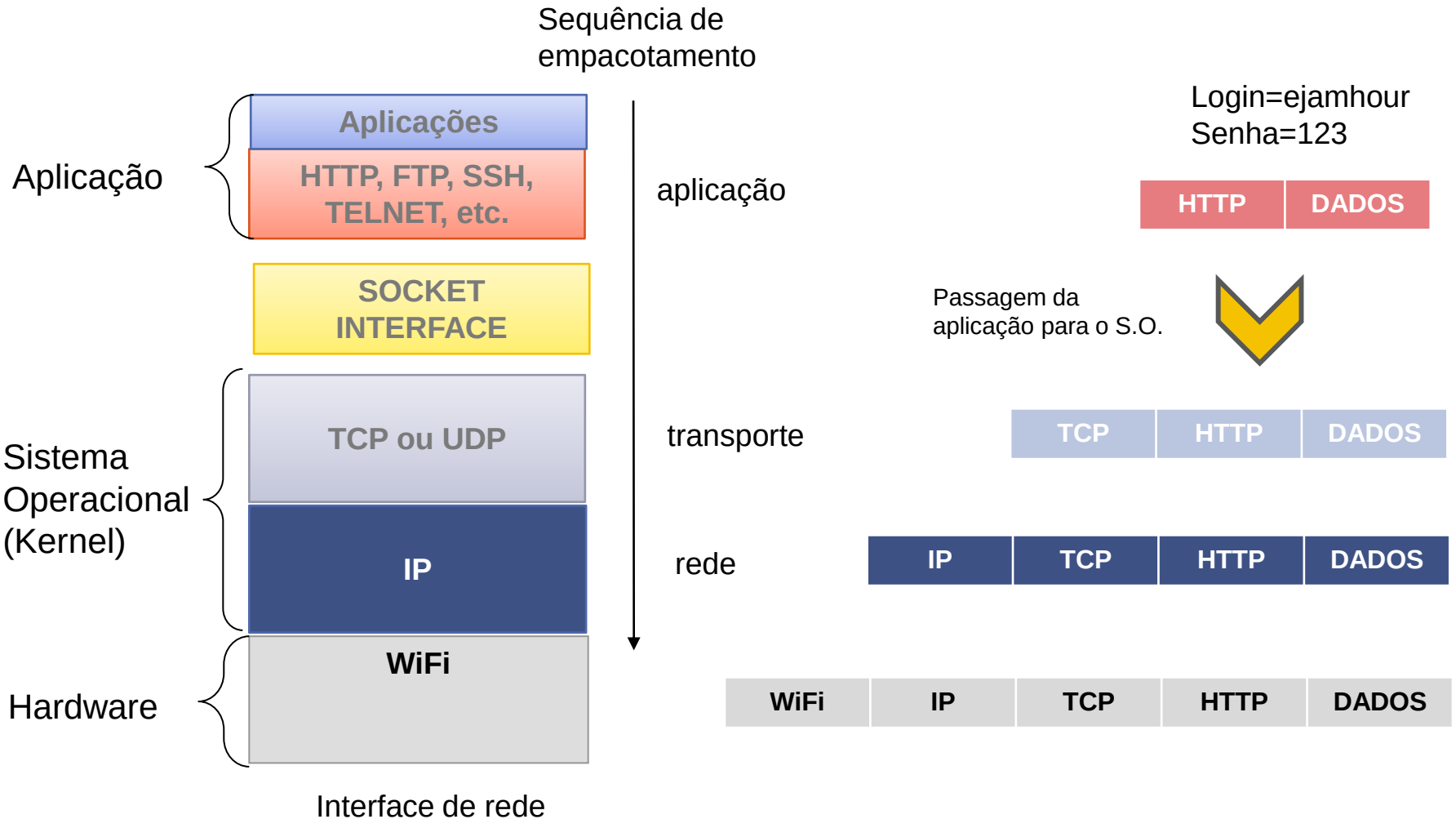
SSL/TLS

Professor

Edgard Jamhour

Como a criptografia pode ser implementada?



Analogia: Bonecas Russas

Wifi - comunicação no interior da residência

IP - comunicação através da Internet

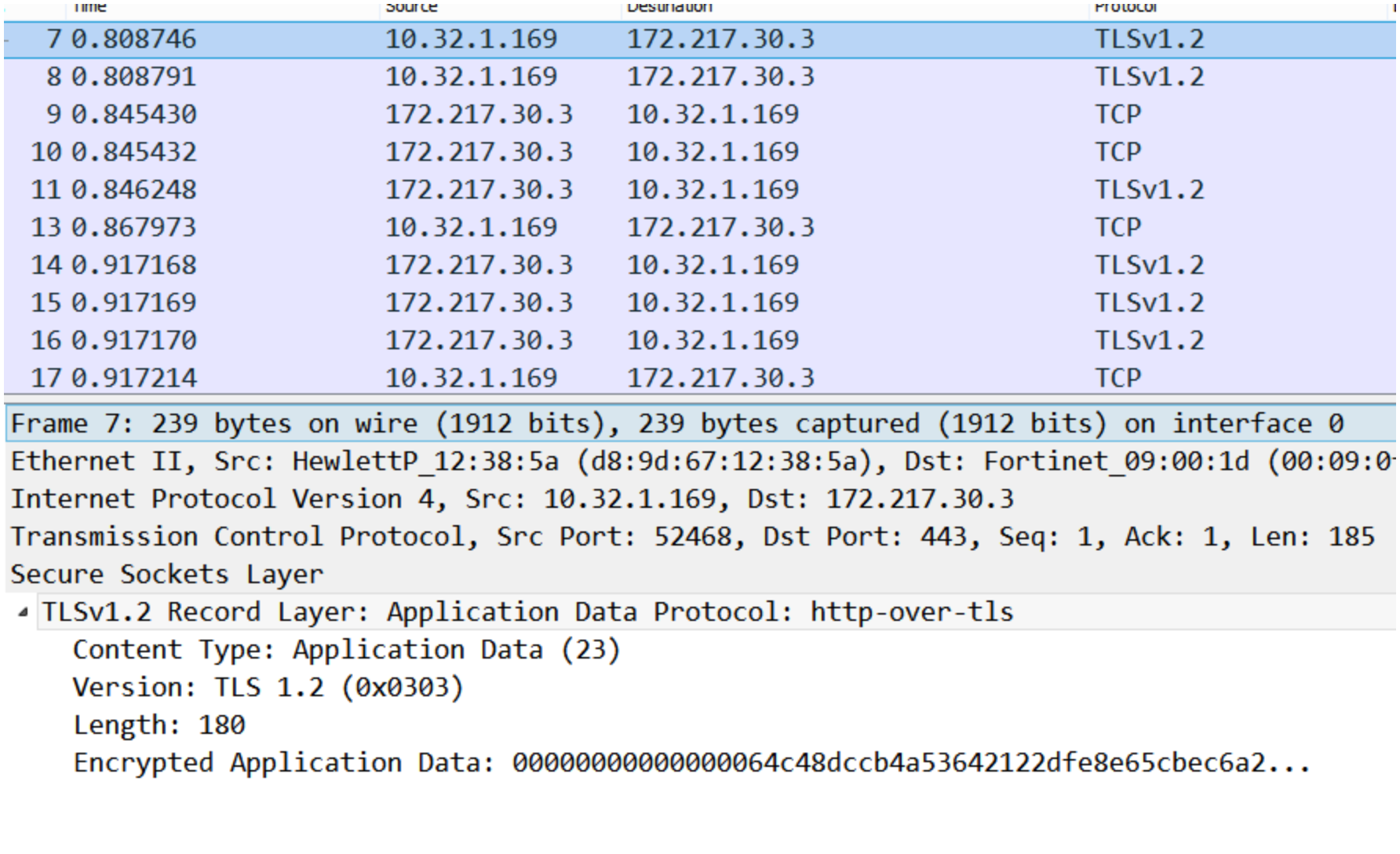
TCP- comunicação no interior do Sistema operacional.

Faz a mensagem chegar em um programa específico.

HTTP - Identifica nomes de servidores e os recursos acessados.

