1.

Esquema MAC (Message Autentication Code):
O esquema MAC assegura a Autenticidade.
É um esquema de natureza da chave Simetrico.

Esquema Assinatura Digital:
O esquema Assinatura Digital assegura a Confidencialidade.
É um esquema de natureza de chave Assimetrica.

A escolha do esquema depende da intenção de envio da mensagem.
Se queremos enviar a mensagem e garantir que não é alterada por terceiros, então devemos escolher o esquema MAC. No entanto, a mensagem é livre de ser lida.
Se queremos enviar a mensagem e garantir que não consegue ser lida por terceiros, então devemos escolher o esquema Assinatura Digital. No entanto, a mensagem é livre de ser alterada sem haver forma de detetar a sua alteração.

2.

Dado: M = [HELL][OWOR][LD!I][SEL!] (HELLOWORLD!ISEL!)
Temos: k = 4; b = 4;
H(M) = Y4 = Ep(LD!I)(SEL!)

Dado: M' = [JFQI][ACIR][LD!I][SEL!] (JFQIACIRLD!ISEL!)
Temos: k = 4; b = 4;
H(M') = Y4 = Ep(LD!I)(SEL!)

Então: M != M' && H(M) == H(M')
Logo, para mensagens diferentes, são gerados as mesmas hashes.
Basta obter os ultimos dois blocos da mensagem para criar hashes iguais com mensagens diferentes.

3.

Uma forma de atacar uma implementação de um esquema de cifra assimétrica com o algoritmo de cifra apresentado, é interceptando o pedido de chave pública que X pede a Y, fazendo assim com que seja devolvido para X a chave pública que nós temos e que irá ajudar a atacar a mensagem.
Com este sucedido, X vai pensar que tem a chave de Y e realizar a encriptação da sua mensagem que irá de seguida enviar. Com isto a acontecer, iremos então interceptar a mensagem enviada juntamente com a nossa chave privada e conseguindo então assim interpretar a mensagem que deveria chegar a Y.

4.

Os criptogramas produzidos pelo A são mais difíceis de criptoanálisar devido à enorme quantidade de tempo que seria necessário para cifrar e
decifrar, para além que são muito mais vulneráveis que o AES, como o DES tem apenas chaves com 56 bits, faz com que o sistema demore mais tempo
na sua análise fazendo igualmente o processo mais lento. No entanto os criptogramas de B são melhores devido à quantidade maior de bits
a serem usados nas suas chaves.

5.

As engine classes da biblioteca JCA trabalham com os dados em arrays de byte[ ]. Todo o input e output de dados é feito neste tipo primitivo. Isto permite com que o output de uma engine class seja utilizado como input em outra engine class, realizando assim a aplicação incremental das respectivas protecções.

6.

6.1.

A resistência à segunda pré-imagem de uma função hash contribui a garantir a autenticidade da chave pública de um certificado, porque dada uma mensagem que seja recebida, é difícil de encontrar outra em que o hash da primeira seja igual ao da segunda. Caso isto não se suceda, ficam muito vulneráveis a ataques de segunda pré-imagem. Conseguindo este ataque, basta descobrir a hash e saber que será sempre coeso mediante a mensagem, que podem fazer os ataques à vontade porque já é de conhecimento o código hash.

6.2.

As chaves privadas dos intermédios são usadas para validar o certificado C, no exemplo de uma empresa, tendo o certificado C como um como o certificado principal do domínio, irá depois ser o principal na hierarquia da corrente, o que irá fazer com que os intermédios abaixo são inseridos como seguros, sendo eles validados com o certificado "mãe" dizendo que são seguros no domínio.

7.

Na alínea 1, foi realizado o seguinte:

- Criou se um ficheiro de texto com a frase “Hello World!”;
- Aplicou-se um algoritmo de encriptação;
- Aplicou-se a desencriptação sem remover o padding;
- Observou-se o padding no novo ficheiro;
- Repetiu se para todos os outros algoritmos.

Concluiu-se que os algoritmos ECB e CBC apresentam padding, enquanto que os algoritmos CFB e OFB não apresentam padding.

Na alínea 2, encriptou-se três ficheiros diferentes com 128-bit AES em modo CBC. Cada um dos ficheiros tinha 5, 10, e 16 bytes, respetivamente.

De seguida, cada ficheiro foi desencriptado sem remover o padding, e observou-se o seguinte resultado:

```
hexdump -C 5b_dec.txt
00000000  31 32 33 34 35 0b 0b 0b  0b 0b 0b 0b 0b 0b 0b 0b  |12345...........|
00000010
```

```
hexdump -C 10b_dec.txt
00000000  31 32 33 34 35 36 37 38  39 41 06 06 06 06 06 06  |123456789A......|
00000010
```

```
hexdump -C 16b_dec.txt
00000000  31 32 33 34 35 36 37 38  39 41 42 43 45 46 47 48  |123456789ABCEFGH|
00000010  10 10 10 10 10 10 10 10  10 10 10 10 10 10 10 10  |................|
00000020
```

Podemos concluir que o ficheiro com 16 bytes não necessita de padding, pois corresponde a 256 bits, que é múltiplo de 126, que é o necessário para o tamanho do bloco.

8.

Para um demo de utilização do programa MAC.jar, executar:

java -jar MAC.jar ./HelloWorld.txt

9.

Para um demo de utilização do programa CipherDecipher.jar, executar:

java -jar CipherDecipher.jar -m cipher -f ./serie1-1920i.pdf -fd ./serie1-1920i.enc -c ./certificates-keys/cert-end.entities/Alice_1.cer -k ./key.out

java -jar CipherDecipher.jar -m decipher -f ./TestFileOut.enc -fd ./TestFileOut.txt -c ./certificates-keys/pfx/Alice_1.pfx -k ./key.out -p changeit

Diogo Martins Nº39185
João Matos Nº42390
SI - Grupo 16 - 1920i