



Disciplina: INE 5680 - Segurança da Informação e de Redes

Professora: Carla Merkle Westphall

Tarefa Prática – Handshake SSL/TLS+Criptografia assimétrica com OpenSSL no Linux (www.openssl.org)

Para executar a tarefa:

- Use screenshots para documentar a execução dos comandos.
- Executar e entregar todas as questões no moodle: entregar a **saída obtida nas questões + todos os arquivos gerados** (arquivos de teste, arquivos de chaves, etc).
- Compactar TUDO num único arquivo para entregar.
- Usar a Kali-Linux que já tem o openssl instalado! Você pode usar sua própria máquina Linux para realizar a tarefa.
- A Kali-Linux também já tem o Wireshark instalado (*Menu Applications -> 09-Sniffing & Spoofing*)

Handshake SSL/TLS (Secure Socket Layer/Transport Layer Security)

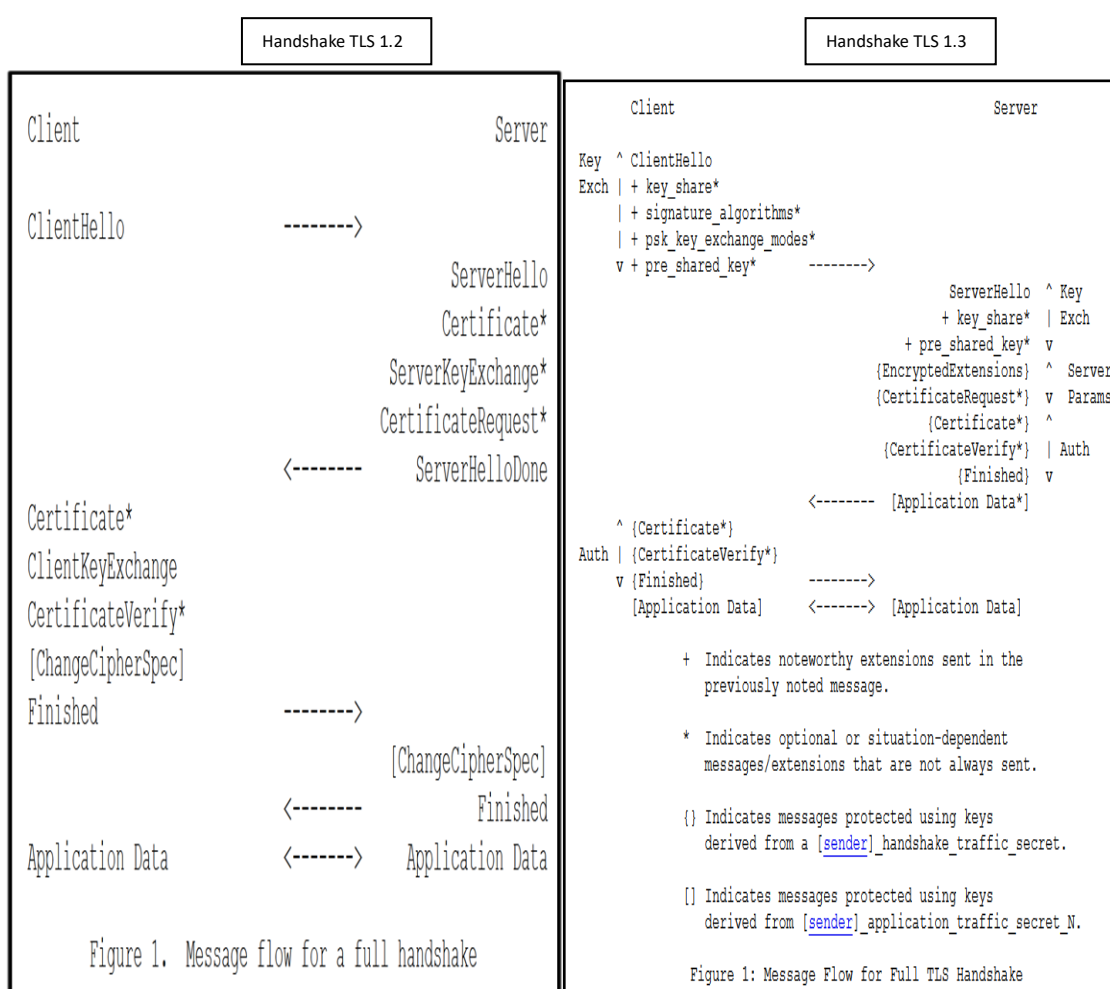
1) É possível verificar as possibilidades do SSL/TLS do seu browser e do seu servidor. Cole os resultados (screenshot) aqui **e comente o que chamou a sua atenção em cada um dos resultados.**

- a) <https://www.ssllabs.com/> - Acesse este site e teste o seu browser (diferentes tipos de browser podem ter resultados diferentes na sua máquina).
- b) <https://www.ssllabs.com/> - Acesse este site e teste um servidor que usa o SSL. Cuide para não acessar apenas um proxy de servidor real.