Dados

Segurança feita pela aplicação



	Time	Source	Destination	Protocol
7	0.808746	10.32.1.169	172.217.30.3	TLSv1.2
8	0.808791	10.32.1.169	172.217.30.3	TLSv1.2
9	0.845430	172.217.30.3	10.32.1.169	TCP
10	0.845432	172.217.30.3	10.32.1.169	TCP
11	0.846248	172.217.30.3	10.32.1.169	TLSv1.2
13	0.867973	10.32.1.169	172.217.30.3	TCP
14	0.917168	172.217.30.3	10.32.1.169	TLSv1.2
15	0.917169	172.217.30.3	10.32.1.169	TLSv1.2
16	0.917170	172.217.30.3	10.32.1.169	TLSv1.2
17	0.917214	10.32.1.169	172.217.30.3	TCP

```
Frame 7: 239 bytes on wire (1912 bits), 239 bytes captured (1912 bits) on interface 0
Ethernet II, Src: HewlettP_12:38:5a (d8:9d:67:12:38:5a), Dst: Fortinet_09:00:1d (00:09:0
Internet Protocol Version 4, Src: 10.32.1.169, Dst: 172.217.30.3
Transmission Control Protocol, Src Port: 52468, Dst Port: 443, Seq: 1, Ack: 1, Len: 185
Secure Sockets Layer
▾ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
    Content Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 180
    Encrypted Application Data: 00000000000000064c48dccb4a53642122dfe8e65cbec6a2...
```

SSL e TLS

SSL: Secure Socket Layer

- Definido pela Netscape
- Ultima versão: 3.0 (**0.3%** das conexões **HTTPs** feitas pelo Firefox)
- Considerado vulnerável atualmente (ataques POODLE e BEAST)

TLS: Transport Layer Security

- Definido pelo IETF
- Versão 1.0: RFC 2246 (Janeiro de 1999)
- TLS 1.0 também é considerado vulnerável a ataques BEAST (javascript)
- Versão atual: 1.2

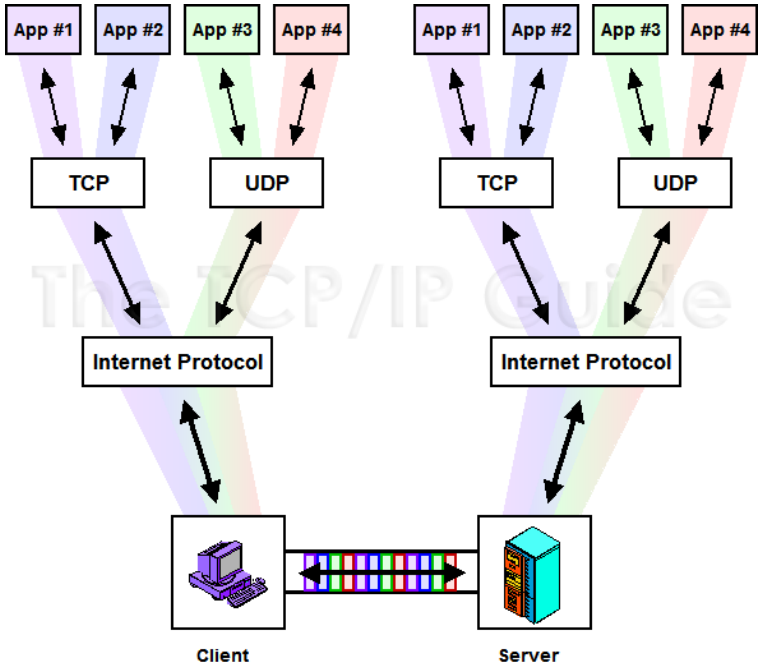
- O **TLS 1.0** é baseado no **SSL 3.0**, mas introduziu melhorias que o tornam mais seguro
  - O TLS foi atualizado pelas versões **TLS 1.1 , TLS 1.2 e TLS 1.3**
  - O TLS 1.1 é considerado vulnerável

INÍCIO de uma conexão segura

- Existem duas formas de um **Cliente** iniciar uma conexão segura (TLS/SSL) com um **Servidor**:
- **Por porta** (modo explícito):
  - O cliente conecta-se a uma porta diferente para iniciar a conexão segura:
  - Exemplo: 80 para conexão normal, 443 para conexão segura
- **Por protocolo** (modo implícito)
  - O cliente envia um “hello” desprotegido para o servidor e inicia um handshake. Se o handshake for bem sucedido, comuta para o modo seguro
  - Exemplo: STARTTLS usado pelo SMTP

OBS. ALPN (Application Layer Protocol Negotiation) permite escolher entre HTTP/1.1 e HTTP/2 na mesma porta, mas ambos são protegidos por TLS.

Inicialização por porta



Portas são números inteiros (entre 0 e 65525) que o **Sistema Operacional** usa para identificar programas.

Elas funcionam como endereços de **Caixa Postal**.

Programas que usam criptografia são associados a portas diferentes.

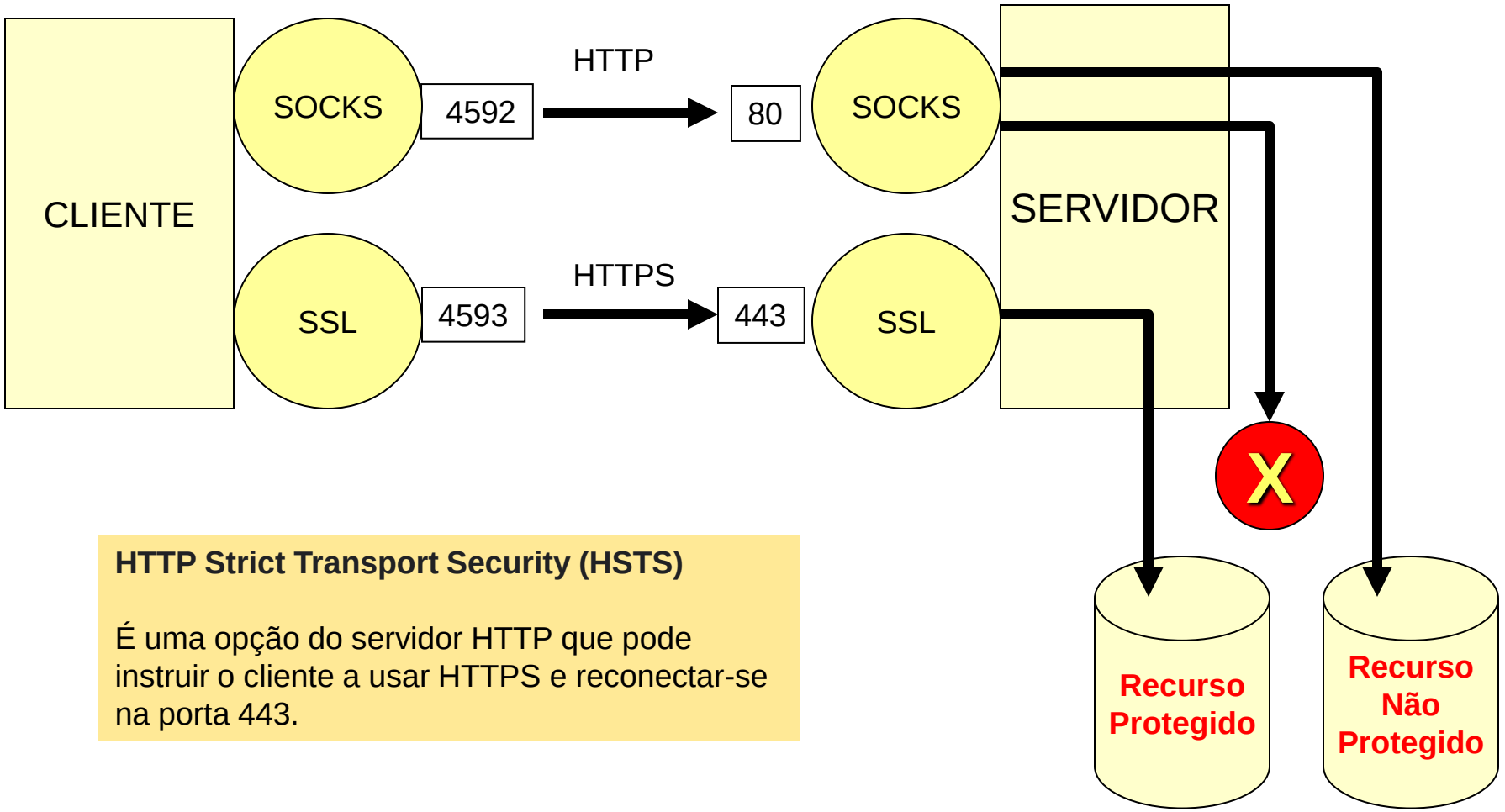
POP	IMAP	SMTP	HTTP	TELNET
110	143	25	80	22
SOCKET Interface				

POPS	IMAPS	SMTPS	HTTPS	TELNETS
995	993	465	443	992
SSL/TLS Interface				
SOCKET Interface				

TCP/IP

Exemplo: HTTPS

Hoje em dia, a maioria dos servidores tem fechado a porta 80, criptografando tudo pela porta 443.



