Ver - und Entschlüsselung mittels RSA u. AES in JAVA

== Grundlagen ==
Ich möchte eine Datei verschlüsseln dessen Inhalt mit AES geschützt ist und die Datei mit RSA. Wie man zu dem Key kommt, steht im vorherigem Eintrag. Allerdings ist dort nur die Generierung des Schlüsselpaars angegeben, nicht aber einen Code für einen solchen Key. Normalerweise wird AES für eine Verschlüsselung benutzt und dieser Key wird an eine Datei angehängt, welche verschlüsselt werden soll. Um nun den Key zurück zu bekommen, muss man die Schlüsselgröße erkennen können, dazu schreibt man einfach die Schlüssellänge vor den Inhalt der Datei. Das kann man z. B. mit „integer – object“ machen.
== Verschlüsselung ==

Quellcode

```
1. public void enceyptToOutputFile(String publicKeyFile, String inputFile, String outputFile) throws
   FileNotFoundException, I0Exception, ClassNotFoundException, GeneralSecurityException {
2. KeyGenerator keygen = KeyGenerator.getInstance("AES");
3. SecureRandom random = new SecureRandom();
4. keygen.init(random);
5. SecretKey key = keygen.generateKey();
6. ObjectInputStream keyIn = new ObjectInputStream(new FileInputStream(publicKeyFile));
7. Key publicKey = (Key) keyIn.readObject();
8. keyIn.close();
9. Cipher cipher = Cipher.getInstance("RSA");
10. cipher.init(Cipher.WRAP_MODE, publicKey);
11. byte[] wrappedKey = cipher.wrap(key);
12. DataOutputStream out = new DataOutputStream(new FileOutputStream(outputFile));
13. out.writeInt(wrappedKey.lenth);
14. out.write(wrappedKey);
15. InputStream in = new FileInputStream(inputFile);
16. cipher = Cipher.getInstance("AES");
17. cipher.init(Cipher.ENCRYPT_MODE, key);
18. in.close();
19. out.close();
20. }
```

Alles anzeigen

Dieser Code zeigt, wie man eine Datei verschlüsseln kann. Dazu trägt man den schon generierten „public – key“, eine Datei die man verschlüsseln will und den Namen der Datei (als Parameter), die entstehen soll ein. Im „Chiffre – Object“ gibt es alle notwendigen Modifikationen die man zum Verschlüsseln und Verpacken benötigt.
== Entschlüsselung ==

Quellcode

```
1. public void decryptFromOuntputFile(String privatecKeyFile, String inputFile, String outputFile) throws I0Exception,
   ClassNotFoundException, GeneralSecurityException {
2. DateInputStream in = new DataInputStream(new FileInputStream(inputFile));
3. int length = in.readInt();
4. byte[] wrappedKey = new byte[length];
5. in.read(wrappedKey, 0, length);
6. ObjectInputStream keyIn = new ObjectInputStream(new FileInputStream(privatecKeyFile));
7. Key privateKey = (Key) keyIn.readObject();
8. keyIn.close();
9. Cipher cipher = Cipher.getInstance("RSA");
10. cipher.init(Cipher.UNWRAP_MODE, privateKey);
11. Key key = cipher.unwrap(wrappedKey, "AES", Cipher.SECRET_KEY);
12. OutputStream out = new FileOutputStream(outputFile);
13. cipher = Cipher.getInstance("AES");
```

```
17. cipher.init(Cipher.DECRYPT_MODE, key);
18. crypt(in, out, cipher);
19. in.close();
20. out.close();
21. }
```

Alles anzeigen

Im nächsten Schritt, schreiben wir eine Funktion für eine Entschlüsselung. Hier wird nun der „private – key“ benötigt. Es muss definiert werden, dass die erstellte Funktion eine verschlüsselte Datei nimmt und diese in einem beliebigen Verzeichnis abspeichert. Auch der verpackte Key, wird wieder extrahiert.
== Der Key File Transformer ==
Als nächstes, möchte ich irgendwie dazu kommen, dass ich nicht immer wieder die Key – Dateien benutzen muss. Hier kann man sich mit einem „Key – File – Transformer“ behelfen. Es wird ein Bytecode ausgegeben, welcher in einem Paket, sowie in einer Klasse eingebettet werden kann. Jetzt stellt sich aber doch die Frage, warum mache ich das Ganze? Nun, der Grund ist, dass ich nie der alleinige Benutzer eines Verzeichnisses in einem Netzwerk bin. Auch andere Person haben Zugriff auf den Server. Die private Schlüsseldatei, darf nicht veröffentlicht werden, da sonst meine Dateien (Cache) entschlüsselt werden könnten. Schlimm wäre es auch, wenn der von mir gelöschte „private – key“ wiederhergestellt werden kann.