Obs.: **forward secrecy** significa que se uma chave for comprometida durante uma sessão, esse conhecimento/fato não afeta a segurança de sessões anteriores. A troca de chaves RSA (RSA key Exchange) não fornece *forward secrecy* pois se alguma chave privada for comprometida, todo o tráfego anterior pode ser decifrado.

2) Leia as recomendações da página <https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices> e faça um pequeno resumo das seções 1 e 2 dessas recomendações.

- 3) Explique de forma geral as quatro fases do handshake de acordo com as páginas do livro do Stallings do capítulo 17 – páginas 418 até 422, que explica o Protocolo de handshake (livro disponível no moodle no [link](#)).
- 4) Observando **o handshake** dos protocolos TLS v1.2 e TLS v1.3 através da leitura do material dos seguintes sites e da observação das figuras, **cite pelo menos 3 diferenças do handshake** entre as duas versões.
- <https://www.cloudflare.com/learning-resources/tls-1-3/>
 - <https://blog.cloudflare.com/rfc-8446-aka-tls-1-3/>
 - RFC TLS 1.3: <https://tools.ietf.org/html/rfc8446> (seção 2)
 - RFC TLS 1.2: <https://tools.ietf.org/html/rfc5246> (seção 7.3)



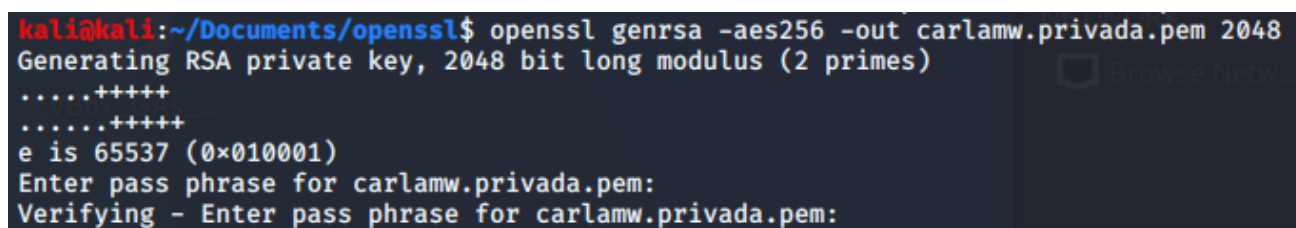
- 5) Mostre a execução dos comandos (parte da execução) que realizam o handshake com os seguintes sites:

```
openssl s_client -connect www.ufsc.br:443 -tls1_2
openssl s_client -connect youtube.com:443 -tls1_3
```

- 6) Agora, escolha um site (TLS 1.2 ou TLS 1.3) e realize a coleta do tráfego do handshake TLS no Wireshark. Para isso, ative a captura de pacotes no Wireshark, abra a conexão com um site que usa HTTPS no seu browser e capture o tráfego. Depois de estabelecer a conexão segura, pare a captura, salve a capture com seu nome (para entregar no moodle o arquivo). Use o “filtro” do Wireshark e coloque a string “ssl” para mostrar só pacotes do SSL. **Comente o handshake obtido no seu tráfego:**
- ClientHello: qual o objetivo? Qual o conjunto de cipher suites? Copie e cole um screenshot do campo Cipher Suites aqui na resposta.
 - ServerHello: qual o objetivo? Copie e cole o valor do Cipher Suite na mensagem Server Hello escolhido pelo servidor. Explique o formato da string usada no Cipher Suite (pode-se pesquisar na [RFC](https://wiki.mozilla.org/Security/Server_Side_TLS#Cipher_names_correspondence_table) ou na seguinte página web: https://wiki.mozilla.org/Security/Server_Side_TLS#Cipher_names_correspondence_table).
 - Qual o algoritmo de troca de chaves (Kx)?
 - Qual o algoritmo usado para autenticação (Au)?
 - Qual o algoritmo de criptografia simétrica e qual o modo (Enc)?
 - Qual o algoritmo de hash usado para o HMAC (Mac)?
 - O que é enviado na mensagem Certificate? Explique.
 - O que é enviado na mensagem ServerKeyExchange? Explique. Se o tráfego for TLS 1.3, olhe o campo key_share nas Extensions do ServerHello.
 - O que é enviado na mensagem ClientKeyExchange? Explique. Se o tráfego for TLS 1.3, olhe o campo key_share nas Extensions do ClientHello.
 - O Diffie-Hellmann foi usado em algum lugar? Explique.