HTTP Strict Transport Security (HSTS)

É uma opção do servidor HTTP que pode instruir o cliente a usar HTTPS e reconectar-se na porta 443.

QUIZ 1

Quem é responsável pela decisão de que a comunicação será protegida por TLS?

A. O servidor Web, mas o browser precisa suporta o TLS.

B. O browser, mas o servidor Web precisa oferecer um porta protegida TLS.

C. Ambos.

ESTABELECIMENTO DA CONEXÃO TLS

Uma conexão TCP protegida pelo TLS ocorre em três FASES

FASE 1: OFFLINE

- Entre o Administrador do Servidor e a CA
- Criação do **CSR** e **assinatura** do certificado **pela CA**

FASE 2: ONLINE

- Essa fase envolve o Cliente e o Servidor
- **Validação do certificado** do Servidor pelo cliente
- Envio de um **segredo aleatório** do cliente para o servidor

FASE 3: ONLINE

- Essa fase envolve o Cliente e o Servidor
- Geração de uma **chave secreta** (de **sessão**) usando o segredo
- Comunicação usando **criptografia simétrica** com a chave de sessão

TLS - FASE 1 (Preparação)

Servidor
(Subjet)

admin

O admin envia um CSR ,
contendo a chave pública para a CA

CSR
[PubS]

2

Autoridade
Certificadora (CA)

CSR
[PubS]

A CA assina o CSR
com sua chave privada

3

Chave privada
{PriCA}

1

O administrador(a) do servidor
gera um par de chaves
assimétricas (pública e privada)

Certificado
[PubS][AssCA]

4

A CA envia o
certificado X509
para o admin

Chave pública
{PubS}

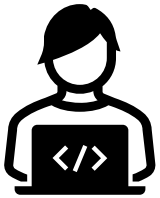
Chave privada
{PriS}

TLS- FASE 2

Servidor
(Subject)

1

O cliente se conecta no servidor



O servidor envia o seu certificado para o cliente

Certificado
[PubS][AssCA]

2

O cliente valida o certificado usando a chave pública da CA

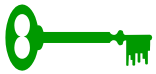
Chave pública
{PubCA}

Certificado
[PubS][AssCA]

3

E extrai a chave pública do servidor

Chave pública
{PubS}



Chave pública
{PubS}

Chave privada
{PriS}

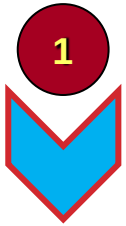
4

{{SegC}PubS}

O cliente envia um segredo aleatório {SegC} criptografado com a chave pública do servidor

TLS - FASE 3

Servidor
(Subject)



Segredo recebido
do cliente

{{SegC}PubS}

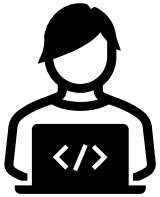
Chave privada
{PriS}

Segredo
{SegC}

O servidor decifra o
segredo do cliente
usando sua chave
privada

Chave Secreta
{SecC}

Cliente



Segredo
{SegC}

Chave Secreta
{SecC}

Cliente e Servidor geram uma chave
secreta a partir do segredo aleatório

{(Dados)SecC}

Cliente e Servidor se comunicam com criptografia
simétrica usando a chave secreta.

QUIZ 2

Quando o cliente consulta a autoridade certificadora (CA)?

A. Nunca

B. Sempre que recebe o certificado do servidor, verificar a assinatura do certificado.

C. Somente se quiser verificar se o certificado foi revogado pela CA.

D. No início da conexão, antes de receber o certificado do servidor.

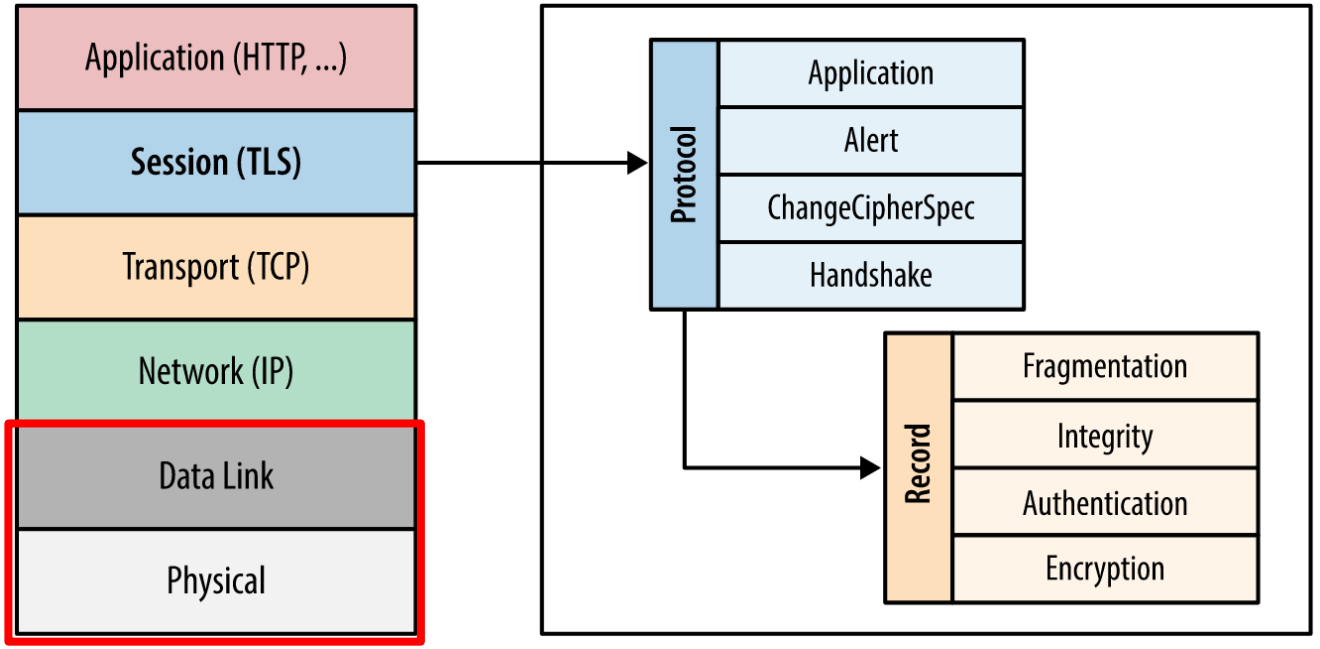
TLS: Objetivos

Segurança criptográfica entre dois pontos.

Interoperabilidade: cliente e servidor de fabricantes diferentes.

Extensibilidade: novos algoritmos de criptografia podem ser incorporados quando necessário.

Eficiência: reduzir o uso de CPU e o tráfego de rede a níveis aceitáveis.

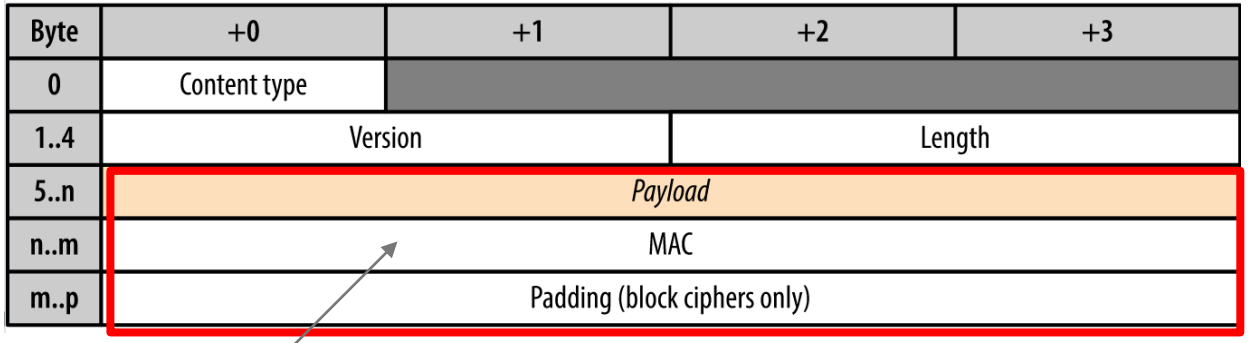


Hardware (interface de rede)

TLS: Sub-Protocolos

- **TLS Handshake Protocol**
  - Utilizado para negociar o algoritmo e as chaves de criptografia antes que o primeiro byte da comunicação seja transmitido.

