**<---- TFG ---->**

* **Conocimientos**

Al parecer se usa mucho más openSSL que las librerías de JAVA, entonces quizá interese hacer una llamada a openSSL. Después he visto que hay librerías de Java de sobra para RSA, hay que seguir investigando.

Mil bits unos 300 dígitos decimales.

* **Ataques**

En un futuro la computacion quántica.

Factoriza primos: <https://sourceforge.net/projects/msieve/files/msieve/Msieve%20v1.51/>

Canal lateral: RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis [5], en el que los autores demuestran cómo es posible extraer, sin demasiadas complicaciones todo hay que decirlo, una clave privada nada menos que de 4.096 bits. La única solución es enmascarar estos efectos físicos mediante técnicas de tipo TEMPEST y similares, lo cual ya se ha hecho con GnuGP de la serie 1.x y posteriores como ha sido comentado al comienzo de este artículo.

* **Libros y Documentación**

<http://www.criptored.upm.es/crypt4you/temas/RSA/leccion6/leccion06.html> y todas las lecciones

TFG de otra persona: <https://www.iit.comillas.edu/pfc/resumenes/46eab0a6a63d1.pdf>

* + **Obligatorio Leer**

**Cosas de Codigo**

XX <http://www.thedeveloperspoint.com/implementing-rsa-in-java/>

XX <http://www.thedeveloperspoint.com/using-java-security-libraries-for-implementing-rsa/>

X es basura <http://java-limos.blogspot.com.es/2011/05/encriptacion-utilizando-el-algoritmo_25.html>

XX <https://javadigest.wordpress.com/2012/08/26/rsa-encryption-example/>

XX mas ejemplos en la pagina <http://www.java2s.com/Tutorial/Java/0490__Security/BasicRSAexample.htm>

XX <http://chuwiki.chuidiang.org/index.php?title=Encriptacion_con_Java>

XX <http://stackoverflow.com/questions/26978189/simple-rsa-encryption-using-javas-built-in-library>

Me falta por leer <https://www.example-code.com/java/rsa.asp>

**Otras Cosas**

Realmente no es nada: <https://community.rsa.com/community/products/securid/authentication-agent-api-81/content?filterID=contentstatus%5Bpublished%5D~category%5Bdevelopment%5D>

Página de rsa: <https://www.rsa.com/en-us>

X es pura basura, código con fallos <http://www.redinfocol.org/cifrados-rsa/>

Me falta por leer pildoras formativas [http://www.criptored.upm.es/thoth/index.php#](http://www.criptored.upm.es/thoth/index.php)

Me falta por leer <http://www.di-mgt.com.au/rsa_alg.html>

Me falta por leer <https://github.com/eduardolundgren/rsa-prime-factorization/blob/master/src/RSA.java>

Me falta por leer <https://github.com/YYZ/RSA/blob/master/crypto.java>

Me falta por leer <https://en.wikipedia.org/wiki/RSA_(cryptosystem)>

* **Librerias**

<https://www.google.es/search?q=apijava+RSA&ie=utf-8&oe=utf-8&client=firefox-b-ab&gfe_rd=cr&ei=h2eXWInFK7L38AfM3qzADw#q=api+java+RSA>

**API clases y paquetes JAVA**

Todas las clases: <https://docs.oracle.com/javase/7/docs/api/allclasses-noframe.html>

Clase BigInteger: <https://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html>

Package security and crypto

<http://docs.oracle.com/javase/7/docs/api/java/security/package-summary.html>

<http://docs.oracle.com/javase/7/docs/api/javax/crypto/package-summary.html>

Clases útiles RSA:

RSAKey , RSAKeyGenParameterSpec , RSAMultiPrimePrivateCrtKey , RSAMultiPrimePrivateCrtKeySpec , RSAOtherPrimeInfo , RSAPrivateCrtKey , RSAPrivateCrtKeySpec , RSAPrivateKey , RSAPrivateKeySpec , RSAPublicKey , RSAPublicKeySpec , RSAKeys?