OpenSSL - GERAR PAR DE CHAVES RSA e entender seus componentes

- 7) a) Gerar sua chave privada usando o comando:
`openssl genrsa -aes256 -out seunome.privada.pem 2048`



```
kali@kali:~/Documents/openssl$ openssl genrsa -aes256 -out carlamw.privada.pem 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
Enter pass phrase for carlamw.privada.pem:
Verifying - Enter pass phrase for carlamw.privada.pem:
```

Formato PEM: “PEM format is simply base64 encoded data surrounded by header lines.”
https://www.openssl.org/docs/man1.1.0/crypto/PEM_read_bio_X509_REQ.html

O arquivo seunome.privada.pem terá o seguinte formato de criptografia PEM (PEM ENCRYPTION FORMAT):

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-256-CBC,2F35552885460E55B5E36C8BFF96A0D1
... dados codificados em base64 ...
-----END RSA PRIVATE KEY-----
```

b) Explique o que é o parâmetro -aes256 do comando.

c) Explique o que significa a seguinte linha do arquivo `seunome.privada.pem` (https://www.openssl.org/docs/man1.1.0/crypto/PEM_read_bio_X509_REQ.html):

```
DEK-Info: AES-256-CBC,2F35552885460E55B5E36C8BFF96A0D1
```

- 8) Gere a chave pública a partir da chave privada com os comandos abaixo (guardar a chave pública no arquivo `seunome.publica.pem`). Explique a saída obtida em cada um dos comandos. **Guarde o arquivo gerado e envie o arquivo da sua chave pública junto nas respostas da tarefa.**

a) Explique a saída obtida no seguinte comando:

```
openssl rsa -in seunome.privada.pem -pubout -out seunome.publica.pem
```

```
Enter pass phrase for carlamw.privada.pem:
writing RSA key
```

```
Arquivo carlamw.publica.pem:
```

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAYR32Fi3R14eVbwME2jvn
2VixFdA3v2jlBsGEJRZ4PXhTAUILBzAgLf8U2sqC3T1CkJ+WegMKAHeeu5fqnuSB
2TpFpvyHBjHcqTRJjxdaVgwpc6Qhig7cVP4IXVL72dHKECSlrK9UCksU8lMTac44
L6g3om+5D6uV4c3MZbA/6kXq3lx00n0ThDE/Foe7n52OaYV+SoCmyQgtwwjzLmr5
Xh5FwxGMemldrrMcpsB0Eyu/Xi/+6y7bzSdwN+LW6upTXaS3P5na+YFod6HefGZN
2s59M14F+Qp6e+xq5RVf7ekTaYr4bTU4Kc1PTETLXjeQ5pJBubsI6y+7k8MChjx9
lwIDAQAB
-----END PUBLIC KEY-----
```

- 9) Digite o seguinte comando e depois abra o arquivo `seunome.publica.componentes`. Explique os componentes que constam nesse arquivo (<https://tools.ietf.org/html/rfc3447#appendix-A>). Comando:

```
openssl rsa -in seunome.privada.pem -out seunome.publica.componentes -text -noout
```

OpenSSL - ASSINATURA DIGITAL

- 10)a) Você deve assinar o arquivo fornecido na tarefa (`msgPlana.txt`). Para isso, crie o hash do arquivo `msgPlana.txt` e com a sua chave privada, assine o hash do arquivo:
- ```
openssl dgst -sha256 -sign seunome.privada.pem -out assinatura msgPlana.txt
```

b) Responda: qual o conteúdo do arquivo `assinatura`? Essa assinatura garante quais características de segurança: integridade, autenticidade, confidencialidade?

- 11) Verifique se o hash assinado está ok, isto é, compare o hash assinado com o hash do arquivo original usando o comando abaixo. Envie sua chave pública para que, durante a correção, possa ser feita a verificação da sua assinatura:
- ```
openssl dgst -sha256 -verify seunome.publica.pem -signature assinatura msgPlana.txt
```

OpenSSL - GERAR SEU CERTIFICADO NA ICPEDU

- 12) Acessar o site <https://e.ufsc.br/certificado-digital-p1-icpedu/emitir-certificado-p1/> e emita o seu certificado digital seguindo todos os passos desta página. Baixe o seu certificado pessoal. Coloque um screenshot mostrando que o certificado foi gerado. Responda: o que é o formato Personal Information Exchange (PKCS12)?
- 13) Importe o certificado no seu browser. Acesse a ajuda no link <https://pessoal.icpedu.rnp.br/public/ajuda>. Visualize as informações do seu certificado. Baixe a versão PEM do seu certificado. Depois, mostre as informações do seu certificado usando um screenshot e explique:
- Qual o algoritmo usado para gerar a chave pública e qual o tamanho da chave?
 - Qual o(s) algoritmo(s) usado(s) para fazer a assinatura digital do seu certificado (você pode usar o seguinte comando para auxiliar: `openssl x509 -text -in certificado.pem`).
 - Qual autoridade certificadora assinou o seu certificado?

