Einen eigenen RSA Key Generator erstellen

== Grundlagen ==
Das wichtigste Paket von JAVA, dass man bei allen JAVA – Operationen benötigt, ist das „java.security.“ - Standard Paket. Dies ist, nach dem man JAVA installiert hat normalerweise vorhanden.
Als erstes, möchte ich einen normalen Key – Generator erklären. Andere Versionen sind im Internet vorhanden wie z.B. auf ? codeplanet.eu/
Die maximale Schlüsselgröße ist 2048 Bit, diese ist aber auf Grund der starken Richtlinien – Zuständigkeit in JAVA 2 SDK begrenzt. Weitere Informationen zu diesem Thema, wieder im Netz oder direkt unter ? oracle.com/technetwork/java/index.html
Es gibt aber auch ein Prüfungsverfahren, welches ohne Bedingung heruntergeladen werden kann. Dies ist bei JAVA Cryptography Extension kurz JCE auch wieder unter ? oracle.com/technetwork/java/index.html erhältlich.
So kann man nun einen 8192 Bit RSA Key erstellen.

== Code für den Generator ==
Quellcode

```
1. import java.io.FileOutputStream;
2. import java.io.IOException;
3. import java.io.ObjectOutputStream;
4. import java.security.GeneralSecurityException;
5. import java.security.KeyPair;
6. import java.security.KeyPairGenerator;
7. import java.security.SecureRandom;
8. public class RSAKeyGenerator {
9. private static final int KEYSIZE = 8192 //KEYSIZE = 8192 -> Schlüsselgröße in Bits
10. public static void main(String[] args) {
11. generateKey("RSA_private.key","RSA_public.key"); //generierte Dateien
12. }
13. public static void generateKey(String privateKey, String publicKey) {
14. try {
15. KeyPairGenerator pairgen = KeyPairGenerator.getInstance("RSA");
16. SecureRandom random = new SecureRandom();
17. pairgen.initialize(KEYSIZE, random);
18. KeyPair keyPair = pairgen.generateKeyPair();
19. ObjectOutputStream out = new ObjectOutourStream(new FileOutputStream(publicKey));
20. out.writeObject(key.Pair.getPublic());
21. out.close();
22. out = new ObjectOutputStream(new FileOutputStream(privateKey));
23. out.writeObject(keyPair.getPublic());
24. out.close();
25. } catch (IOException e) {
26. System.err.println(e);
27. } catch (GeneralSecurityException e) {
28. System.err.println(e);
29. }
30. }
31. }
```

Alles anzeigen

== Kompilieren ==
Jetzt muss dieser Code kompiliert und in der Befehlszeile ausgeführt werden. Wenn man zum exportieren eine .jar – Datei wählt, muss man java -jar binary.jar ausführen. Nachdem man diesen Code ausgeführt hat, sind zwei neue Dateien im aktuellen Verzeichnis entstanden.

1. Datei = "RSA_private.key"

2. Datei = "RSA_public.key"

Diese beiden Dateien, werden mit einer Schlüsselgröße von 8192 Bits hergestellt. So wie sie entstanden sind, sind diese auch schon bereit und können genutzt werden.
Der Public – Key wird zum Verschlüsseln benutzt und der Private – Key zum Entschlüsseln von ausgewählten Dateien.

== Quellen: ==
de.wikipedia.org/wiki/RSA-Kryptosystem
codeplanet.eu/
oracle.com/
oracle.com/technetwork/java/index.html
de.wikipedia.org/wiki/Asymmetrisches_Kryptosystem
de.wikipedia.org/wiki/Symmetrisches_Kryptosystem

== Weiteres zu diesem Thema ==
Vielleicht interessiert sich der Eine oder Andere für dieses Thema und hätte gerne Weiteres zum Thema Verschlüsselungen und Entschlüsselungen bezogen auf dieses Thema, kann ich nach Verlangen zusammenschreiben und hier ablegen oder per email zukommen lassen. Oder einfach mal hier ? http://www.codeplanet.eu/tutorials/java/…sa-in-java.html
hinein schnuppern.

MfG >>>spaceher0<<<