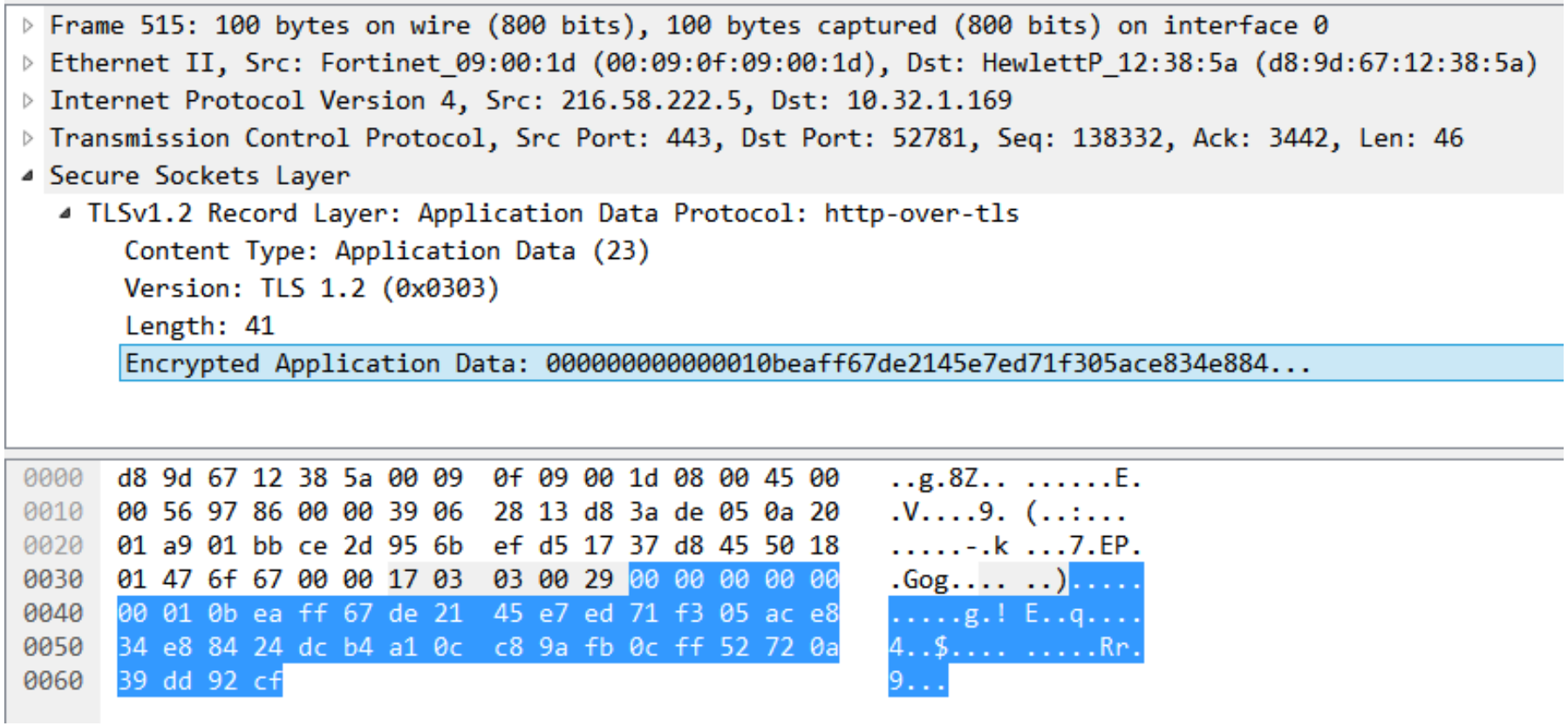
- **TLS Record Protocol**
  - Utilizado para encapsular os protocolos das camadas superiores.
  - Para controle de integridade, MAC (Message Authentication Code) ou HMAC é adicionado a cada registro.



Toda essa parte é criptografada

Observe que os pacotes são assinados individualmente !!!

TLS Record Protocol



QUIZ 3

Quais protocolos o TLS protege?

A. Todos os protocolos do pacote, incluindo os usados pelo hardware

B. Os protocolos usados pelo S.O. e pela aplicação.

C. Apenas os protocolos usado pela aplicação.

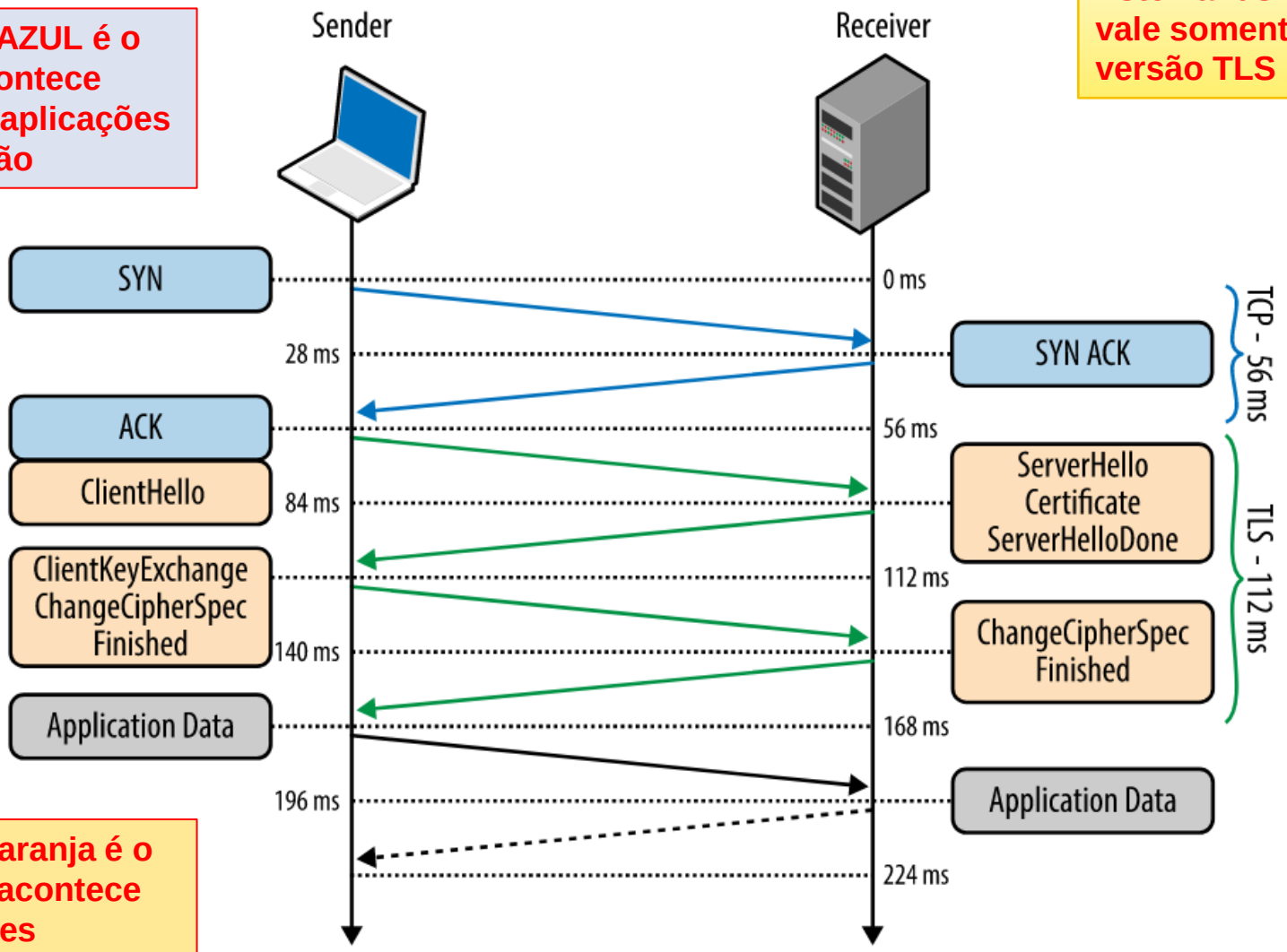
D. Depende da situação.

