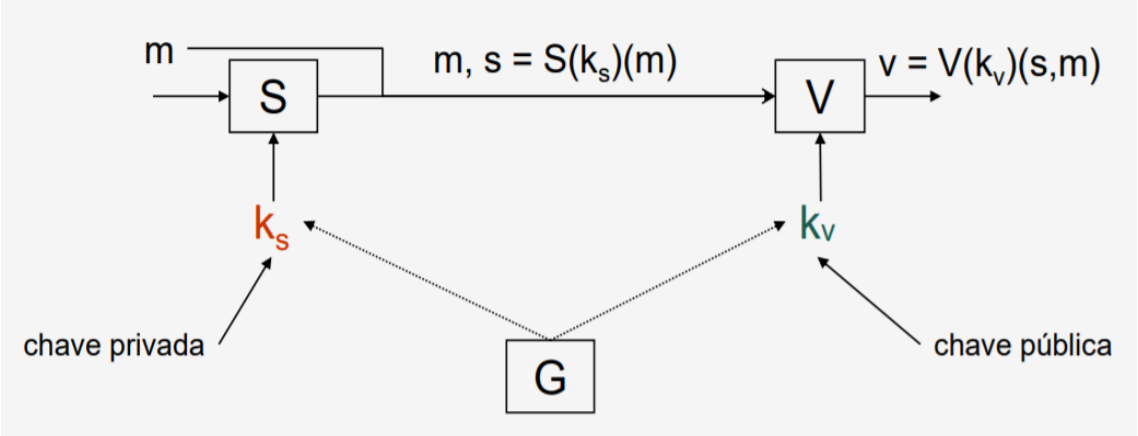
Segurança Informática

Licenciatura em Engenharia Informática, Redes e Telecomunicações

1ª Série de Exercícios



Grupo 4

43874 João Florentino

46435 Mihail Ababii

46919 Bárbara Castro

1º Semestre - Ano letivo 2020/2021

11 de novembro de 2020

Resumo

Este trabalho incide sobre o funcionamento de esquemas simétricos, cifras com chave simétrica, autenticação com este tipo de chave e as suas respetivas propriedades de segurança, funções de *hash* criptográficas (resistência a colisões e resistência à segunda pré-imagem), chaves, modos de operação, *padding*, dimensão de blocos e chaves, modos ECB e CBC, *initial vector*, modo de operação em *stream*, paralelização, cifras autenticadas e certificados.

Palavras-chave: assinatura digital, ataques, autenticação, certificados, chave privada, chave pública, criptografia, esquema simétrico, função de *hash*, modo CBC, modo ECB, *padding*, pré-imagem

Índice

Lista de Figuras

1. Introdução

Na realização deste trabalho, foi-nos proposto que estudássemos modos de operação, esquemas criptográficos com utilização de cifra simétrica, certificados digitais e infraestruturas de chaves publicas e por último que implementássemos uma aplicação recorrendo à biblioteca java JCA.

Numa primeira parte foi-nos pedido que estudássemos os modos de operação, em concreto, quais as vantagens de se usar certos modos de operação em relação a outros e verificar quais destes usavam *padding.* Apos isto foi-nos pedido que verificássemos, a partir de uma função, como poderíamos realizar uma quebra na integridade de um esquema *hash*. De seguida realizamos um estudo sobre as vantagens de implementação incremental de proteções e como estas poderiam ser usadas na biblioteca JCA. Por último foi-nos pedido que realizássemos uma aplicação que iria cifrar e decifrar um documento ou ficheiro e outra que iria servir para assinar e verificar a integridade de um documento assinado.

Ferramentas utilizadas para o desenvolvimento do trabalho

No desenvolvimento do trabalho aqui descrito, recorremos à ferramenta "*openssl*", incluídas na máquina virtual que nos foi disponibilizada pelo docente da cadeira, onde pudemos simular a encriptação e desencriptação de mensagens com diferentes algoritmos e modos. Para a resolução das duas últimas perguntas, recorremos à utilização da linguagem de programação JAVA ("*Java Cryptographic Architecture"*).

2. Resolução dos problemas do trabalho

Problema 1:

No contexto dos esquemas de cifra simétrica, apresente duas vantagens do modo de operação CTR (counter mode) em relação ao modo CBC:

CBC (cipher-block chaining), é um modo de operação que serve para cifrar uma mensagem ou um documento, através de blocos. Este modo de operação aplica a cada bloco de texto simples uma função XOR junto com o bloco cifrado anterior antes do texto ser criptografado. Além disso para que cada mensagem seja única, mesmo sendo de um mesmo texto, um vetor de inicialização deve ser utilizado no primeiro bloco, preferencialmente gerado aleatoriamente.

