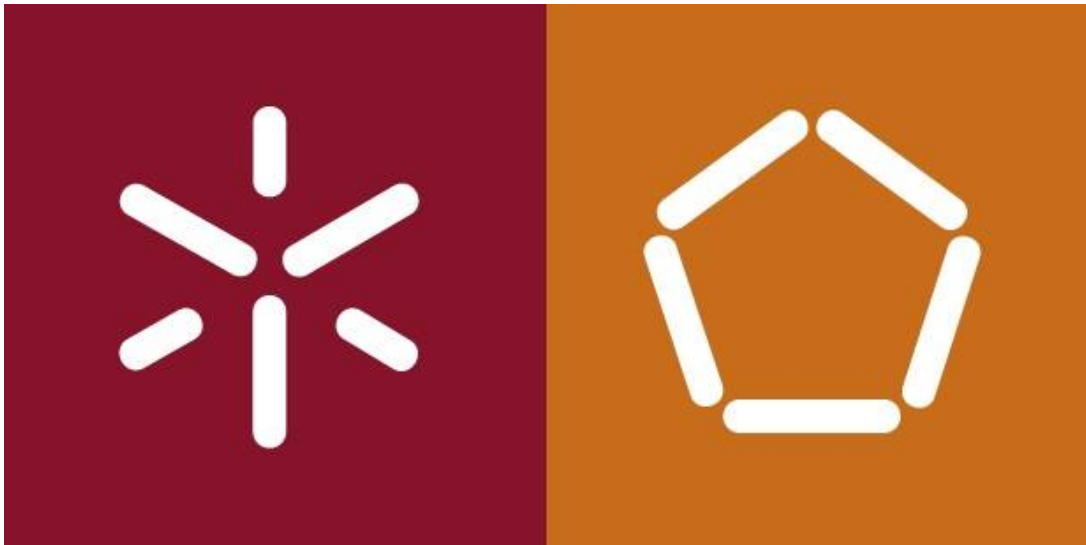


Universidade do Minho



Projeto em tecnologia de identificação - Assinaturas eIDAS

Mestrado Integrado em Engenharia Informática

Engenharia de Segurança

2º Semestre , 2018/2019

Grupo 12

A77070 - João Pedro Pereira Alves

A70132 - Nuno André Lopes Leite

Gualtar, Braga
19 de Junho de 2019

Conteúdo

1	Introdução	2
2	Solução	3
3	Implementação	3
3.1	Configuração do ambiente de demonstração	3
3.2	Disponibilização de Funcionalidades do DSS	4
4	Manual de Utilização	5
5	Resultados	7
6	Conclusão	9
A	Vagrantfile	10

1 Introdução

A resolução deste trabalho prático surge no âmbito da unidade curricular de Engenharia de Segurança, pertencente ao perfil de especialização de Criptografia e Segurança da Informação, lecionada no 2º Semestre do 4º ano do Mestrado Integrado em Engenharia Informática.

A temática abordada neste projeto enquadra-se nas tecnologias de identificação, mais propriamente, nas tecnologias de autenticação e identificação via assinaturas digitais especializadas. Especificamente, são utilizadas e abordadas as assinaturas *eIDAS* (*electronic IDentification, Authentication and trust Services*), um serviço de assinaturas digitais qualificadas definido nos regulamentos da união europeia.

O presente trabalho possui dois objetivos principais:

- A configuração e disponibilização de um ambiente de demonstração do DSS.
- A disponibilização das funcionalidades de assinatura, extensão de assinatura e validação da mesma, com base nos *web services* REST/SOAP disponibilizados pelo DSS, utilizando para o efeito uma de 5 alternativas:
 - Aplicação de linha de comando.
 - Aplicação desktop.
 - *nextcloud*.
 - *owncloud*.
 - *OpenKM*.

É expectável que, após a conclusão do trabalho, seja possível utilizar a demonstração que foi configurada em conjunto com as funcionalidades disponibilizadas numa das alternativas previamente mencionadas, num esquema que permita assinar um documento ou estender ou validar uma assinatura.

2 Solução

Esta secção pretende apenas dar, de certa forma, o início à implementação do trabalho, ao evidenciar a alternativa que será escolhida para disponibilizar as funcionalidades necessárias, bem como indicar as fases pela qual o projeto passa, para cumprir com os objetivos.