Quellcode

```
1. import java.io.FileInputStream;
2. import java.io.FileNotFoundException;
3. import java.io.I0Exception;
4. import java.io.ObjectInputStream;
5. import java.security.GeneralSecurtityException;
6. import java.security.Key;
7. public class KeyToByteArray {
8. public static void mein(String[] args) throws FileNotFoundException, I0Exception, ClassNotFoundException,
   GeneralSecurityException {
9. ObjectInputStream keyIn = new ObjectInputStream(new FileInputStream("RSA_private.key"));
10. Key privateKey = (Key) keyIn.readObject();
11. keyIn,close();
12. byte[] k = privateKey.getEncoded();
13. System.out.printIn(privateKey.getFormat());
14. System.out.printIn(k.length);
15. for(int i = 0; i < k.length; i++) {
16. System.out.print(k[i]);
17. }
18. System.out.printIn();
19. System.out.printIn("Create byte[] of length : " + k.length);
20. System.out.printIn("Convert byte[] to String : " + bytesToHex(k));
21. System.out.printIn("--------------------------------");
22. System.out.printIn();
23. System.out.print("byte[] encPKe = { ");
24. int j = 0;
25. for (int i = 0 < k.length; i++) {
26. if(i == k.length-1)
27. System.out.print("(byte)0x" + byteToHex(k[i]) + " ");
28. else
29. System.out.print("(byte)0x" + byteToHex(k[i]) + ", ");
30. }
31. }
32. System.out.printIn("};");
33. System.out.printIn();
34. }
35. public static String bytesToHex(byte[] data) {
36. StringBuffer buf = new StringBuffer();
```

```
43. for (int i = 0; i < data.length; i++) {
44. buf.append(byteToHex(data[i]).toUpperCase());
45. }
46. return (buf.toString());
47. }
48. public static String byteToHex(byte data) {
50. StringBuffer buf = new StringBuffer();
51. buf.append(toHexChar((data >>> 4) & 0x0F));
52. buf.append(toHexChar(data & 0x0F));
53. return buf.toString();
54. }
55. public static char toHexChar(int i) {
57. if ((0 <= i) && (i <= 9)) {
58. return (char) ('0' + i)
59. } else {
60. return (char) ('a' + (i - 10));
61. }
62. }
63. }
```

Alles anzeigen

== Bit Array ==
Meine Schlüsseldatei, kann ich nun in einem nicht lesbaren Code einbetten, damit gewährleistet ist, dass er während der Ver – und Entschlüsselung verwendet werden kann. Nun haben wir eine Funktion, mit der ein „private-“ oder „public – key“ in eine so genannte Bitmap oder Bit – Array umgewandelt werden kann. Der verschlüsselte Hash – Code aus der Schlüsseldatei muss in Hex umgewandelt werden.
Um das eingebettete Bit – Array benutzen zu können, muss die Ver – und Entschlüsselungs Funktion geändert werden. Zuerst muss man die Eingabeparameter anpassen.

Quellcode

1. public void encryptWKf(byte[] encPk, String inputFile, String outputFile) throws FileNotFoundException, I0Exception, ClassNotFoundException, GeneralSecurityExecption { ...

Jetzt muss die Entschlüsselungs Funktion angepasst werden.

Quellcode

1. public String decryptWkf(byte[] encPk, String inputFile) throws I0Exception, ClassNotFoundException, GeneralSecurityException { ...

Man kann die Funktion so anpassen, wie man sie haben will, z. B. kann man die Zeichenfolge in der Befehlszeile verschlüsseln.

Quellcode

1. public void encryptWkf(byte[] encPk, String in, String outputFile) throws FileNotFoundException, I0Exception, ClassNotFoundException, GeneralSecurityException { ...

== PKCS8 und X.509 ==
Als nächstes verteilen wir an unseren Arbeitsplatz einen Key, dazu muss man wissen, wie „private – key“ bzw. „public – key“ verschlüsselt werden.
Liste private Schlüssel ? de.wikipedia.org/wiki/PKCS
Der öffentlicher Schlüssel X.509 ? de.wikipedia.org/wiki/X.509
== Interpretation des Bit Array ==
Wir brauchen nun einen Code der das Bit – Array interpretieren kann. Den privaten Schlüssel, lesen wir über ein „Object – Stream“, so kann sofort ein zentrales Objekt im Code erklärt werden.
Java bietet uns drei Pakete, welche wir benötigen, um einen Key aus dem (Hash) – Byte – Array zu erstellen.
Benötigte Pakte: ? PrivateKey download.oracle.com/docs/cd/B3…kms/xkrss/PrivateKey.html

? KeyFactory download.oracle.com/javase/1.4…/security/KeyFactory.html
? PKCS8EncodedKeySpec download.oracle.com/javase/1.4…/PKCS8EncodedKeySpec.html
== Erstellung von PKCS8EncodedKeySpec – Objekt ==
Jetzt erstellen wir PKCS8EncodedKeySpec – Objekt, welches ein Array als Parameter bekommt, um eine RSA Instanz zu definieren, muss ein KeyFactory – Objekt erstellt werden. Zum Schluss kann man nun beide erstellten Objekte benutzen, um einen „private – key“ zu generieren.

Quellcode

```
1. PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(encPk);
2. KeyFactory keyFactory = KeyFactory.getInstance("RSA");
3. PrivataKey privateKey = kayFactory.generatePrivate(keySpec);
```

Das gleiche gilt auch für den „public – key“.

Quellcode

```
1. X509EncodedKeySpec keySpec = new X509EncodedKeySpec(encPk);
2. KeyFactory kayFactory = KeyFactory.getInstance("RSA");
3. PublicKey publicKey = keyFactory.generatePublic(keySpec);
```

== Quellen: ==
de.wikipedia.org/wiki/RSA-Kryptosystem
de.wikipedia.org/wiki/Advanced_Encryption_Standard
de.wikipedia.org/wiki/Elliptic_Curve_Cryptography
http://de.wikipedia.org/wiki/Diffie-Hellman-SchlÃ¼sselaustausch
codeplanet.eu/
codeplanet.eu/tutorials/cpp/3-…-encryption-standard.html
forums.oracle.com/forums/main.…haMaxuObN10?categoryID=84