Universidade Federal de Uberlândia
Sistemas de Informação
Auditoria e Segurança da Informação

Vitor Manoel Gonçalves Teixeira
11811BSI201
Breno Henrique de Oliveira Ferreira
11821BSI245
Alex Júlio Silva de Abreu
11811BSI307

Relatório - OpenSSL 2

Uberlândia - Minas Gerais
Setembro de 2020

Tutorial - Ciframento Híbrido:

Nesse tutorial, serão explicadas algumas funcionalidades do openssl durante a simulação de um protocolo de ciframento híbrido envolvendo uma Autoridade Certificadora fictícia, com o objetivo de compartilhar uma mensagem criptografada de forma segura entre dois usuários. Durante o tutorial, os passos serão descritos como se os indivíduos Alice e Bob estivessem os executando, na vida real. Para simular um compartilhamento seguro dos dados entre Alice e Bob, foram criados dois diretórios, onde o compartilhamento é representado pela cópia de um diretório para o outro.

Diretórios: Criam-se estes diretórios para Alice e Bob:

```
mkdir alice
mkdir bob
```

Chave Raiz: Antes de a simulação começar, são geradas a SK(chave privada) e PK(chave pública) da Autoridade Certificadora(CA) fictícia, utilizando a função genrsa do openssl para gerar a SK, e a função rsa - um algoritmo de ciframento assimétrico - com a flag -pubout para que a PK pudesse ser extraída da SK.

Não foi criado um diretório específico para as chaves da CA, para ilustrar de forma mais clara que, na vida real, essa autoridade estaria hospedada em outro local, bem diferente de Bob e Alice.

Note que o resultado final são 2 arquivos: A SK(root.key) e a PK(root.pub).

```
openssl genrsa -out root.key 4096
openssl rsa -in root.key -pubout -out root.pub
```

Chave do Usuário: Após isso, Bob gera sua própria SK e extrai dela a PK, também utilizando os mesmos comandos, porém com uma pequena alteração:

```
openssl genrsa -out bob.key 2048
openssl rsa -in bob.key -pubout -out bob.pub
```

Observação: A SK da CA foi criada com mais bits que a do Bob, para possibilitar que a PK do Bob pudesse ser assinada com ela.

Note que o resultado final são 2 arquivos: A SK(bob.key) e a PK(bob.pub).

Certificado: A CA assina a chave pública de Bob com sua SK utilizando a flag -sign da função rsautl do openssl. A chave assinada é salva como “sigbob”. Esse procedimento serve como uma simulação do processo de a CA gerar um certificado para Bob utilizando sua PK.

```
openssl rsautl -sign -in bob.pub -inkey root.key -out sigbob
```

Recebimento do Certificado de Bob: Bob envia sua PK assinada para Alice que, com posse da PK da CA, verifica se tal chave é válida ao fazer uso da função rsautl do openssl. Para que a PK da CA possa ser usada nessa verificação, a flag -pubin é necessária.

Como Alice já tem conhecimento da PK de Bob, ela precisa apenas comparar as duas chaves: a PK original de bob(bob.pub), e a PK obtida com a verificação de assinatura(decbob.pub). Se ambas forem idênticas, o “certificado” de Bob está correto.

```
cp sigbob ../alice/
openssl rsautl -inkey root.pub -pubin -in sigbob -out decbob.pub
```

Comparação das chaves(exemplo):

trab2 > bob.pub

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA2GYnxtfjCiRLYYfc3VFG
PVlW5JKstDc25cx0WQ0cmsIoAvAfs1FkLAzTTC1ks8R6FpbX2cskTzvH8sBeZTm4
syE9ReMoMLMPy5DWQ5+haoPvwjlNa/bPTO1R4zzp2o0VyU6PBh6G8OipDoCsThdf
b58opHpwO91XKfvcNEad11QLYgsA3aO2cnxNEk/W1syQZ8wcuQZ9QxIbRzc3UiKT
7t3Cx8zpIm84Ia5IXgMPilifbJcfAtds4FIL8PXSXsgHhO0OYGOJE6+4YbQTWLA1
ohAeAtQ7G90a6cxUssP34fCOlS4f4CWw4HqpXIFzm69u252NGnzxTgsJ8rMO3j+5
4QIDAQAB
-----END PUBLIC KEY-----
```

trab2 > decbob.pub

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA2GYnxtfjCiRLYYfc3VFG
PVlW5JKstDc25cx0WQ0cmsIoAvAfs1FkLAzTTC1ks8R6FpbX2cskTzvH8sBeZTm4
syE9ReMoMLMPy5DWQ5+haoPvwjlNa/bPTO1R4zzp2o0VyU6PBh6G8OipDoCsThdf
b58opHpwO91XKfvcNEad11QLYgsA3aO2cnxNEk/W1syQZ8wcuQZ9QxIbRzc3UiKT
7t3Cx8zpIm84Ia5IXgMPilifbJcfAtds4FIL8PXSXsgHhO0OYGOJE6+4YbQTWLA1
ohAeAtQ7G90a6cxUssP34fCOlS4f4CWw4HqpXIFzm69u252NGnzxTgsJ8rMO3j+5
4QIDAQAB
-----END PUBLIC KEY-----
```

Chave de Sessão: Alice gera uma chave segura usando a função rand do openssl, que gera bits pseudo-aleatórios, cifra ela usando a PK de Bob e a envia para ele.

```
openssl rand -out sec.key 32
openssl rsautl -encrypt -pubin -inkey decbob.pub -in sec.key -out session.key
cp session.key ../bob
```

Segredo: Alice cifra uma mensagem contendo a frase “Hi Bob”(message.txt), utilizando as flags -aes-256-ctr(algoritmo de ciframento simétrico) e -pbkdf2(uma função-chave de derivação), -e(ciframento), e -pass file:(que recebe o nome do arquivo que contém a senha para decifrá-lo pós-ciframento) do openssl e envia a mensagem cifrada(encrypted.enc) para Bob.

```
openssl enc -aes-256-ctr -pbkdf2 -e -in message.txt -out encrypted.enc -pass file:sec.key
cp encrypted.enc ../bob
```

Segredo Revelado: Bob decifra a chave de sessão(session.key) enviada por Alice com sua SK utilizando a flag -decrypt da função rsautl do openssl.

Após isso, Bob usa a chave decifrada(decalice.key, a “chave segura” originalmente gerada por Alice com o openssl) para decifrar a mensagem criptografada recebida, usando a função enc do openssl com as flags -d(para decifrar) e -aes-256-ctr para especificar o tipo de algoritmo do ciframento.

```
openssl rsautl -decrypt -inkey bob.key -in session.key -out decalice.key
openssl enc -d -aes-256-ctr -pbkdf2 -in encrypted.enc -out decrypted.txt -pass file:decalice.key
```