A solução idealizada passa por quatro fases:

- Encontrar uma ferramenta/opção que permita configurar e disponibilizar a aplicação web de demonstração do DSS, de tal forma que possibilite a sua utilização localmente.
- Utilizar a ferramenta para executar a demonstração localmente.
- Verificar qual será a melhor solução em termos de linguagem de programação para codificar as funcionalidades necessárias, sendo que a alternativa escolhida é uma aplicação de comando linha.
- Utilizando a linguagem de programação escolhida, implementar as funções de controlo e comunicação necessárias, para disponibilizar as funcionalidades pedidas.

Na secção seguinte (Implementação), serão abordadas as ferramentas escolhidas, para seguir com o projeto, bem como a configuração efetiva do ambiente local e a codificação da aplicação, que irá fazer de ponte de comunicação entre o utilizador e o DSS.

3 Implementação

Esta secção encontra-se dividida em duas sub-secções, sendo que o conteúdo a abordar está repartido da seguinte forma:

- Na primeira, abordar a ferramenta e os mecanismos que permitiram configurar e disponibilizar localmente o ambiente de demonstração do DSS.
- Na segunda, abordar a linguagem de programação e as ferramentas escolhidas que permitiram construir a aplicação de linha de comando que faz a ponte de comunicação entre o utilizador e o DSS.

3.1 Configuração do ambiente de demonstração

Com o intuito de disponibilizar um ambiente de demonstração online, o grupo decidiu, como sugerido, instalar o serviço numa máquina virtual, de maneira a que este ambiente não tenha qualquer interferência com o *host*, uma vez que se encontra isolado.

Contudo, achou-se por bem automatizar este processo, de maneira a diminuir o esforço necessário para configurar o sistema. Com isto em mente, optou-se por usar o *Vagrant*, que é uma ferramenta que permite configurar e gerir ambientes virtualizados, no mesmo *workflow*. Esta ferramenta está disponível para todos os sistemas operativos usuais (Windows, MAC OS e Linux).

Desta forma, e com base na documentação da ferramenta, foi possível desenvolver o ficheiro de configuração *Vagrantfile*, disponibilizado em anexo (ver anexo A). Este permite, quando executado o comando `vagrant up`, criar, se ainda não tiver sido criada, e iniciar uma máquina virtual, cujo o provisionamento é descrito no ficheiro de configuração. Este instala a aplicação de demonstração do DSS e torna-a disponível, sempre que a máquina é iniciada.

Todavia, é necessário ter em conta que esta ferramenta disponibiliza uma série de imagens de sistemas operativos, de modo a tirar partido de uma das maiores vantagens de um ambiente virtualizado. Para este caso, foi utilizada a imagem disponibilizada como *ubuntu/xenial64*, imagem esta que deve ser previamente instalada com o comando `vagrant box add ubuntu/xenial64`.

Concluindo, a utilização do *Vagrant* permite-nos provisionar uma máquina virtual, contendo o ambiente de demonstração web do DSS, que é acessível através de um browser segundo um IP pré-definido. Este acesso ao ambiente de demonstração será abordado na secção do Manual de Utilização.

3.2 Disponibilização de Funcionalidades do DSS

Relativamente à aplicação que irá permitir ao utilizador comunicar com o DSS, tal como foi dito anteriormente, esta consiste numa série de comandos que são interpretados, de forma a disponibilizar algumas das funcionalidades do DSS. A linguagem de programação utilizada para codificar esta aplicação de comando linha foi o **Python**.

Para além disso, este fornece uma série de serviços web, que se encontram na base da comunicação entre a aplicação desenvolvida e a demonstração. Estes encontram-se disponíveis como serviços *REST* e são os seguintes:

- Serviço de assinatura de um documento;
- Serviço extensão de assinatura de um documento;
- Serviço de validação da assinatura de um documento.

Apesar de existirem mais serviços disponibilizados, estes são os mais relevantes, para este caso de estudo.

Posto isto, foram utilizados os seguintes módulos **Python**, de maneira a simplificar a codificação da aplicação cliente:

- **requests** que permite realizar pedidos HTTP/1.1 de forma simples e eficaz;
- **argparse** que simplifica a escrita de interfaces comando linha;
- **menu** que é análogo, de certa forma, ao **argparse**, permitindo escrever menus para aplicações comando linha;
- **json** que permite realizar operações com tipos de dados *JSON*, que serão utilizados na definição de parâmetros nos pedidos HTTP.

A funcionalidade de assinatura não é disponibilizada na aplicação de comando linha criada, por indicação do docente, devido ao facto de também ter que se construir uma interface para a utilização do cartão de cidadão para assinar digitalmente os documentos.

