1- Assinaturas cegas (Blind signatures) baseadas no Elliptic Curve Discrete Logarithm Problem (ECDLP)

Pergunta 1.1

Nesta pergunta pretende-se que se altere o código fornecido anteriormente de forma a que o *input* e o *output* sejam alterados.

No ficheiro **init-app.py** está o que é necessário para que o assinante desenvolva o processo de inicialização, isto é, encontra-se as diferentes formas sobre como inicializar uma assinatura cega.

No ficheiro **blindSignature-app.py** contém as alterações necessárias para que apenas mostre o resultado da assinatura, sendo que vai buscar os componentes necessários para a sua criação a um ficheiro pré-definido.

No ficheiro **ofusca-app.py** tem aquilo que foi pedido, isto é, devolve a mensagem cega e guarda num ficheiro as componentes Blind components e pRComponents.

Ao correr o ficheiro **desofusca-app.py** recebemos como *output* a assinatura, e para a sua criação vamos a um ficheiro buscar.

Finalmente, ao correr o ficheiro **verify-app.py** obtemos a informação que nos diz se a mensagem é ou não válida.

Na figura 1 temos o resultado de correr estes ficheiros no terminal, e tal como podemos observar, o *output* de cada um destes ficheiros é menor.

```
user@CSI:~/1718-G3/Aula3/perg1$ python init-app.py
pRDashComponents:
65f4c009b0a6622b9a831a040a476f175041f9114bb0b05a07058c52c9981b43.851448ff7910c58a677846efbe14c18267c215bd89cb625eec8b66016a15
37be
user@CSI:~/1718-G3/Aula3/perg1$ python ofusca-app.py -msg ola 1234 -RDash 65f4c009b0a6622b9a831a040a476f175041f9114bb0b05a070
58c52c9981b43.851448ff7910c58a677846efbe14c18267c215bd89cb625eec8b66016a1537be
b2b8cb6011325cd32671d1c0aa9f3803181b79fc1ae4b6616f1a43f497d125dd
user@CSI:~/1718-G3/Aula3/perg1$ python
blindSignature-app.py  desofusca-app.py       init-app.py           ofusca-app.py          verify-app.py
user@CSI:~/1718-G3/Aula3/perg1$ python blindSignature-app.py -key
blindSignature-app.py  init-app.py           key.pem               ofusca-app.py          xcomponents.data
desofusca-app.py       key.cert              message.components    verify-app.py
user@CSI:~/1718-G3/Aula3/perg1$ python blindSignature-app.py -key key.pem -bmsg b2b8cb6011325cd32671d1c0aa9f3803181b79fc1ae4b
6616f1a43f497d125dd
578f9c003746b8924a6f7b2b3ec185ec5707d0b367d862fdbaf106daaac770f44edec0145d90e474f2c0f6e2ba8ed5a898a05d748c6e29e69ff0bf897ee61
803
user@CSI:~/1718-G3/Aula3/perg1$ python desofusca-app.py -s 578f9c003746b8924a6f7b2b3ec185ec5707d0b367d862fdbaf106daaac770f44e
dec0145d90e474f2c0f6e2ba8ed5a898a05d748c6e29e69ff0bf897ee61803 -RDash 65f4c009b0a6622b9a831a040a476f175041f9114bb0b05a07058c5
2c9981b43.851448ff7910c58a677846efbe14c18267c215bd89cb625eec8b66016a1537be
d70781831755c067dbf4e1b8afac9327406719259a483c389a0424ca23ca9452
user@CSI:~/1718-G3/Aula3/perg1$ python verify-app.py -cert key.pem -msg ola 1234 -sDash d70781831755c067dbf4e1b8afac932740671
9259a483c389a0424ca23ca9452 -f
blindSignature-app.py  desofusca-app.py       init-app.py           ofusca-app.py          verify-app.py
user@CSI:~/1718-G3/Aula3/perg1$ python verify-app.py -cert key.pem -msg ola 1234 -sDash d70781831755c067dbf4e1b8afac932740671
9259a483c389a0424ca23ca9452
blindSignature-app.py  init-app.py           key.pem               ofusca-app.py          xcomponents.data
desofusca-app.py       key.cert              message.components    verify-app.py
user@CSI:~/1718-G3/Aula3/perg1$ python verify-app.py -cert key.pem -msg ola 1234 -sDash d70781831755c067dbf4e1b8afac932740671
9259a483c389a0424ca23ca9452 -f message.components
Output
Valid signature
user@CSI:~/1718-G3/Aula3/perg1$ cd
```