TLS Handshake

IMPORTANTE: O TLS protege apenas o TCP. Para proteger UDP usa-se DTLS (bem mais recente).

A parte em AZUL é o TCP. Ela acontece mesmo em aplicações sem proteção

Este handshake vale somente até a versão TLS 1.2



A parte em laranja é o TLS. Ela só acontece em aplicações protegidas.

Source: https://hpbn.co/transport-layer-security-tls/

Atualização do TLS 1.3

TLS 1.2 e 1.3 estão sendo atualmente utilizados

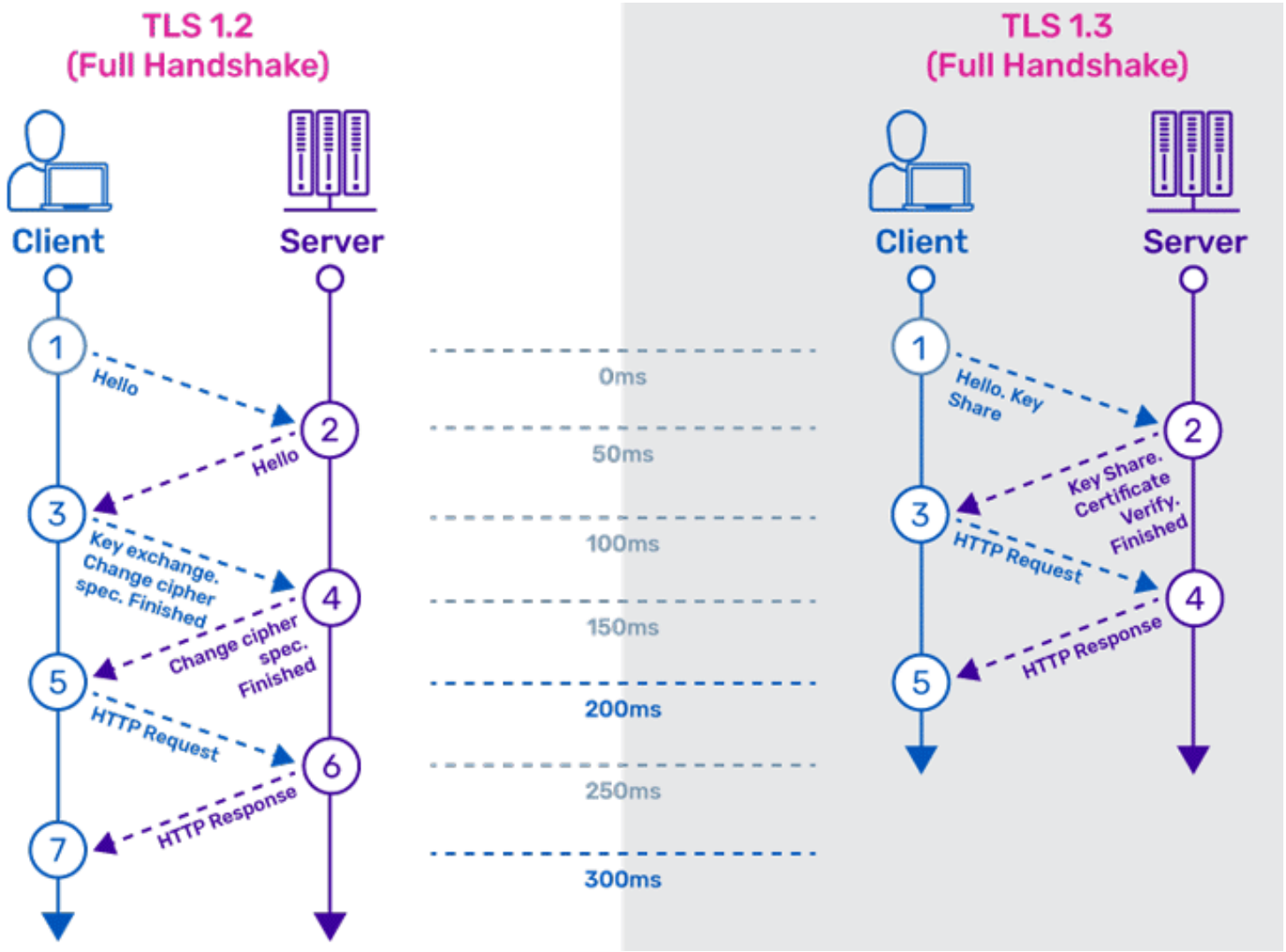
- TLS 1.2: Agosto de 2008 (RFC 5246)
- TLS 1.3: Agosto de 2018 (RFC 8446)

O que o TLS 1.3 mudou?

a. O RSA não é mais usado para combinar a chave de sessão, apenas DH e ECDH.
b. O RSA e EC podem ser usados para autenticação.
c. Handshake está mais rápido, com menos mensagens
d. O certificado X509 é transmitido de forma criptografada
e. Vários algoritmos, como DES, RC4, AES-CBC foram tornados obsoletos.
f. 0-RTT: permite usar um handshake ainda mais simplificado se já houver uma chave negociada anteriormente.

site sobre TLS 1.3

Atualização do TLS 1.3



QUIZ 4

Qual a diferença entre HTTP e HTTPS?

A. HTTPS é um protocolo novo, criado para ter mais segurança.

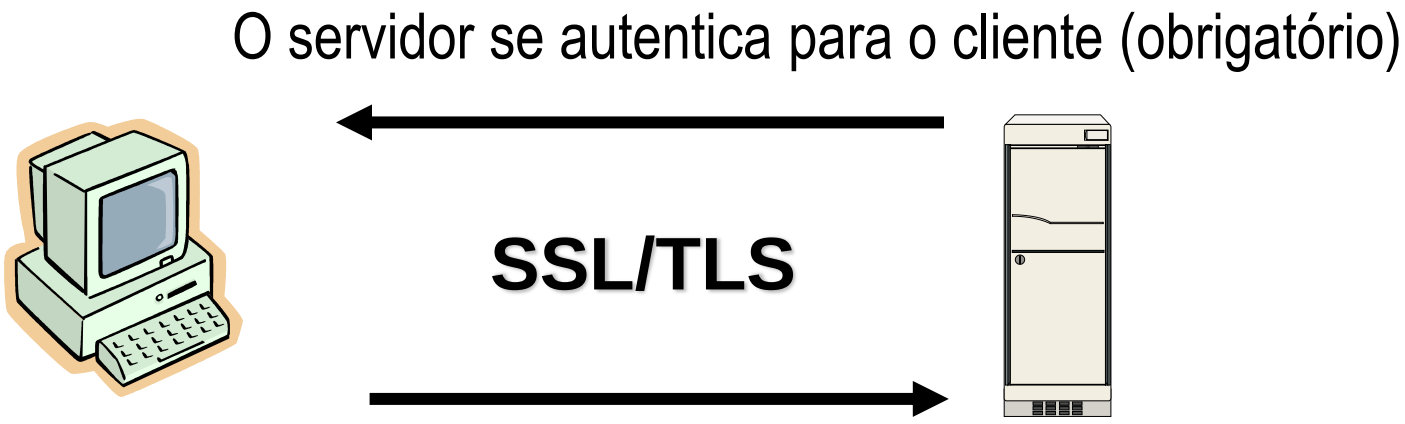
B. HTTPS é um protocolo que usa TLS ao invés de TCP.