Assim sendo, a aplicação de comando linha criada permite a extensão da assinatura de um documento, bem como a validação da(s) assinatura(s) digita(l/is) presente(s) no documento. Esta está implementada de tal forma que torne a utilização interativa, isto é, as opções que o utilizador pode escolher num dado momento são mostradas, para facilitar a utilização da ferramenta.

O ponto de entrada e principal da aplicação é um menu simples de terminal que permite a interação com a mesma, possibilitando a execução de uma operação de extensão de assinatura ou uma operação de execução de validação de assinatura(s).

Os pedidos que são feitos ao *web service* pela aplicação utilizam o ficheiro *json* de pedido que se encontra na documentação do DSS, sendo que os parâmetros que são necessários alterar, são alterados de acordo com a informação inserida pelo utilizador. Estes ficheiros json são lidos da diretoria *application/json/*.

Após iniciada, o fluxo de execução da aplicação é sempre o seguinte:

1. Escolha da operação a executar (extensão ou validação).
2. Inserção da informação necessária, consoante a operação pedida, de forma interativa.
3. Visualização de mensagem de erro, caso existam, ao inserir a informação.
4. Receção da resposta relativamente ao pedido efetuado, no ecrã no caso da validação e gravada num ficheiro no caso da extensão.
5. Voltar ao ponto 1.

Na secção de resultados, serão apresentados *screenshots* que permitem visualizar as interações existentes com o utilizador.

4 Manual de Utilização

Primeiramente, é conveniente referir que a aplicação de comando linha desenvolvida comunica com o web service local configurado na máquina virtual, utilizando o endereço:

10.0.0.101:8080

Como já foi previamente referido, para executar a máquina virtual com a demonstração, é necessário estar situado na diretoria onde se encontra o *VagrantFile* (raíz) e executar o comando:

```
vagrant up
```

Caso não seja possível instalar a ferramenta vagrant, o endereço de comunicação nos pedidos feitos na aplicação Python deve ser alterado para o endereço correspondente ao local onde se encontra a demonstração do DSS a correr na máquina que está a utilizar.

Para a aplicação cliente em linhas de comando, foram desenvolvidos dois tipos de interação com o utilizador. O primeiro, permite-lhe invocar o comando, que executa o programa, sem argumentos, apresentando-lhe um menu, com o qual poderá exercer ações, conforme os seus objetivos. De seguida, encontra-se a representação do menu referido.

DSS : Digital Signature Service

1. Extend a signature
2. Validate a signature
3. Close

>>>

Intuitivamente, o utilizador poderá seleccionar qualquer uma das opções fornecidas, sendo-lhe, de seguida, solicitados os vários parâmetros, que permitem concluir a ação pretendida. É importante notar que a aplicação verifica a validade dos mesmos, sendo que, caso esta não se verifique, o utilizador será informado sobre a utilização correta.

O segundo modo de utilização permite, juntamente com o comando de invocação do programa, a passagem de certos argumentos, que disputam as mesmas ações que o primeiro modo de utilização. Desta forma, a comunicação dos parâmetros também é feita através de argumentos, e a sua validação também é tratada, sendo que, qualquer violação desta, também será dada a conhecer ao utilizador. Posto isto, o seu manual de utilização pode ser apresentado com a passagem do argumento `-h`, `--help`, sendo-lhe representado o seguinte:

```
usage: dss-client.py [-h] {extendDocument,validateSignature} ...
```

DSS : Digital Signature Service

optional arguments:

`-h, --help` show this help message and exit

service:

`{extendDocument,validateSignature}`

rest web service

`extendDocument` the method allows to extend an existing signature to a stronger level

`validateSignature` this service allows to validate signature (all formats/types) against a validation policy.

Posto isto, foi também implementado um manual de utilização para as diferentes ações, que pode ser invocado com a passagem do serviço em questão como argumento e o mesmo visto em cima (`-h`, `--help`). De seguida, encontram-se representados estes manuais, para os diferentes serviços.

```
usage: dss-client.py extendDocument [-h] --signed-file signed-file
                                     [--container container]
                                     --sig-format sig-format --level level
```

optional arguments:

`-h, --help` show this help message and exit

`--signed-file signed-file, -f signed-file`
signed file

`--container container, -c container`
container

`--sig-format sig-format`
signature format

`--level level, -l level`
level

```
usage: dss-client.py validateSignature [-h] --signed-file signed-file
```

```
--original-file original-file
```

optional arguments:

