**Criptografia**

Resumos:

* [Mecanismos criptográficos](http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/09-crypto/criptografia/index.html)
* [Segurança na plataforma Java](http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/09-crypto/seguranca-java/index.html)
* [Representação de dados binários em texto](http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/09-crypto/binarios-texto/index.html)

Exemplos:

* [Testes de Criptografia em Java [ZIP](http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/09-crypto/crypto.zip)](http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/09-crypto/crypto.zip)
  + **Cifra simétrica**
  + **Cifra assimétrica**
  + Resumo (*digest*)
  + Geração de números aleatórios seguros
  + Assinaturas digitais
  + Leitura e escrita de chaves criptográficas
  + Criptografia em texto (XML)

    Para executar apenas o teste da cifra assimétrica, escrever:  
    mvn test -Dtest=AsymCrypto\*#testCipherPublic\*  
 

* [Certificados digitais [ZIP](http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/09-crypto/cert-util.zip)](http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/09-crypto/cert-util.zip)
  + **Assinaturas digitais**
    - com chave privada em ***keystore***
    - com chave pública em **certificados X.509**
  + Verificação de assinatura de certificado

    Para executar apenas os testes de assinatura digital, escrever:  
    mvn test -Dtest=X509DigitalSig\*  
 

**Exercício**

**Terceira parte do projeto**

O [ponto de continuação](http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/proj/P3_continue.zip) contém novos módulos, nomeadamente o módulo security onde deverão ser criadas classes para a implementação da segurança (criptografia, *handlers*, etc).  
Começar por editar os pom.xml para substituir CXX pelo identificador do grupo.  
Adicionar os novos módulos ao projeto no Git.

O objetivo deste exercício é criar métodos para proteger dados com uma cifra assimétrica.  
Uma chave pública será usada para cifrar os dados.  
Uma chave privada será usada para decifrar.  
Deste modo é possível garantir que apenas o dono da chave privada consegue ler a informação cifrada.

1. Criar a classe CryptoUtil no módulo security.
2. Criar método asymCipher() que recebe dados (byte[]) e uma chave e devolve esses dados cifrados.
   * Usar javax.crypto.**Cipher** para "RSA/ECB/PKCS1Padding"
   * Cifrar com:  
     cipher.init(Cipher.ENCRYPT\_MODE, publicKey);  
     byte[] cipherBytes = cipher.doFinal(plainBytes);
3. Criar o método asymDecipher() na classe que recebe os dados cifrados e uma chave e devolve os dados decifrados.
   * Decifrar com:  
     cipher.init(Cipher.DECRYPT\_MODE, privateKey);
4. Criar testes JUnit na classe CryptoUtilTest para verificar o funcionamento correto dos métodos implementados.
   * E também para verificar o que acontece quando algo não está correto (ex. chave errada, dados adulterados, etc).

*Sugestões:*

* Utilize este par de chaves para testes (coloque os ficheiros em src/test/resources):
  + Chave pública armazenada em [certificado](http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/09-crypto/example.cer)
  + Chave privada armazenada em *[keystore](http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/09-crypto/example.jks)*
    - Store password: 1nsecure
    - Key password: ins3cur3
* Pode imprimir os dados binários em texto com javax.xml.bind.DatatypeConverter.printHexBinary (hexadecimal) ou printBase64Binary (base 64)
* Melhore o código para lidar bem com as [exceções](http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/01-tools/exceptions/index.html).

**Próximos passos**:

* Criar métodos na classe CryptoUtil para fazer e verificar assinatura digital
  + Usar java.security.**Signature** para "SHA256withRSA"
* Usar a classe CryptoUtil juntamente com *Handlers* intercetores de mensagens SOAP para proteger as comunicações no projeto
  + Adicionar a dependência para o módulo security onde necessário
  + A comunicação entre mediator-ws-cli e mediator-ws deve ocultar o número do cartão de crédito
  + A comunicação entre supplier-ws-cli e supplier-ws deve garantir a autenticidade, integridade e frescura das mensagens

**Bom trabalho!**