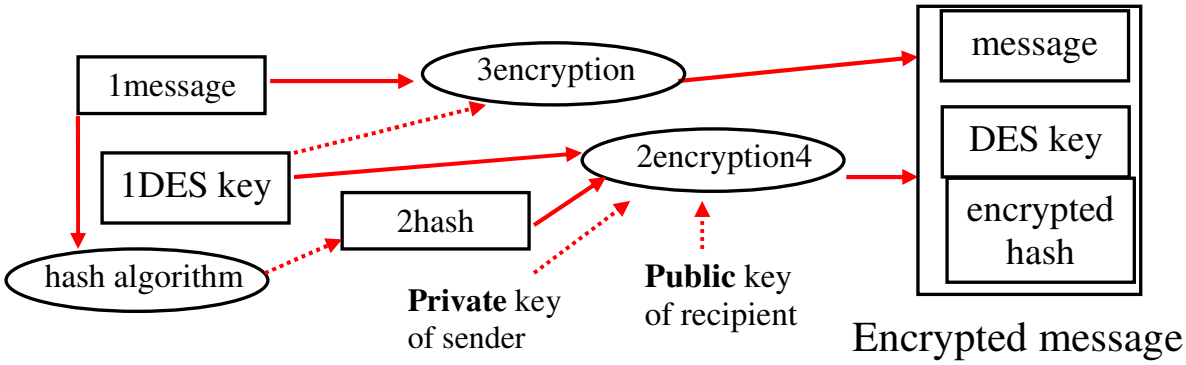
Crypto-programmering med Java

I programspråket Java är det förhållandevis enkelt att använda digitala signaturer och kryptering vilket finns i Java API:erna JCA (Java Cryptography Architecture) och JCE (Java Cryptography Extension).

Vi skall i denna labb skriva två program (uppgiften kan även utföras i ett program om tillräckligt med spårutskrifter används så man kan följa flödet) med hjälp av Java (J2SE 1.5.x eller nyare).
Det som skall skrivas är en modul som kan signera och kryptera och modul som kan dekryptera och verifiera korta meddelanden med en symmetrisk algoritm (AES, DES, Blowfish etc.). Den hemliga nyckeln för den symmetriska algoritmen skall krypteras med en assymetrisk algoritm (RSA, DSA etc.) och läggas till hela meddelandet.

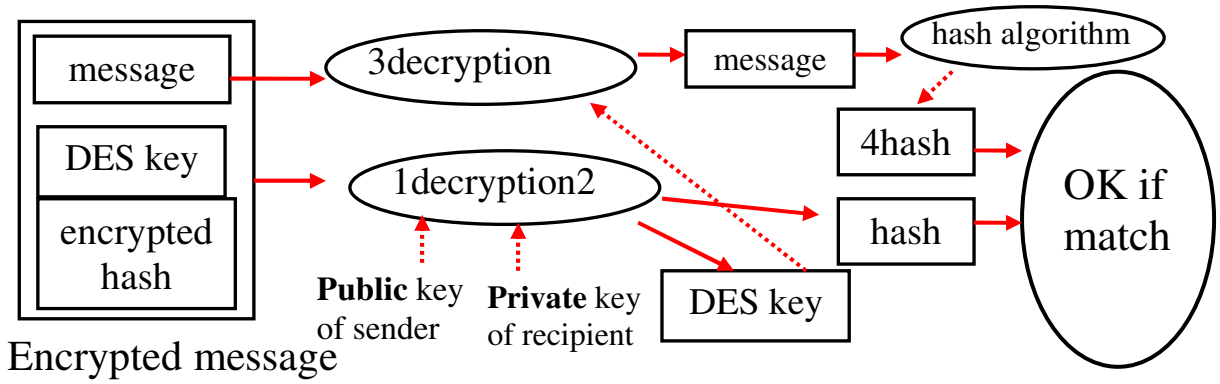
Ett nyckelpar (privat och publik nyckel) måste alltså också skapas. Det är viktigt att du tolkar den beskrivande texten korrekt. Pilar med punktlinje innebär att kryptering eller hashning utförs. Jag har satt en siffra framför eller bakom orden för att identifiera händelser. Se förklaring nedan.