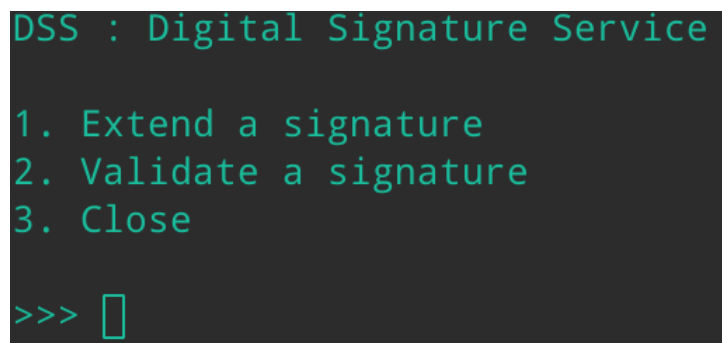
```
-h, --help            show this help message and exit
--signed-file signed-file, -f signed-file
                        signed file
--original-file original-file, -o original-file
                        original file(s)
```

5 Resultados

Nesta secção pretendemos mostrar alguns *screenshots* que evidenciam os resultados obtidos neste trabalho, pelo grupo.

Com a implementação referida na secção anterior, conseguimos atingir os objetivos propostos que passavam por conseguir estender uma assinatura e validar uma assinatura, sendo que a assinatura de um documento em si, por motivos já abordados neste relatório, ficou fora do âmbito deste projeto.

Após iniciar a máquina virtual com a demonstração do DSS e, posteriormente, iniciar a aplicação de comando linha *dss-client.py*, será apresentado no terminal o menu que pode ser visto na figura 1, que permite a escolha (numérica) entre estender uma assinatura de um documento, validar uma assinatura, ou simplesmente sair da aplicação.



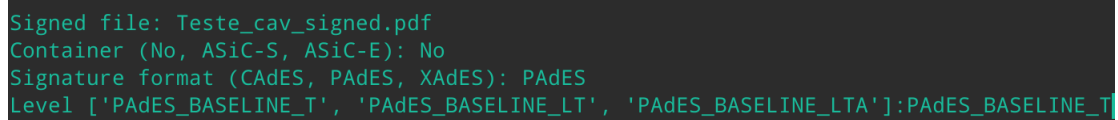
```
DSS : Digital Signature Service

1. Extend a signature
2. Validate a signature
3. Close

>>> █
```

Figura 1: Menu inicial da aplicação desenvolvida

Ao escolher a opção para estender uma assinatura, terá que ser necessário inserir a informação, de forma interativa, sobre o tipo de assinatura a estender, como o *container* que usa, bem como o formato da assinatura e o nível da mesma além de, claro, inserir o caminho para o ficheiro assinado, para que possa ser estendido. Esta troca de informação interativa pode ser visualizada na figura 2.



```
Signed file: Teste_cav_signed.pdf
Container (No, ASiC-S, ASiC-E): No
Signature format (CAvES, PAdES, XAdES): PAdES
Level ['PAdES_BASELINE_T', 'PAdES_BASELINE_LT', 'PAdES_BASELINE_LTA']:PAdES_BASELINE_T
```

Figura 2: Informação submetida ao estender um documento

Após inserir a informação necessária, é impresso na consola o código de resposta HTTP, bem como uma informação indicando que o documento com a assinatura estendida foi gravado na diretoria indicada, como mostra a figura 3. Um exemplo de um ficheiro com a assinatura estendida também pode ser visto na figura 4. Este ficheiro foi submetido junto com o código e este relatório, pelo que se encontra na diretoria *application/*.

```
200
Document with extended signature saved in application/extends_resp.pdf
press enter to continue
```

Figura 3: Resposta à operação de extensão de um documento

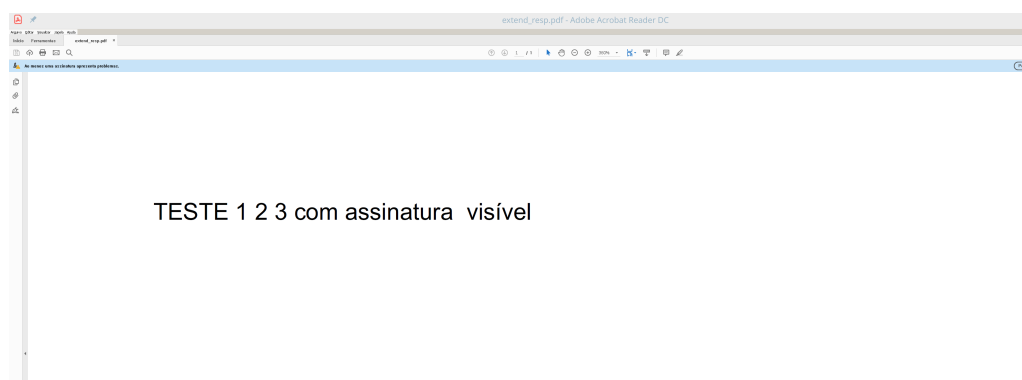


Figura 4: Ficheiro com assinatura estendida

Assim conclui a interação do utilizador com a aplicação para conseguir estender uma assinatura de um documento.

