CST em Análise e Desenvolvimento de Sistemas
SEG786203 – Segurança da Informação
Prof. Emerson Ribeiro de Mello
mello@ifsc.edu.br

Laboratório 1: criptografia

15/10/2024

1 Ferramenta OpenSSL

O OpenSSL[1] é uma biblioteca de criptografia de código aberto que implementa o protocolo TLS, além de fornecer ferramentas para gerar chaves, certificados, cifrar, decifrar e assinar e verificar mensagens. No Ubuntu, você pode instalar o OpenSSL com o comando: `sudo apt install openssl`. Na Listagem 1 são ilustrados os comandos para criação de uma chave simétrica, para cifrar e decifrar um arquivo com uma chave simétrica usando o OpenSSL.

Listagem 1: Criptografia simétrica de arquivos

```
# Gerando a chave simétrica de 32 bytes (256 bits) e salvando em chave.txt no formato base64
openssl rand -base64 32 > chave.txt

# Cifrar o arquivo mensagem.txt com a chave simétrica
openssl enc -aes-256-cbc -in mensagem.txt -out mensagem.enc -iter 1000 -pass file:chave.txt

# Decifrar o arquivo cifrado com a chave simétrica
openssl enc -d -aes-256-cbc -in mensagem.enc -out mensagem.dec -iter 1000 -pass file:chave.txt

# Usando o PBKDF2 para criar uma chave derivada de uma senha mestra e cifrar a mensagem
openssl enc -aes-256-cbc -salt -in mensagem.txt -out mensagem.enc -iter 1000 -pbkdf2 -k senha

# Decifrar o arquivo com a senha mestra
openssl enc -d -aes-256-cbc -in mensagem.enc -out mensagem.dec -iter 1000 -pbkdf2 -k senha
```

O OpenSSL também pode ser usado para gerar chaves assimétricas, assinar e verificar mensagens. Na Listagem 2 são ilustrados os comandos para gerar um par de chaves RSA, assinar e verificar uma mensagem.

Listagem 2: Criptografia assimétrica para assinatura digital

```
# Gerar um par de chaves RSA
openssl genrsa -out chave-privada.pem 2048
openssl rsa -in chave-privada.pem -pubout -out chave-publica.pem

# Gerando um par de chaves ECC com OpenSSL
openssl ecparam -name prime256v1 -genkey -noout -out chave-privada-ec.pem
openssl ec -in chave-privada-ec.pem -pubout -out chave-publica-ec.pem

# Imprimindo a chave no formato PEM
openssl rsa -in chave-privada.pem -text -noout
openssl ec -in chave-privada-ec.pem -text -noout

# Assinar a mensagem com a chave privada
openssl dgst -sha256 -sign chave-privada.pem -out mensagem.sig mensagem.txt

# Verificar a assinatura com a chave pública
openssl dgst -sha256 -verify chave-publica.pem -signature mensagem.sig mensagem.txt
```

[1] https://www.openssl.org

Para cifrar um arquivo com uma chave pública e decifrar com a chave privada, você pode usar o OpenSSL como ilustrado na Listagem 3. Porém, o tamanho do arquivo a ser cifrado não pode ser maior que o tamanho da chave RSA.

Listagem 3: Criptografia assimétrica para cifrar arquivos

```
# Cifrar o arquivo com a chave pública
openssl pkeyutl -encrypt -pubin -inkey chave-publica.pem -in mensagem.txt -out mensagem.enc

# Decifrar o arquivo com a chave privada
openssl pkeyutl -decrypt -inkey chave-privada.pem -in mensagem.enc -out mensagem.dec
```

Caso você queira cifrar um arquivo maior que o tamanho da chave RSA, você pode usar o OpenSSL para cifrar o arquivo com uma chave simétrica e cifrar a chave simétrica com a chave pública. Na Listagem 4 são ilustrados os comandos para cifrar e decifrar um arquivo com uma chave simétrica e cifrar e decifrar a chave simétrica com uma chave pública.