CTR (Counter mode), à semelhança do CBC, é também um modo de operação que serve para cifrar uma mensagem ou documento, através de blocos. Este modo de operação aplica uma função XOR com um contador criptografado, contador esse que ira ser incrementado para cada bloco subsequente.

As diferenças entre os dois modos são ao nível da propagação de erros. Enquanto que em CBC como o próximo bloco depende do anterior, se existir um erro no primeiro bloco, este erro ira ser propagado para o próximo, enquanto que em CTR isso não ira acontecer pois são blocos independentes. Também existe uma diferença ao nível de decifra, a qual em CTR poderá ser feita sem ordem especifica, devido aos blocos serem independentes. Isto já não se verifica em CBC pois a reordenação dos blocos cifrados afeta a decifra.

Problema 2:

Considere um novo esquema criptográfico AE. O objetivo é fazer uma cifra simétrica com garantias de integridade, ou seja, caso os criptogramas sejam modificados no canal de comunicação, tal seria detetado pelo destinatário.

As funções AEe e AEd realizam a cifra e decifra autenticada, sendo E uma primitiva de cifra simétrica, H uma função de hash criptográfica e jj a concatenação de bytes.

AEe(k)(m) = E(k)(m) jj H(E(k)(m))

AEd(k)(c; h) = (se H(c) == h então m = D(k)(c) senão falha de integridade)

Note que a função de decifra opera sobre criptogramas (c) e o valor de hash (h) que foram colocados no canal de comunicação pela função de cifra.

Descreva de que forma pode ser comprometida a propriedade de integridade do esquema.

Uma vez que a função de hash é uma função finita, um atacante poderia usar um ataque de segunda pré imagem. Este ataque consiste em ser capaz de encontrar um input específico que gera um mesmo resultado de output, neste caso um hash igual ao do ficheiro original que a vítima queria enviar. Desta forma, o atacante consegue geral uma assinatura igual, contudo sendo um ficheiro diferente, comprometendo então a integridade do sistema.

Problema 3:

Na biblioteca JCA, como é que as engine classes (ex: Cipher, Signature, Mac) possibilitam a aplicação incremental das respetivas proteções? Qual a vantagem de aplicar proteções incrementalmente?

A biblioteca JCA possibilita a aplicação incremental das respetivas proteções através do método update e do método dofinal. O método update recebe parte da mensagem e retorna parte do criptograma, enquanto o método dofinal é responsável por receber a parte final da mensagem e retornar o final do criptograma. Estes métodos são providenciados durante a execução do método init.

A vantagem de se usar aplicação incremental de proteções é que desta maneira garantimos um aumento de segurança e espaço de memoria, por exemplo, quando transferimos um ficheiro de grande dimensão, este ficheiro fica armazenado em memoria. Ao aplicar proteções incrementais, à medida que o ficheiro é transferido, vão sendo aplicadas estas proteções, que deste modo garantem uma redução do espaço usado em memoria, pois, o ficheiro é transferido em blocos, e ainda aumenta a segurança pois o ficheiro não se encontra tanto tempo em plaintext.

Problema 4:

Considere os certificados digitais X.509 e as infraestruturas de chave pública:

4.1. A assinatura de um certificado folha tem em conta toda a cadeia de certificados?

A cadeia de certificados é tida em conta ao realizar a verificação de um dado certificado, no entanto, a assinatura de um certificado depende meramente dos próprios bytes que o constituem e da chave privada que lhe é passada, pelo que não tem em conta toda a cadeia de certificados.

4.2. Existem campos num certificado que estejam protegidos por um esquema de cifra (simétrica ou assimétrica)?

Não existem campos cifrados num certificado, devido a este apenas servir como prova de autenticidade, como tal, não é necessário a utilização de cifras (simétricas ou assimétricas) nos campos do mesmo.

4.3. Considere dois sistemas informáticos cujas comunicações estão cifradas usando cifra assimétrica, após troca de chaves públicas em certificados X.509. A realização de um ataque de man-in-the-middle implica conseguir alterar algo do lado cliente?

A frase anterior é falsa, pois um ataque man-in-the-middle implica conseguir intercetar e alterar a chave publica presente no canal de comunicação, não afetando nada no lado do cliente. Ao realizar este ataque o cliente ira receber a chave publica do atacante, deste modo, o cliente, ao transmitir informação esta será lida pelo atacante. Também pode ser considerado um ataque ao servidor pois este quando tenta trocar a chave com o cliente, na verdade está a trocá-la com o atacante.

Problema 5:

Considere o enunciado do laboratório "Crypto Lab" disponível em https://seedsecuritylabs.org/Labs_16.04/Crypto/Crypto_Encryption/. Realize a tarefa "Task 4: Padding". Descreva sucintamente os resultados.