De seguida, pretendemos evidenciar o que acontece quando é executada a operação de validação da assinatura de um documento.

Ao escolher a opção para executar a operação de validação da assinatura, será necessário indicar à aplicação o caminho para o ficheiro e para o ficheiro original (opcional), tal como indica a figura 5.

```
Signed file: Teste_cav_signed.pdf
Original file: Teste_cav.pdf
```

Figura 5: Informação inserida ao executar uma operação de validação de assinatura

Após a execução do pedido, é impressa na linha de comandos a resposta recebida quanto à validação da assinatura do documento pedido, tal como evidencia a figura 6.

```
-----
Validation results for signature Teste_cav_signed.pdf:
Validation status: TOTAL_PASSED
Validation status description: Qualified Electronic Signature
File signed by: JOSÉ EDUARDO PINA DE MIRANDA

CERTIFICATE CHAIN:
Certificate 1:
  ID: 2BEC003CF0E34ADBB680D6382DB43883AC5AA8755937EA23DA7AA7C4F902BBBD
  Entity Name: JOSÉ EDUARDO PINA DE MIRANDA
Certificate 2:
  ID: 119F0FE4C3CE621A53B6C4F4FF5611A1C9250C4F7BA3DE45F4855FF29F4C4615
  Entity Name: EC de Chave Móvel Digital de Assinatura Digital Qualificada do Cartão de Cidadão 00001

WARNINGS:
Warning 1
  The trusted certificate doesn't match the trust service
Warning 2
  Authority info access is not present!
-----
press enter to continue

```

Figura 6: Resposta recebida após operação de validação

Assim conclui a demonstração dos resultados obtidos com a criação da aplicação de comando linha, de onde podemos retirar que, de facto, conseguimos implementar tanto a extensão de assinatura como a validação de uma assinatura no documento.

6 Conclusão

Terminada a resolução deste trabalho prático, é da nossa opinião que os resultados obtidos são bastante satisfatórios, visto que os dois objetivos inicialmente propostos foram cumpridos em pleno, com exceção da assinatura de um documento que foi um dos objetivos que foi retirado do âmbito da realização deste projeto, posteriormente.

A maior dificuldade sentida pelo grupo está relacionada com o facto de que a documentação existente em relação aos *web services* não evidenciava claramente os parâmetros que eram necessários passar aos métodos da API, pelo que foi necessária uma pesquisa mais extensa pelo exemplo de um pedido de forma a tentar perceber que tipo de parâmetros seriam necessários.

A partir do momento que conseguimos ultrapassar a dificuldade referida, na nossa opinião, conseguimos produzir resultados bastante satisfatórios que conseguiram cumprir com o que nos propusemos inicialmente.

A Vagrantfile

```
PUBLIC_KEY = File.read(File.expand_path('~/.ssh/id_rsa.pub')).strip

Vagrant.configure("2") do |config|

  config.vm.box = "ubuntu/xenial64"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "512"
    vb.cpus = 1
  end

  config.vm.provision "shell", inline: <<-SHELL
    echo "#{PUBLIC_KEY}" >> /home/ubuntu/.ssh/authorized_keys
    apt-get -y update
    apt-get -y upgrade
    apt-get -y autoremove
    apt-get install -y vim wget openjdk-8-jdk unzip
    wget https://ec.europa.eu/cefdigital/artifact/repository/
      esignaturedss/eu/europa/ec/joinup/sd-dss/dss-demo-bundle
      /5.4/dss-demo-bundle-5.4.zip
    unzip dss-demo-bundle-*
    chmod +x dss-demo-bundle-5.4/apache-tomcat-8.5.35/bin/*.sh
  SHELL

  config.vm.provision "shell", inline: ". /dss-demo-bundle-5.4/
    apache-tomcat-8.5.35/bin/startup.sh", run: 'always'

  config.vm.define "dss" do |dss|
    dss.vm.network "private-network", ip: "10.0.0.101"
  end
end
```