response.println("Content-Type: text/html");
   response.println();
   response.println("Hello World!"+i);
   i++;
   s.close();
   s =(SSLSocket)ss.accept();
}
```

Testa din server

I de flesta system finns programmet curl (en textbaserad webbläsare). Det finns att ladda ner också på nätet.
I terminal:

```
curl -k https://127.0.0.1:1234
```

Eller se lite mer:

```
curl -vk https://127.0.0.1:1234
```

Eller använd Firefox och tillåt certifikatet manuellt, när webbläsaren frågar dig om tillåtelse.

********** *Men glöm inte **s**:et i http**s**://... i url:en.* **********