Segurança de Redes e Sistemas de Computadores 2019/2020
1º Semestre Aula Prática (LAB 4 – LAB 4.1)

Guidelines

Goals

This Lab is dedicated to programming with Asymmetric Cryptography in JAVA, supported by JCA / JCE. The activities, experiences and exercises include some relevant observations in using asymmetric cryptographic methods and algorithms:

- Use of asymmetric cryptographic algorithms and their parameters, to encrypt and decrypt data or messages, with an understanding of the correction and limitations of these parameters in practical constructions.
- Use of asymmetric cryptographic constructions and their parameters, for the production of cryptographic envelopes, which can be used to confidentially protect different types of contents. In particular, we use such envelopes to protect the distribution of secrets, cryptographic keys or sensitive parameters for security associations between two or more principals (or processes).
- Understanding the control of generation and use of asymmetric cryptographic keys and related computational weight, as well as the facilities for the management of such keys (or keypairs) in Java keystores, according to the types of that keystores and the respective Java tools (keytool, KeyStore Explorer)

Sequence of proposed activities

Download the Lab 4.1 archive. It contains different code examples and demos that must be ready to compile, execute, parameterize and modify, for different experimental observations. This must be done based on a theoretical rationale on the operation of the various algorithms that are being used.

1) 1-Using RSAandElGamal

Use of asymmetric cryptographic algorithms for data encryption / decryption, using in this case the RSA algorithm

Observe the BaseRSAExample.java example experimentally. Follow the exercises and practical observations that will be proposed at the LAB.

Then look at the examples RSA.java and ElGamal.java to see the similarities in the use of asymmetric cryptographic constructions. You can see how the code structure and parameterizations allow for portable code that can be configured with different algorithms and related parameters. Note the differences between the use of the ElGamal algorithm and the RSA (which you can then complement with your theoretical knowledge).

2) 2-PKCS1-PaddedRSA

See and understand the parameterization of Standard Padding, in the case of RSA.
Understand why Padding is important for security and see why using Padding in this case is very different (in the effect and in the purpose) than using Padding in symmetric encryption.
Remember that, in the lecture class, different padding patterns and schemes were introduced and discussed in terms of security an purpose.

Also check that compared to the previous observation, we now start generating the pair keys avoiding their static initialization.

See the effect and experimentally observe the repercussions of the generation time of RSA key pairs with different sizes (in bits): 128, 256, 512, 1024, 2048, 4096, 8192 ...

Remember what you have learned (lecture classes) about the RSA processing complexity for encryption/decryption or for signing/verification operations. Try to observe experimentally the effect of the complexity when we have different keysizes in the generated keypairs.

See also a very important issue: we cannot operate with RSA messages bigger than the keyzizes (because this determines the modulus in the RSA processing of modular exponential operations). Considering this observation and the performance of RSA, comparing with symmetric cryptographic algorithms, what does it suggests for the practical use of RSA?

3) 3-OAEP-PaddedRSA

Repeat the previous practical observations, now in the case of using RSA with OAEP standardized padding. Consolidate your knowledge about the use of RSA and how to use it with different parameterizations.

4) 4-CostKeyGenerationRSA

The objective here is to have the sensitivity of the impact of dynamic generation of RSA key pairs with different sizes.

Compare RSA key generation notes with different sizes using the openssl tool:
$ openssl genrsa NNNN
with NNNN = 512, 1024, 2048, 4096, 8192, etc…

What do you observe?

Try a comparative observation on the performance of different cryptographic algorithms (symmetric, asymmetric and secure hash functions)

openssl speed md5 sha1 sha256 sha512

openssl speed des cast bf aes
openssl speed aes rc4
openssl speed sha512 aes rsa
openssl speed rsa dsa
openssl speed ecdsa

What do you observe?

5) 5-KeystoresManipulation

Explore the keytool Java tool and the code provided to have a reference on how you can use keystores to manage keys (keypairs, and public vs. private keys). Learn how the keytool can be used to manipulate those keystores (which can be generated/created in different keystore types). You can also check how simple it is to manipulate keys in keystores in your Java programs.

6) 6-openssl

Explore the openssl tool to see that it can is can be used for the generation of keypairs (for RSA for example) and to store/manage the generated keys or public key certificates in files, with standard representation formats: PEM, PKCS#12. See how these formats can be interoperated between the openssl tool and the Java keytool.
Analyze the format of PEM Files.