Task 4 – Problema 1:

Para cifras de bloco, quando o tamanho de um *plaintext* não é múltiplo do tamanho do bloco, é necessário aplicar *padding*, que serve como forma de preencher o espaço não ocupado em cada bloco para assim o tamanho do bloco ser múltiplo do tamanho do *plaintext* em bytes. As experiências que se seguem têm como objetivo verificar como funciona o *padding* e em que modos de operação é aplicado. Foram utilizados os modos de operação ECB, CBC, CFB e OFB para encriptar um ficheiro de texto, com a cifra AES.

O ficheiro utilizado é um ficheiro de texto chamado "toEncrypt.txt" com o seguinte conteúdo:

"OLA TUDO BEM"

Para todas as vezes que encriptamos ficheiros utilizamos sempre o comando (mudando apenas o modo de operação e os ficheiros de saída):

"openssl enc -aes-256-<*Modo de Operação*> -e -in toEncrypt.txt -out toEncrypt<*Modo De Operacao*>.txt"

Modo ECB:

Conseguimos observar que este modo de operação utilizava *padding* através da realização da decifra do documento cifrado. Quando cifrado e de seguida decifrado observamos que este não tem o mesmo tamanho logo, foi acrescentado *padding*.

```
[10/31/20]seed@VM:~/.../ECB$ xxd toEncrypt.txt
00000000: 4f4c 4120 5455 444f 2042 454d 0a         OLA TUDO BEM.
[10/31/20]seed@VM:~/.../ECB$ xxd originalECBnoPad.txt
00000000: 4f4c 4120 5455 444f 2042 454d 0a03 0303  OLA TUDO BEM....
[10/31/20]seed@VM:~/.../ECB$
```

Figura 1-Ficheiro de texto e ficheiro depois de ser decifrado com ECB

Modo CBC:

Conseguimos observar que este modo de operação utilizava *padding* através da realização da decifra do documento cifrado. Quando cifrado e de seguida decifrado observamos que este não tem o mesmo tamanho logo, foi acrescentado *padding*.

```
[10/31/20]seed@VM:~/.../CBC$ xxd toEncrypt.txt
00000000: 4f4c 4120 5455 444f 2042 454d 0a         OLA TUDO BEM.
[10/31/20]seed@VM:~/.../CBC$ xxd originalCBCnoPad.txt
00000000: 4f4c 4120 5455 444f 2042 454d 0a03 0303  OLA TUDO BEM....
```

Figura 2-Ficheiro de texto e ficheiro depois de ser decifrado com CBC

Modo CFB:

Conseguimos observar que este modo de operação não utiliza *padding* através da análise da decifra do documento cifrado. Quando cifrado e de seguida decifrado conseguimos observar que o documento continua com o mesmo tamanho logo, não foi acrescentado *padding*.

```
[10/31/20]seed@VM:~/.../CFB$ xxd toEncrypt.txt
00000000: 4f4c 4120 5455 444f 2042 454d 0a         OLA TUDO BEM.
[10/31/20]seed@VM:~/.../CFB$ xxd originalCFBnoPad.txt
00000000: 4f4c 4120 5455 444f 2042 454d 0a         OLA TUDO BEM.
[10/31/20]seed@VM:~/.../CFB$
```

Figura 3-Ficheiro de texto e ficheiro depois de ser decifrado com CFB

Modo OFB:

Conseguimos observar que este modo de operação não utiliza *padding* através da análise da decifra do documento cifrado. Quando cifrado e de seguida decifrado conseguimos observar que o documento continua com o mesmo tamanho logo, não foi acrescentado *padding.*

```
[10/31/20]seed@VM:~/.../OFB$ xxd toEncrypt.txt
00000000: 4f4c 4120 5455 444f 2042 454d 0a         OLA TUDO BEM.
[10/31/20]seed@VM:~/.../OFB$ xxd originalOFBnoPad.txt
00000000: 4f4c 4120 5455 444f 2042 454d 0a         OLA TUDO BEM.
[10/31/20]seed@VM:~/.../OFB$
```

Figura 4-Ficheiro de texto e ficheiro depois de ser decifrado com OFB

Task 4 – Problema 2:

Neste problema, criaram-se três ficheiros com 5 *bytes*, 10 *bytes* e 16 *bytes*, respetivamente. Recorrendo ao comando "echo -n" para criar estes ficheiros, e utilizamos os seguintes comandos:

