Universidade da Beira Interior
Faculdade de Engenharia
Departamento de Informática

Segurança Informática
Guia para Aula Laboratorial 7
Licenciatura em Engenharia Informática
Licenciatura em Informática Web
Licenciatura em Tecnologias e Sistemas da Informação

Computer Security
Guide for Laboratory Class 7
Degree in Computer Science and Engineering
Degree in Web Informatics
Degree in Information Technologies and Systems

Sumário
Exercícios de construção e verificação de códigos de autenticação de mensagens. Simulação dos passos do protocolo de acordo de chaves Diffie-Hellman.

Summary
Exercises concerning the construction and verification of message authentication codes. Simulation of the steps of the Diffie-Hellman key agreement protocol.

Pré-requisitos:
Algumas das tarefas propostas a seguir requerem o uso de *software* para efetuar cálculos, o acesso a um sistema com interpretador de programas escritos em linguagem de programação Python e que disponibilize a ferramenta OpenSSL. Sugere-se, assim, o uso de uma distribuição comum de Linux, onde todas estas condições estarão provavelmente preenchidas.

1 Código de Autenticação da Origem da Informação

Message Authentication Code

Considere que havia feito o protocolo de acordo de chaves Diffie-Hellman e que tinha estabelecido o segredo (simétrico) de 128 bits `5555ffff1234aedf9876cbcb6546789e`. Este segredo pode ser usado para cifrar ou calcular MACs.

Q1.: O que significa MAC?

M essage A uthentication C ode

Uma forma (relativamente fraca mas) simples de construir um MAC para um ficheiro é calcular o seu valor de *hash* com uma função de *hash* criptográfica (e.g., SHA1) e cifrar esse valor com uma cifra por blocos segura (e.g., AES).

Tarefa 1 ***Task 1***

Crie o ficheiro `texto-limpo.txt` e guarde lá dentro o seu nome seguido da sua idade. Crie o um MAC para este ficheiro conforme sugerido em cima:

1. Calcule o valor de *hash* para o SHA1 do ficheiro para `texto-limpo.sha1`;

   60737fc02e523237e63e0476910cea548e33f110

2. Cifre o valor de *hash* com o AES para o ficheiro `texto-limpo.aes-sha1`

Escreva os dois comandos *OpenSSL* que lhe permitem obter o efeito desejado:

```
$ openssl dgst -sah1 texto-limpo.sha1
```

```
$ openssl enc -e -aes128 -K 5555ffff1234aedf9876cbcb6546789e -in texto-limpo.sha1 -out texto-limpo-aes.sha1 -iv 0
```

Tarefa 2 ***Task 2***

Pegue numa moeda e atire-a ao ar. Se sair cara, altere o ficheiro que criou antes, deixando o MAC intacto. Caso contrário, não faça nada. Envie o ficheiro e o MAC para um(a) colega via e-mail (mas não lhe diga se alterou ou não o ficheiro). Espere também receber um de volta, e efetue o algoritmo de verificação do mesmo. Escreva os três comandos que lhe permitem obter o efeito desejado:

`$ openssl` _____

_____

`$ openssl` _____

_____

`$ diff` _____

_____

Q2.: O MAC que recebeu verifica ou não verifica?
☑ Sim verifica.
☐ Não, não verifica.
☐ Verifica metade, e a outra metade não. :S

Q3.: O MAC que criou é igual ao MAC que o(a) seu(ua) colega criou?
☐ Sim, é, porque a chave de cifra foi a mesma.
☑ Não, não é, porque o ficheiro é diferente.

Q4.: Dado ser um mecanismo da criptografia de chave simétrica, quantas entidades diferentes podem fazer ou verificar um MAC no seio de uma comunicação?
☐ Ninguém.
☑ Todos os que possuem a chave secreta.
☐ Todos os que possuem a chave privada.
☐ Todos os que possuem a chave pública.

Q5.: Quais das seguintes concretizam garantias dadas por um MAC?
☑ Garantia de que o ficheiro não sofreu erros aleatórios durante a transmissão (integridade).
☐ Garantia de que o ficheiro não foi visto por ninguém durante o caminho que fez entre o transmissor e o recetor.
☑ Garantia de que o ficheiro não foi alterado intencionalmente durante a transmissão (autenticação da origem da informação).
☐ Garantia vitalícia.
☐ 100.000 km ou 3 anos.
☐ Garantia de que o recetor não sabe de quem veio o ficheiro (anonimato).

Tarefa 3 ***Task 3***

Procure saber como se pode criar um HMAC usando o OpenSSL. **Q6.: É possível?**
☐ Não, não é possível.
☐ Sim, é, com um só comando:

`$ openssl` _____

_____

Q7.: Qual a proveniência da letra H no acrónimo HMAC?
☐ Helefant. ☐ *Holonymy*. ☐ *Hash*.
☐ *High*. ☐ *Home*. ☐ *Hack*.