C. HTTPS é o mesmo HTTP, protegido por TLS, e transportado por TCP.

D. HTTPS significa HTTP under TLS.

TLS handshake

- Escolha dos algoritmos de assinatura e criptografia
- Troca de certificados entre o cliente e o servidor
- Troca de segredo compartilhado para geração da chave secreta

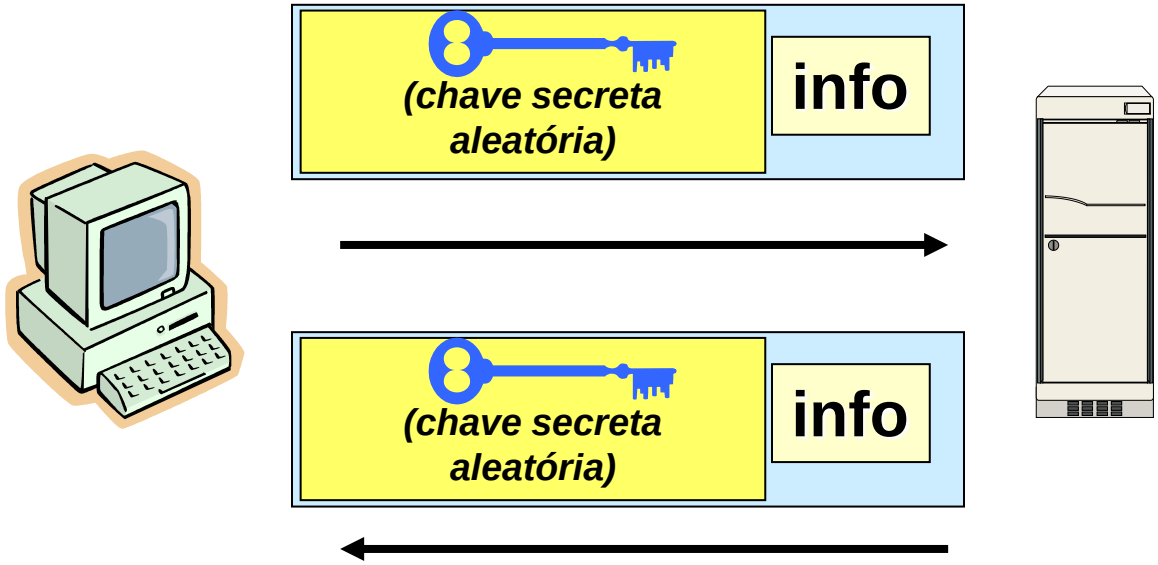


Escolha do algoritmo de criptografia

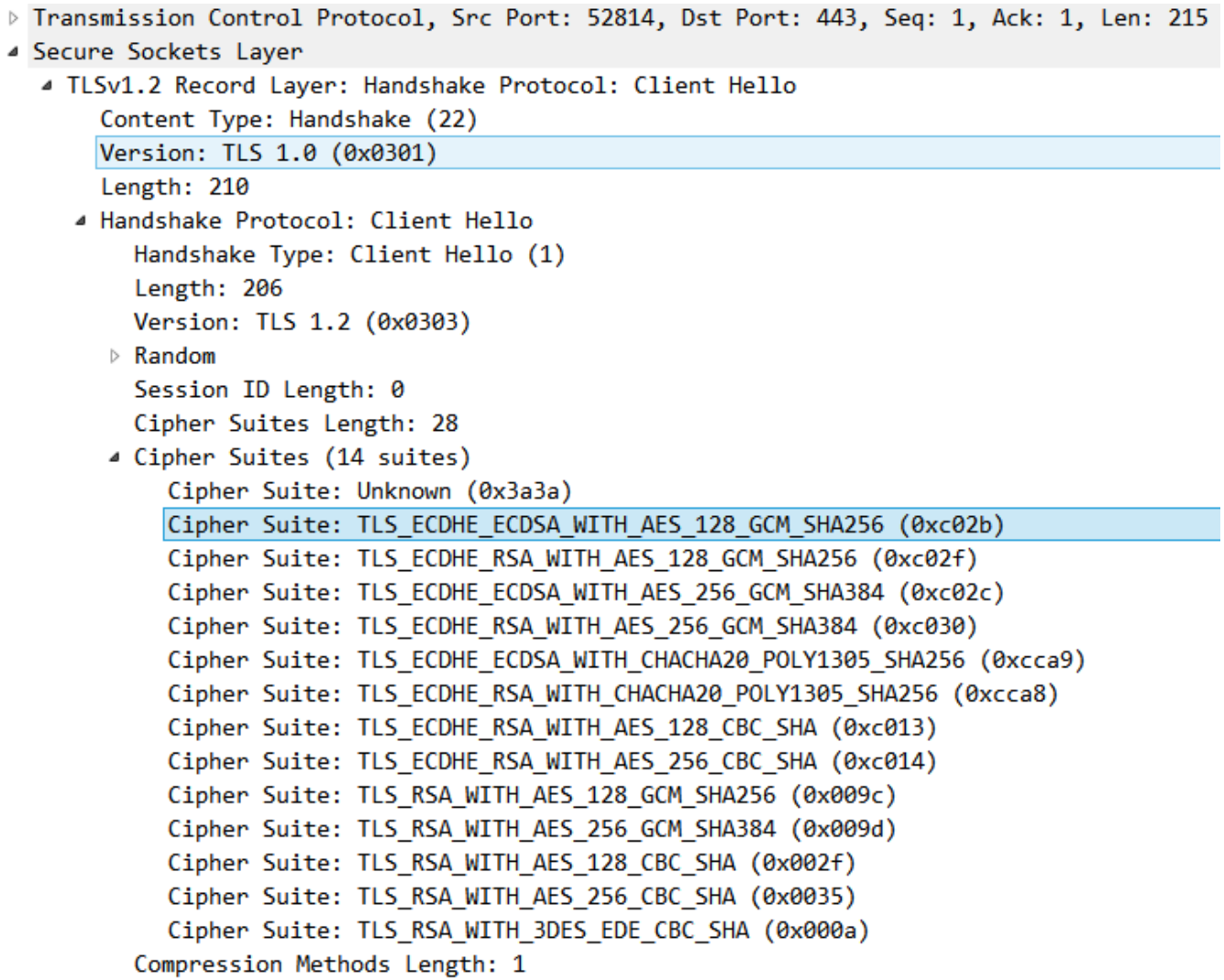
O cliente oferece uma lista de opções de algoritmos de criptografia para o servidor.

O servidor escolhe um dos algoritmos e informa o cliente.

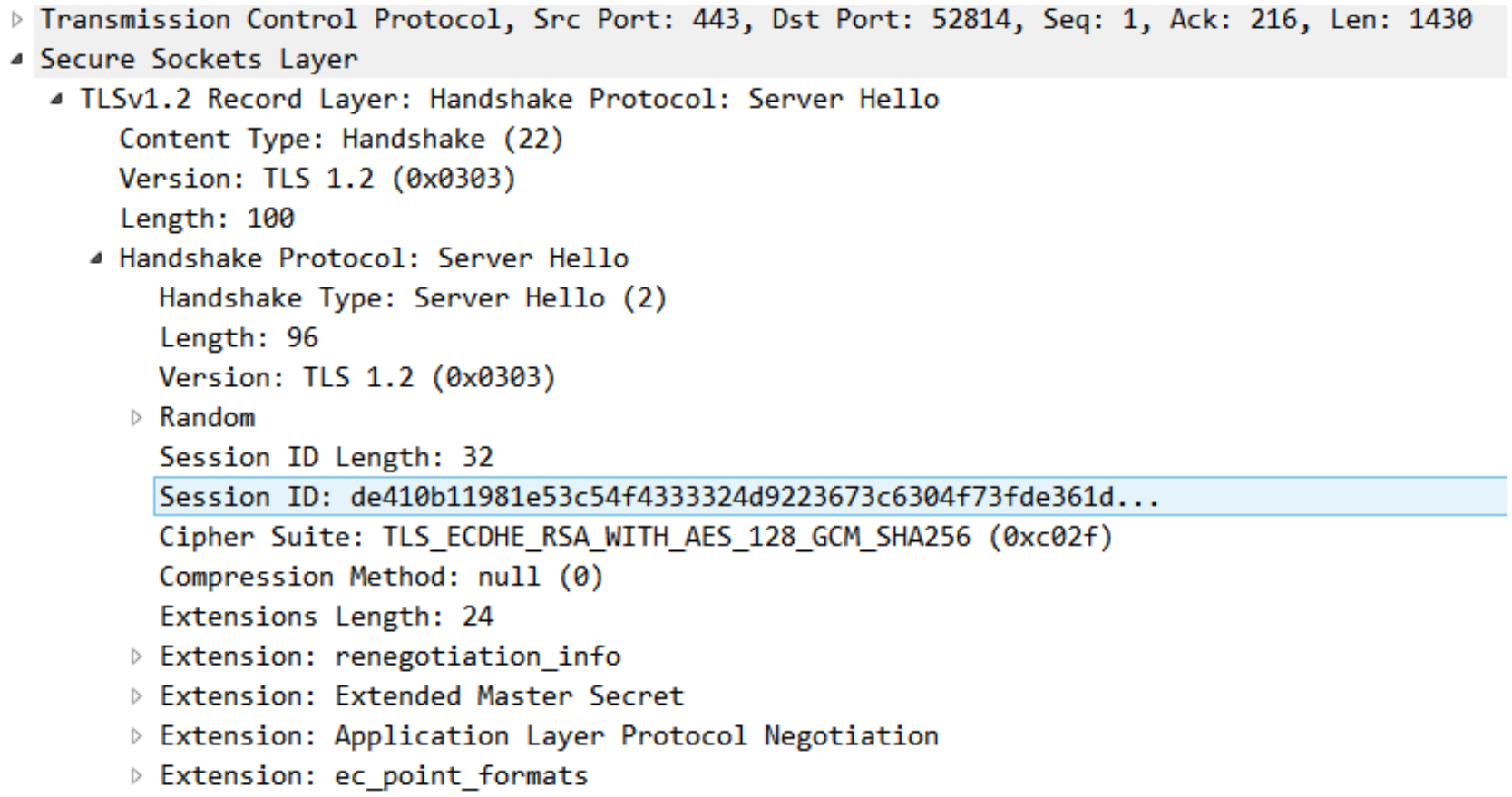
A comunicação segura acontece usando uma chave secreta gerada de acordo com o algoritmo escolhido.