1º ficheiro: echo -n "12345" > f1.txt

openssl enc -aes-128-ecb -d -in f1.txt -out df1.txt

openssl enc -aes-128-ecb -d **-nopad** -in df1.txt -out vf1.txt

2º ficheiro: echo -n "1234567890" > f2.txt

openssl enc -aes-128-ecb -d -in f2.txt -out df2.txt

openssl enc -aes-128-ecb -d **-nopad** -in df2.txt -out vf2.txt

3º ficheiro: echo -n "1234567890abcdef" > f3.txt

openssl enc -aes-128-ecb -d -in df1.txt -out df1.txt

openssl enc -aes-128-ecb -d **-nopad** -in df1.txt -out vf1.txt

```
administrador@ubuntu-admin:~/Desktop/TASK4$ openssl enc -aes-128-cbc -d -nopad -in df1.txt -out vf1.txt \-K 001122334455667
78889aabbccddeeff \-iv 0102030405060708
administrador@ubuntu-admin:~/Desktop/TASK4$ openssl enc -aes-128-cbc -e -in f2.txt -out df2.txt \-K 0011223344556677889aab
bccddeeff \-iv 0102030405060708
administrador@ubuntu-admin:~/Desktop/TASK4$ openssl enc -aes-128-cbc -d -nopad -in df2.txt -out vf2.txt \-K 001122334455667
78889aabbccddeeff \-iv 0102030405060708
administrador@ubuntu-admin:~/Desktop/TASK4$ openssl enc -aes-128-cbc -e -in f3.txt -out df3.txt \-K 0011223344556677889aab
bccddeeff \-iv 0102030405060708
administrador@ubuntu-admin:~/Desktop/TASK4$ openssl enc -aes-128-cbc -d -nopad -in df3.txt -out vf3.txt \-K 001122334455667
78889aabbccddeeff \-iv 0102030405060708
```

Figura 6- Encriptação de três ficheiros (ficheiros f1, f2 e f3), utilizando o Modo CBC e aplicando "-nopad"

```
administrador@ubuntu-admin:~/Desktop/TASK4$ hexdump -C vf1.txt
00000000  31 32 33 34 35 0b 0b 0b  0b 0b 0b 0b 0b 0b 0b 0b  |12345...........|
00000010
administrador@ubuntu-admin:~/Desktop/TASK4$ hexdump -C vf2.txt
00000000  31 32 33 34 35 36 37 38  39 41 06 06 06 06 06 06  |123456789A......|
00000010
administrador@ubuntu-admin:~/Desktop/TASK4$ hexdump -C vf3.txt
00000000  31 32 33 34 35 36 37 38  39 41 42 43 44 45 46 01  |123456789ABCDEF.|
00000010
```

Figura 5- Verificação dos tamanhos dos ficheiros

3. Conclusão

Após realização deste trabalho foi-nos possível concluir que todos os modos de operação têm as suas vantagens e desvantagens e a forma de decisão de qual usar passa por cada situação em questão. Quanto a CTR verificamos que este modo de operação apresenta uma maior simplicidade que deverá ser aplicada caso não se interesse a ordem especifica de encriptação. Não existe propagação de erro e os blocos podem ser cifrados e decifrados independentemente, o que provoca uma maior eficiência. No entanto, devido a essa grande simplicidade este modo de operação não apresenta tanta segurança quanto CBC.

Em CBC existe uma maior segurança pois cada bloco cifrado depende do anterior, contudo, quando existe um erro num bloco, esse erro ira propagar-se pelos blocos seguintes o que provoca que a encriptação por vezes não seja a melhor.

Também nos foi ainda possível concluir que em relação aos modos de operação, existem alguns que utilizam *padding,* ou seja, existem modos de operação que acrescentam bytes de informação à mensagem para que o tamanho do bloco seja múltiplo do número de bytes do *plaintex.*

Quanto ao esquema de *Hash* conseguimos concluir que quanto menor o número de Bits utilizados e se não apresentar uma proteção contra segunda pré imagem, então é um esquema de integridade que facilmente é quebrado.

Com este trabalho também nos foi possível aprendermos mais sobre aplicação incremental de proteções e as suas vantagens. Aprendemos que ao aplicarmos estas proteções incrementalmente, poupamos espaço em memoria e também reduzimos a possibilidade de ataque, uma vez que o texto não fica tanto tempo em *plaintext.*

Em relação aos certificados digitais X.509 e as infraestruturas de chaves publicas, conseguimos observar que os certificados folha por si só não garantem integridade do site ou do documento que assinam, no entanto se conseguirmos comprovar a validade da cadeia de certificados em questão, o problema já não se aplica. No topo da cadeia de documentos encontra-se o certificado raiz que se auto assina e usa a sua chave publica para assinar o próximo certificado. Este certificado encontra-se instalado explicitamente num dado sistema de confiança.

4. Referências:

1. https://seedsecuritylabs.org/Labs_16.04/PDF/Crypto_Encryption.pdf
2. Slides disponibilizados pelo docente da unidade curricular