Vad som händer vid **sändningen** av signerat krypterat meddelande är följande:



Sändning
1. Ett meddelande och en symmetrisk nyckel skapas.
2. En hashsumma beräknas på meddelandet och krypteras med sändarens privata nyckel.
3. Meddelandet krypteras med den symmetriska nyckeln.
4. Sändarens symmetriska nyckel krypteras med mottagarens publika nyckel och läggs i ”behållaren” tillsammans med meddelandet och hashen.
Obs! Ellipsen med både 2 och 4 i är egentligen två ellipser men de får inte plats i samma diagram.

Vad som händer vid **mottagningen** av ett signerat krypterat meddelande är följande:



Mottagning
1. Hos mottagaren dekrypteras symmetriska nyckeln med mottagarens privata nyckel.
2. Hashsumman dekrypteras med sändarens publika nyckel.
3. Sedan dekrypteras själva meddelandet med den symmetriska nyckeln som nyligen blev dekrypterad.
4. Slutligen så beräknas en hash på meddelandet och jämförs med den nyligen dekrypterade hashen som skickats med i ”behållaren”.
Obs! Ellipsen med både 1 och 2 i är egentligen två ellipser men de får inte plats i samma diagram.

Resurser på nätet som kan hjälpa dig:

- Herong's Tutorial Notes - http://www.herongyang.com/jdk/
- RSA info - http://www.math.su.se/~johan/lusttur/ht03/rsa/index.shtml
- RSA exempel - http://users.du.se/~hjo/cs/common/div/rsakryptering.zip

Java JCA and JCE

- http://java.sun.com/javase/technologies/security.jsp
- http://java.sun.com/j2se/1.5.0/docs/guide/security/
- http://java.sun.com/j2se/1.5.0/docs/guide/security/CryptoSpec.html
- http://java.sun.com/j2se/1.5.0/docs/guide/security/jce/JCERefGuide.html

Java Tutorial for signing code

- http://java.sun.com/docs/books/tutorial/security/apisign/index.html

Ett exempelprogram finns i labbkatalogen som använder en del kod från ovanstående resurser, med lite ändringar och tillägg så kan det lösa en stor del av uppgiften. Programmet är tillverkat med Netbeans 5.0 och bör gå att öppna ifrån en likadan eller nyare miljö hos er.

Netbeans är gratis och går att ladda hem ifrån http://www.netbeans.org/, använd den version som innehåller JDK (Java Developer Kit) integrerat så slipper du ladda hem JDK separat. Dokumentation till JDK finns på Suns hemsida.
java -jar Crypto.jar arg1 arg2 … kan det ena exempelprogrammet startas med, se källkoden för hjälp (obs! den är odokumenterad).
Det andra programmet är en applet och körs via en webbläsare.

Pseudokod för att lösa problemet
För att få G skall man lämna in ett program som fungerar. Funktionen för detta problem är i stort sett densamma som för krypterade och signerade mail.

Symmetrisk nyckel används för att kryptering/dekryptering av stora meddelanden går snabbare än med assymetrisk nyckel. Det spelar ingen roll med vilken assymetrisk nyckel du krypterar med bara det är den andra i nyckelparet som dekrypterar.

Om allt görs i ett program så skulle pseudokoden kunna vara denna:

1. Skapa mottagare och sändare objekt (name, assymKey).

2. Generera ett assymetriskt nyckelpar och tilldela ovanstående objekt nycklar.

3. Låt sändarobjektet generera ett meddelandeobjekt och en symmetrisk nyckel (message, hash, symKey).

4. Skapa meddelande, ta fram hashen på meddelandet, kryptera dessa med symmetriska nyckeln och spar till meddelandeobjektet.

5. Kryptera symmetrisk nyckel med en av assymetrisk nycklarna och spar till meddelandeobjektet.

6. Simulera sändning...

7. Låt mottagarobjektet dekryptera symmetriska nyckeln med den andra assymetriska nyckeln.

8. Dekryptera hash och meddelande med den hemliga symmetriska nyckeln.

9. Ta hashen på meddelandet igen och jämför mot den bifogade hashen i meddelandet.

10. Om hasharna är lika är meddelandet detsamma som sändes.

Redovisning:
Tillverka de två program eller ett om du valt den lösning som beskrivits ovan och bifoga nödvändiga nycklar (om det behövs) för att kunna testköra din lösning. Bifoga även enkel dokumentation om det krävs för att förstå hur programmen används.

Lägg dina svar från uppgiften i labben i en katalog, komprimera katalogen och lämna in den packade filen.