TLS HANDSHAKE: Cliente HELLO



TLS HANDSHAVE: SERVER HELLO



QUIZ 5

Analisando as capturas do Wireshark, qual o tipo de certificado utilizado e qual o algoritmo simétrico escolhido?

A. ECDHE e RSA

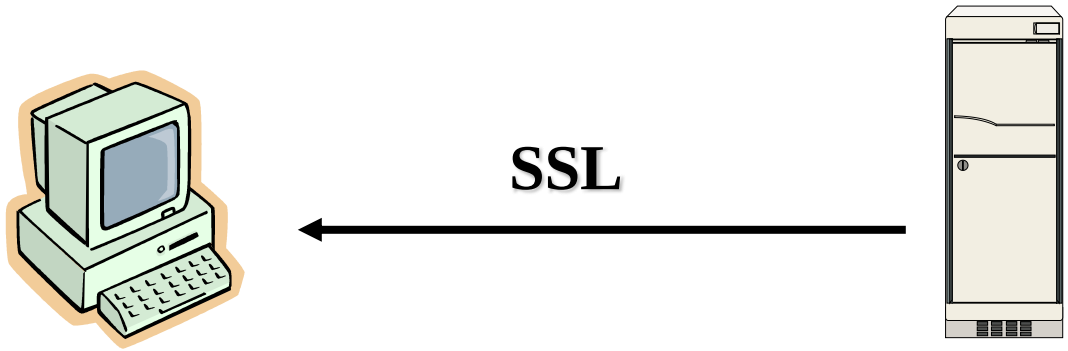
B. RSA e SHA256

C. ECDHE_RSA e AES_128

D. ECHDHE e RSA_WITH_AES_129.

Autenticação do Servidor

- O envio do certificado do servidor para o cliente é obrigatório.
- Caso o certificado tenha sido assinado por uma CA intermediária, o certificado da CA também precisa ser enviado.

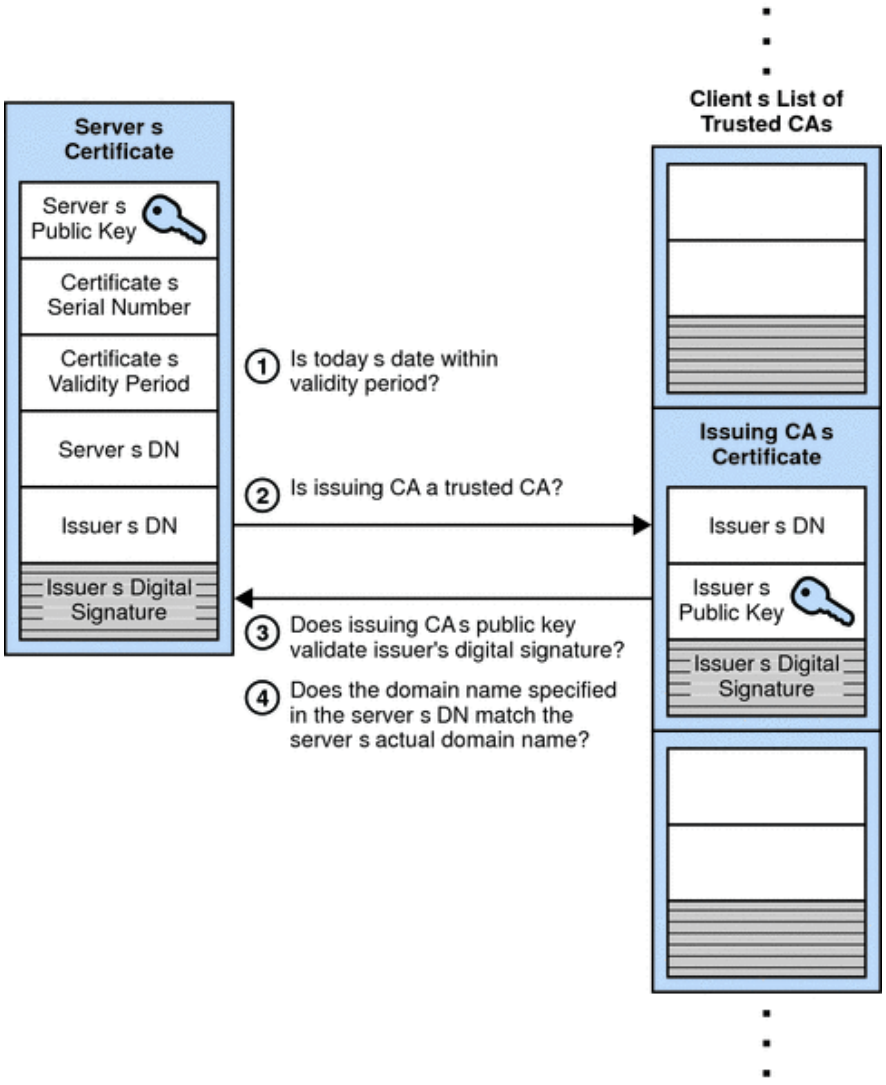


Chave pública do servidor	Identificação do Servidor	Identificação do CA	Assinatura Digital de uma CA

TLS HANDSHAKE: Server Certificate

```
▲ Handshake Protocol: Certificate
    Handshake Type: Certificate (11)
    Length: 2249
    Certificates Length: 2246
  ▲ Certificates (2246 bytes)
      Certificate Length: 1172
    ▲ Certificate: 3082049030820378a00302010202087f96df3e0c57054730... (id-at-commonName=*.gcp.gvt2.com,id-at-organizationName=Google Inc,ic
      ▲ signedCertificate
          version: v3 (2)
          serialNumber: 9193781146911442247
        ▷ signature (sha256WithRSAEncryption)
        ▲ issuer: rdnSequence (0)
          ▷ rdnSequence: 3 items (id-at-commonName=Google Internet Authority G2,id-at-organizationName=Google Inc,id-at-countryName=US)
        ▷ validity
        ▲ subject: rdnSequence (0)
          ▷ rdnSequence: 5 items (id-at-commonName=*.gcp.gvt2.com,id-at-organizationName=Google Inc,id-at-localityName=Mountain View,id-at
        ▷ subjectPublicKeyInfo
        ▷ extensions: 8 items
      ▷ algorithmIdentifier (sha256WithRSAEncryption)
        Padding: 0
        encrypted: 6b989afe5d03da0dc05820162c2d46317e8f19d0938d2250...
      Certificate Length: 1068
    ▲ Certificate: 3082042830820310a003020102021001000212588b0fa59a7... (id-at-commonName=Google Internet Authority G2,id-at-organizationName
      ▲ signedCertificate
          version: v3 (2)
          serialNumber: 0x0100212588b0fa59a777ef057b6627df
        ▷ signature (sha256WithRSAEncryption)
        ▲ issuer: rdnSequence (0)
          ▷ rdnSequence: 3 items (id-at-commonName=GeoTrust Global CA,id-at-organizationName=GeoTrust Inc.,id-at-countryName=US)
        ▷ validity
        ▲ subject: rdnSequence (0)
          ▷ rdnSequence: 3 items (id-at-commonName=Google Internet Authority G2,id-at-organizationName=Google Inc,id-at-countryName=US)
        ▷ subjectPublicKeyInfo
        ▷ extensions: 8 items
      ▷ algorithmIdentifier (sha256WithRSAEncryption)
        Padding: 0
        encrypted: ca49e5acd76464775bbe71facff41e23c79a6963545feb4c...
```