Figura 1: Resultado, no terminal, da criação e verificação de uma assinatura cega.

2- Protocolo SSL/TLS

Pergunta 2.1

Nesta pergunta pretende-se que tendo por base três universidades não europeias se efetue o teste SSL de modo a que para cada uma se anexe os resultados do SSL Server test, relativamente ao site com pior rating que se analise o resultado do SSL Server test e que por fim se explique o significa "DNS CAA"e os seus efeitos práticos.

As três universidades escolhidas são:

- MIT, Universidade dos Estados Unidos;

- U. Alberta, Universidade do Canadá;
- UNAM, Universidade do México.

Na diretoria, encontram-se três ficheiros *.pdf* que contém os resultados do SSL Server test de cada uma destas universidades.

Analisando os resultados, vê-se que o pior resultado é da universidade do México que obteve classificação F, ao passo que as outras têm classificação de A. O servidor universidade do México apresenta muitos problemas, por exemplo, uma vulnerabilidade é ao OpenSSL Padding Oracle (CVE-2016-2107), outra deve-se ao facto deste servidor aceitar cifras RC4 com protocolos antigos que o tornam mais vulnerável, por fim não suporta Foward Secrecy, isto é, não suporta protocolos de comunição segura com os browsers. Dentro dos Cipher Suites, muitos protocolos são tidos como inseguros ou fracos, devido ás vulnerabilidades referidas anteriormente. Desta forma, e uma vez que estas vulnerabilidades ainda não foram corrigidas, considera-se que este servidor é inseguro e que devia ser retificado com urgência.

No campo "*DNS CAA*"todas as universidades apresentam um No. Este campo é para uma proposta que visa a melhorar os ecossitemas PKY no sentido em que lhes dão o controlo para restringir quais as CAs que podem emitir certificados para um determinado domínio.

3- Protocolo SSH

Pergunta 3.1

Para esta pergunta é nos pedido para escolher dois servidores ssh de universidades não europeias. Assim, e tendo em conta as universidades escolhidas acima, escolhemos:

- MIT, Universidade dos Estados Unidos;
- U. Alberta, Universidade do Canadá.

Na diretoria, encontra-se dois ficheiros *.txt* que contém os resultados do ssh-audit a cada uma destas universidades.

O software usado pelos dois servidores ssh destas universidades é o OpenSSH, sendo que no MIT a versão é 6.7p1 e a da UAlberta é a 5.3.

Dentro de estas duas universidades, a versão de software tem mais vulnerabilidades é a 5.3, uma vez que esta apresenta 10 vulnerabilidades e a outra versão apenas 5.

A versão que apresenta a vulnerabilidade mais grave (de acordo com o CVSS score identificado no CVE details) é a versão 6.1p1. Esta vulnerabilidade é a CVE-2016-8858 com um cvss score de 7.8. Enquanto que, as duas vulnerabilidades mais graves da versão 5.3 têm ambas 7.5 de cvss score (CVE-2014-1692 e CVE-2010-4478).

A vulnerabilidade indicada no ponto anterior é grave, pois esta permite ataques do tipo denial of service, através do envio duplicado de *KEXINIT requests*, em termos práticos isto torna o serviço indisponível(em caso de ataque bem sucedido), podendo por isso ser considerada grave.