OpenSSL - GERAR UM CERTIFICADO AUTO-ASSINADO ("auto" porque é assinado com SUA própria chave privada)

- 14) Para iniciar o processo de criação do SEU certificado, você deve inicialmente REQUISITAR uma assinatura no seu certificado (auto-assinado). A extensão `.csr` significa *Certificate Signing Request*. Usar o comando:
- ```
openssl req -new -key seunome.privada.pem -out certificado.csr
```

Exemplo obtido na saída:

```
kali@kali:~/Documents/openssl$ openssl req -new -key carlamw.privada.pem -out certificadocarlamw.csr
Enter pass phrase for carlamw.privada.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:BR
State or Province Name (full name) [Some-State]:SC
Locality Name (eg, city) []:Florianopolis
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UFSC
Organizational Unit Name (eg, section) []:SIN-2020
Common Name (e.g. server FQDN or YOUR name) []:Carla
Email Address []:carla.merkle.westphall@ufsc.br

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
kali@kali:~/Documents/openssl$
```

- 15) Agora o certificado X.509 AUTO-ASSINADO será efetivamente criado (assinado por você mesmo, usando a SUA chave privada), usando o comando: `openssl x509 -req -days 90 -sha512 -in certificado.csr -signkey seunome.privada.pem -out certificado.crt`

### Exemplo:

```
kali@kali:~/Documents/openssl$ openssl x509 -req -days 90 -sha512 -in certificadocarlamw.csr -signkey carlamw.privada.pem -out certificadocarlamw.crt
Signature ok
subject=C = BR, ST = SC, L = Florianopolis, O = UFSC, OU = SIN-2020, CN = Carla, emailAddress = carla.merkle.westphall@ufsc.br
Getting Private key
Enter pass phrase for carlamw.privada.pem:
kali@kali:~/Documents/openssl$
```

### Referências

1. Comandos: [http://wiki.openssl.org/index.php/Command\\_Line\\_Uutilities](http://wiki.openssl.org/index.php/Command_Line_Uutilities)
2. Livro OpenSSL Cookbook: <https://www.feistyduck.com/library/openssl-cookbook/online/>
3. Manpages: <https://www.openssl.org/docs/manpages.html>
4. Comandos: <https://www.openssl.org/docs/man1.1.0/apps/>
5. Simple Introduction: <https://sandilands.info/sgordon/simple-introduction-to-using-openssl-on-command-line>
6. Encrypt and decrypt files to public keys via the OpenSSL Command Line: [https://raymii.org/s/tutorials/Encrypt\\_and\\_decrypt\\_files\\_to\\_public\\_keys\\_via\\_the\\_OpenSSL\\_Command\\_Line.html#Get\\_the\\_public\\_key](https://raymii.org/s/tutorials/Encrypt_and_decrypt_files_to_public_keys_via_the_OpenSSL_Command_Line.html#Get_the_public_key)
7. Atividade - <http://en.wikiversity.org/wiki/Wireshark/HTTPS>
8. Wireshark/HTTPS - <http://wiki.wireshark.org/SSL>
9. RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2 - <https://tools.ietf.org/html/rfc5246>
10. RFC TLS 1.3 - The Transport Layer Security (TLS) Protocol Version 1.3 - <https://tools.ietf.org/html/rfc8446>
11. TLS - <https://hpbnc.co/transport-layer-security-tls/>
12. The First Few Milliseconds of an TLS 1.2 Connection - <https://tlseminar.github.io/first-few-milliseconds/>
13. A Detailed Look at RFC 8446 (a.k.a. TLS 1.3): <https://blog.cloudflare.com/rfc-8446-aka-tls-1-3/>
14. TLS 1.3 (with AEAD) and TLS 1.2 cipher suites demystified: how to pick your ciphers wisely: <https://www.cloudinsidr.com/content/tls-1-3-and-tls-1-2-cipher-suites-demystified-how-to-pick-your-ciphers-wisely/>
15. Browsing Experience Security Check: <https://www.cloudflare.com/ssl/encrypted-sni/>
16. The New Illustrated TLS Connection: <https://tls12.ulfheim.net/>