Listagem 4: Criptografia assimétrica para cifrar arquivos grandes

```
# Criando um arquivo com dados aleatórios e tamanho de 1MB
dd if=/dev/urandom of=grande.dat bs=1M count=1

# Gerar a chave simétrica
openssl rand -base64 32 > chave.txt

# Cifrar o arquivo com a chave simétrica
openssl enc -aes-256-cbc -in grande.dat -out grande.enc -iter 1000 -pass file:chave.txt

# Cifrar a chave simétrica com a chave pública
openssl pkeyutl -encrypt -pubin -inkey chave-publica.pem -in chave.txt -out chave.enc

# Decifrar a chave simétrica com a chave privada
openssl pkeyutl -decrypt -inkey chave-privada.pem -in chave.enc -out chave.txt

# Decifrar o arquivo com a chave simétrica
openssl enc -d -aes-256-cbc -in grande.enc -out saida.dat -iter 1000 -pass file:chave.txt
```

1.1 Exercício em dupla

1. Um componente da dupla deve baixar o PDF com os slides da aula de criptografia, gerar uma chave simétrica de 256 bits e cifrar o PDF com a chave simétrica. O outro componente da dupla deve decifrar o PDF com a chave simétrica para obter o conteúdo original.

    - Na raiz do repositório criar diretório `exercicio-1-1` e dentro desse criar arquivo `Readme.md`, onde deverá colocar todos os passos executados, na ordem exata, por cada componente da dupla para concluir a tarefa.

2. Um componente da dupla deve baixar o PDF com os slides da aula de criptografia e deve usar a criptografia de chave pública para cifrar o PDF. O outro componente da dupla deve decifrar o PDF e verificar se o conteúdo está íntegro, ou seja, se o conteúdo decifrado é igual ao conteúdo original e se a assinatura digital é válida (usando a chave pública do primeiro componente).

    - Na raiz do repositório criar diretório `exercicio-1-2` e dentro desse criar arquivo `Readme.md`, onde deverá colocar todos os passos executados, na ordem exata, por cada componente da dupla para concluir a tarefa.

2 Implementação da cifra de César em Java

A cifra de César é uma cifra de substituição simples, onde cada letra do texto é deslocada um número fixo de posições no alfabeto (para direita ou esquerda). Na Equação 1 e Equação 2 são apresentadas as fórmulas para criptografar e descriptografar uma letra com a cifra de César e com deslocamento à esquerda.

$$\mathcal{E}_n(x) = (x - n) \mod 26 \quad (1)$$

$$\mathcal{D}_n(x) = (x + n) \mod 26 \quad (2)$$

, sendo x a posição da letra no alfabeto e n o deslocamento.

2.1 Exercício em dupla

1. Um componente da dupla deve implementar a cifra de César em Java. O programa deve receber o deslocamento como argumento da linha de comando e o nome do arquivo de texto a ser criptografado. O programa deve ler o arquivo de texto, cifrar o texto e imprimir o texto crifado na tela.

   - Na raiz do repositório crie um diretório com o nome `exercicio-2-1` e dentro desse crie um projeto Java com gradle. O projeto deve conter um arquivo `Readme.md` com as instruções para compilar e executar o programa.

2. O outro componente da dupla deve implementar um programa que fará uso da força bruta[2] para decifrar o arquivo. O programa deve receber o nome do arquivo de texto cifrado como argumento de linha de comando e tentar todos os deslocamentos possíveis para decifrar o texto. Com ajuda do arquivo `palavras.txt` (disponível em https://github.com/pythonprobr/palavras/blob/master/palavras.txt), o programa deve verificar se o texto decifrado contém palavras em português para determinar se encontrou a chave correta. Para isso, considere que o texto decifrado contém palavras em português se pelo menos 50% das palavras do texto decifrado estiverem no arquivo `palavras.txt`. Se o texto decifrado contém palavras em português, o programa deve imprimir o texto decifrado e perguntar ao usuário se o texto está correto. Se o texto estiver correto, o programa deve imprimir o deslocamento usado na criptografia. Se o texto estiver incorreto, o programa deve tentar o próximo deslocamento.

   - Na raiz do repositório crie um diretório com o nome `exercicio-2-2` e dentro desse crie um projeto Java com gradle. O projeto deve conter um arquivo `Readme.md` com as instruções para compilar e executar o programa.

Discuta com os colegas e com o professor sobre as estratégias para implementar a cifra de César e a força bruta para decifrar o texto. Ler o arquivo de palavras e armazenar em um `HashSet` pode ser uma estratégia eficiente para verificar se uma palavra está no arquivo, pois a busca em um `HashSet` é $O(1)$ (algo que irão aprender em Estrutura de Dados).

[2] Existem outras técnicas mais eficientes, como a análise de frequência de letras https://pt.wikipedia.org/wiki/Frequ%C3%AAncia_de_letras, http://www.numaboa.com.br/criptografia/criptoanalise/309-Ferramenta-de-frequencia, http://www.numaboa.com.br/criptografia/criptoanalise/1051-exemplo