Tarefa 4 ***Task 4***

Para terminar esta parte, escreva os comandos que lhe permitem verificar um HMAC produzido pelo OpenSSL:

`$ openssl` _____

_____

`$ diff` _____

_____

2 Protocolo de Acordo de Chaves Diffie-Hellman

Diffie-Hellman Key Agreement Protocol

O protocolo de acordo de chaves Diffie-Hellman, na sua forma original, usa grupos cíclicos sobre o conjunto dos números inteiros

$$\mathbb{Z}_p^* = \{1, 2, ..., p-1\},$$

em que p é um número primo e em que o cálculo do logaritmo discreto é intratável. Os grupos cíclicos são assim designados porque existe um número g, chamado *gerador*, que elevado a todas as potências de 0 a $p-1$, gera todos os elementos do grupo. No conjunto particular de inteiros delimitado por 1 e $p-1$, em que p é primo, é sempre possível encontrar um número que gera todos esses inteiros.

Tarefa 5 ***Task 5***

Nesta primeira tarefa, procura-se concretizar melhor o significado de gerador. Para isso, verifique se o número 2 é um gerador do grupo $\mathbb{Z}_{17}^*$, i.e., se

$$2^0 \bmod 17$$
$$2^1 \bmod 17$$
$$2^3 \bmod 17$$
$$...$$
$$2^{16} \bmod 17$$

geram todos os números entre 1 e 16. **Dica:** pode usar o *Libreoffice Calc* para fazer todos os cálculos necessárias.

Q8.: O número 2 é um gerador do grupo $\mathbb{Z}_{17}^*$?
☐ É sim senhor.
☐ Não, não é.
☑ Não, não é. Mas o 3 já é!
☐ É impressão minha, ou estas aulas acabaram de ficar muito mais estranhas do que já eram?

Tarefa 6 *Task 6*

Em casa, o Prof. gerou aleatoriamente um número x entre 1 e 23, e calculou $5^x \bmod 23$. O resultado foi 13. **Q9.: Qual foi o número aleatório que o Prof. usou e de que forma chegou a esse resultado?**

O número usado foi o 14.
Justificação: para encontrar este número usei um método muito bom baseado em
☐ Sorte. ☐ Magia negra. ☑ Tentativa e erro.

Note que, contrariamente ao que acontece nestas aulas, os números que estão envolvidos nestes cálculos têm normalmente mais do que 512 bits. Por exemplo, se pedir ao OpenSSL que gere os parâmetros do Diffie-Hellman com

```
$ openssl dhparam -text -C 512
```

e converter o resultado de hexadecimal para decimal usando uma instrução Python2 semelhante a

```
>>> int("number-in-hex",16)
```

irá obter algo parecido com:
112922958253978119684805383698547963508352786070916647541551325309046326112702324204595015986774547476988830231108889763781618167956948185661614468404472 03,
que **é**, convínhamos, **um pouco maior que 17 ou 23**.

Q10.: Consegue estimar quanto demoraria a encontrar um x aleatoriamente escolhido entre 1 e o número colocado em cima, tendo em conta o tempo que demorou a resolver a questão anterior?
☐ Entre 10 a 20 segundos.
☐ Entre 10 a 20 minutos.
☐ Entre 10 a 20 horas.
☐ Entre 10 a 20 dias.
☐ Entre 10 a 20 meses.
☐ Entre 10 a 20 anos.
☐ Entre 10 a 20 décadas.
☐ Entre 10 a 20 séculos...

Tarefa 7 *Task 7*

Forme um **grupo de três** (colegas). Escolham **duas pessoas** para fazer o protocolo de acordo de chaves Diffie-Hellman (a Alice e o Bob) e **um terceiro para escutar** as comunicações (a Claire). Use os parâmetros públicos $p = 23$ e $g = 5$. **Troque os valores públicos** usados pelo protocolo **oralmente**, de forma a que o(a) terceiro(a) colega escute esses valores.

Procedam da seguinte forma:

1. O(a) colega *Alice* gera um número secreto x entre 1 e 22 (inclusive) e calcula $X = 5^x \bmod 23$ **às escondidas**;
2. O(a) *Alice* transmite oralmente o número X ao(à) colega;
3. O colega *Bob* gera um número secreto y entre 1 e 22 (inclusive) e calcula $Y = 5^y \bmod 23$ **às escondidas**;
4. O(a) *Bob* transmite oralmente o número X ao(à) colega;
5. O(a) *Alice* calcula $k = Y^x \bmod 23$;
6. O(a) *Bob* calcula $k = X^y \bmod 23$.

Nota: os dois intervenientes principais não devem dizer a chave a que chegaram oralmente, mas devem tentar verificar se chegaram, de facto, à mesma chave.

Entretanto, o(a) colega que está a escutar tudo, tenta derivar a chave k. Note que, neste caso, como 23 é um número pequeno, esta tarefa do atacante será possível!

O cenário criado anteriormente simula um ataque de homem no meio passivo. **Q11.: Concorda com esta afirmação?**
☑ Sim, concordo, porque o atacante apenas conseguia escutar as comunicações.

- ☐ Sim, concordo, porque o atacante apenas conseguia escutar as comunicações num sentido.
- ☐ Não, não concordo, já que o atacante conseguia escutar as comunicações.
- ☐ Não, não concordo, já que o atacante conseguia escutar e alterar as comunicações.

Repare que o segredo que gerou no âmbito do protocolo anterior é muito pequeno e, por isso, claramente inseguro. Para que fosse seguro, p deveria ter mais do que 512 bits. Lidar com números deste tamanho requer a utilização de bibliotecas e funções específicas. No caso de programas em C, pode ser usada a biblioteca `bn.h` (*big number*), enquanto que em Java, pode-se recorrer à classe `BigInteger`.