Clases útiles KEY:

Key , KeyPublic , KeyPrivate , KeyGenerator , KeyPairGenerator , KeyFactory??, KeyPair , KeyPairGenerator

Clases útiles cipher:

Cipher , CipherInputStream , CipherOutputStream , EncryptedPrivateKeyInfo , NullCipher , SecretKeyFactory

Otras clases: SecureRandom , BigInteger , Certificate

EXCEPTIONS--> <https://docs.oracle.com/javase/7/docs/api/java/security/package-summary.html>

**Otras librerias**

otra libreria por si no se quiere usar las librerias desarrolladas por java: <https://github.com/google/keyczar>

otra mas <http://www.bouncycastle.org/latest_releases.html>

otra para noobs <http://www.jasypt.org/index.html>

oracle java

<http://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html>

<https://docs.oracle.com/cd/E19944-01/819-4480/IM_installation_appendixF_JCE.html>

libreria oracle RSA

<https://docs.oracle.com/cloud/latest/as111170/SCRPJ/index.html?oracle/security/crypto/core/RSA.html>

<https://docs.oracle.com/cloud/latest/as111170/SCRPJ/oracle/security/crypto/core/RSA.html>

--------------------

X <http://www.criptored.upm.es/descarga/articuloRSAcumple36.pdf>

X canal lateral: <http://www.criptored.upm.es/descarga/ListadoActasCIBSI_TIBETS2013.zip>

X va con el de arriba (son las diapos): <http://www.cibsi.utp.ac.pa/documentos/Ataques_por_Canal_Lateral.pdf>

X canal lateral shamir: <http://www.tau.ac.il/~tromer/papers/acoustic-20131218.pdf>

X <http://www.criptored.upm.es/software/sw_m001d.htm>

X Criptoretos: <http://www.criptored.upm.es/paginas/criptoretos.htm>

**java.security.KeyPairGenerator:**  
The KeyPairGenerator class is used to generate pairs of public and private keys. Key pair generators are constructed using the getInstance factory methods (static methods that return instances of a given class). A Key pair generator for a particular algorithm creates a public/private key pair that can be used with this algorithm. It also associates algorithm-specific parameters with each of the generated keys.

**java.security.KeyPair:**  
This class is a simple holder for a key pair (a public key and a private key). It does not enforce any security, and, when initialized, should be treated like a PrivateKey.

**javax.crypto.Cipher:**  
This class provides the functionality of a cryptographic cipher for encryption and decryption. It forms the core of the Java Cryptographic Extension (JCE) framework.

In the getInstance Using modes such as CFB and OFB, block ciphers can encrypt data in units smaller than the cipher’s actual block size. When requesting such a mode, you may optionally specify the number of bits to be processed at a time by appending this number to the mode name as shown in the “DES/CFB8/NoPadding” and “DES/OFB32/PKCS5Padding” transformations.

Codigo guay

*runningTime* = System.*nanoTime*();

System.***out***.println("RunningTime: "

+ (*runningTime* = System.*nanoTime*() - *runningTime*) / 1000000

+ " ms");

Horas dedicadas:

Hasta viernes 10 de febrero: estimadas 10 ~14 horas

Sabado 11 de febrero: 4,5 horas

Domingo 12 de febrero: 4 horas

Del 13 al 17 de febrero: 2 horas por dia.

Sabado 18 de febrero: 10:45 empiezo programar, 13:00 paro. 19:42 empiezo, 20:52 termino

Domingo 19 de febrero: 11:45 -13:20 cansado. 17:05- 19:00. 23:00 – 00:00

<http://www.eclipse.org/efxclipse/install.html>.

- Relativas a los elementos Javafx a utilizar. Por ejemplo: **import**

**javafx.scene.control.ProgressIndicator** para el indicador de progreso.

- Relativas a las estructuras de datos a utilizar: Collections, BitSet, List etc.

- Relativas a “.math”: BigDecimal, BigInteger. Estas son necesarias dados los grandes

números a utilizar.