Certificados de Servidor

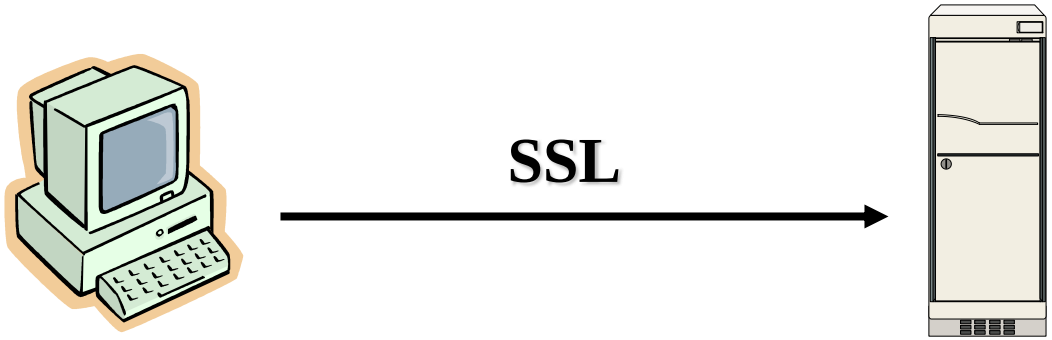


- Both Alice and Bob generate their own public and private keys.
- Both Alice and Bob hide their respective private keys.
- Alice shares her public key with Bob, and Bob shares his with Alice.
- Alice generates a new message for Bob and signs it with her private key.
- Bob uses Alice’s public key to verify the provided message signature.

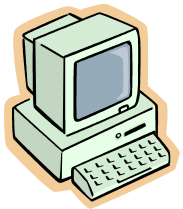
Autenticação do Cliente

O envio do certificado do cliente é requisitado pelo servidor.

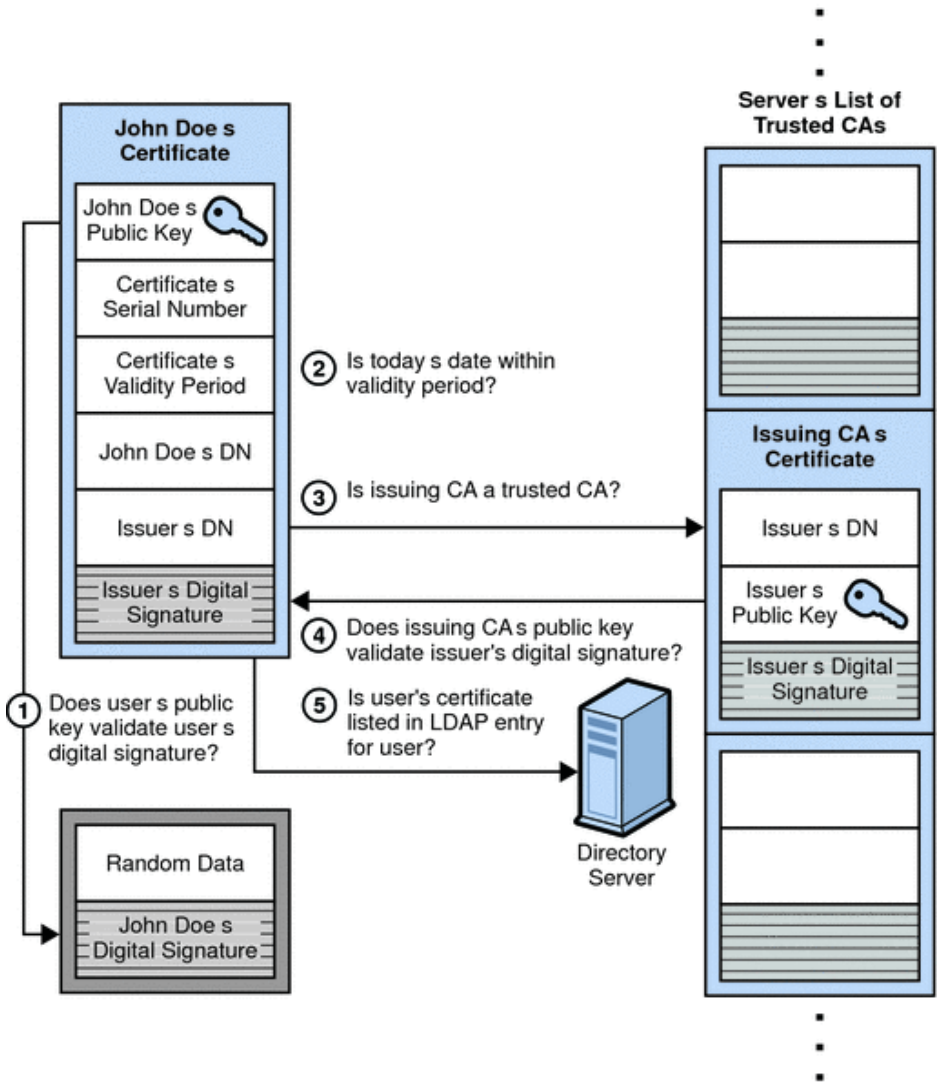
Esse mecanismo é opcional, e permite autenticar a máquina do usuário sem intervenção manual.



Chave pública do Cliente	Identificação do Cliente	Identificação do CA	Assinatura Digital de uma CA



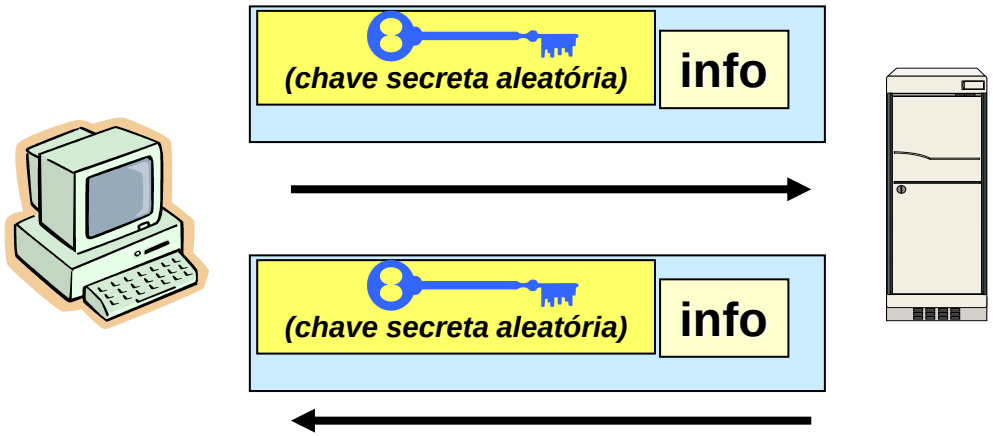
Certificados de Cliente



Criação do segredo compartilhado

```
▷ Transmission Control Protocol, Src Port: 52815, Dst Port: 443, Seq: 202, Ack: 3782, Len: 258
◢ Secure Sockets Layer
  ◢ TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 37
    ◢ Handshake Protocol: Client Key Exchange
        Handshake Type: Client Key Exchange (16)
        Length: 33
      ◢ EC Diffie-Hellman Client Params
          Pubkey Length: 32
          Pubkey: f8d152cc383d62b75be6586401708d0b7bd63677ee57594e...
  ▷ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  ◢ TLSv1.2 Record Layer: Handshake Protocol: Multiple Handshake Messages
```

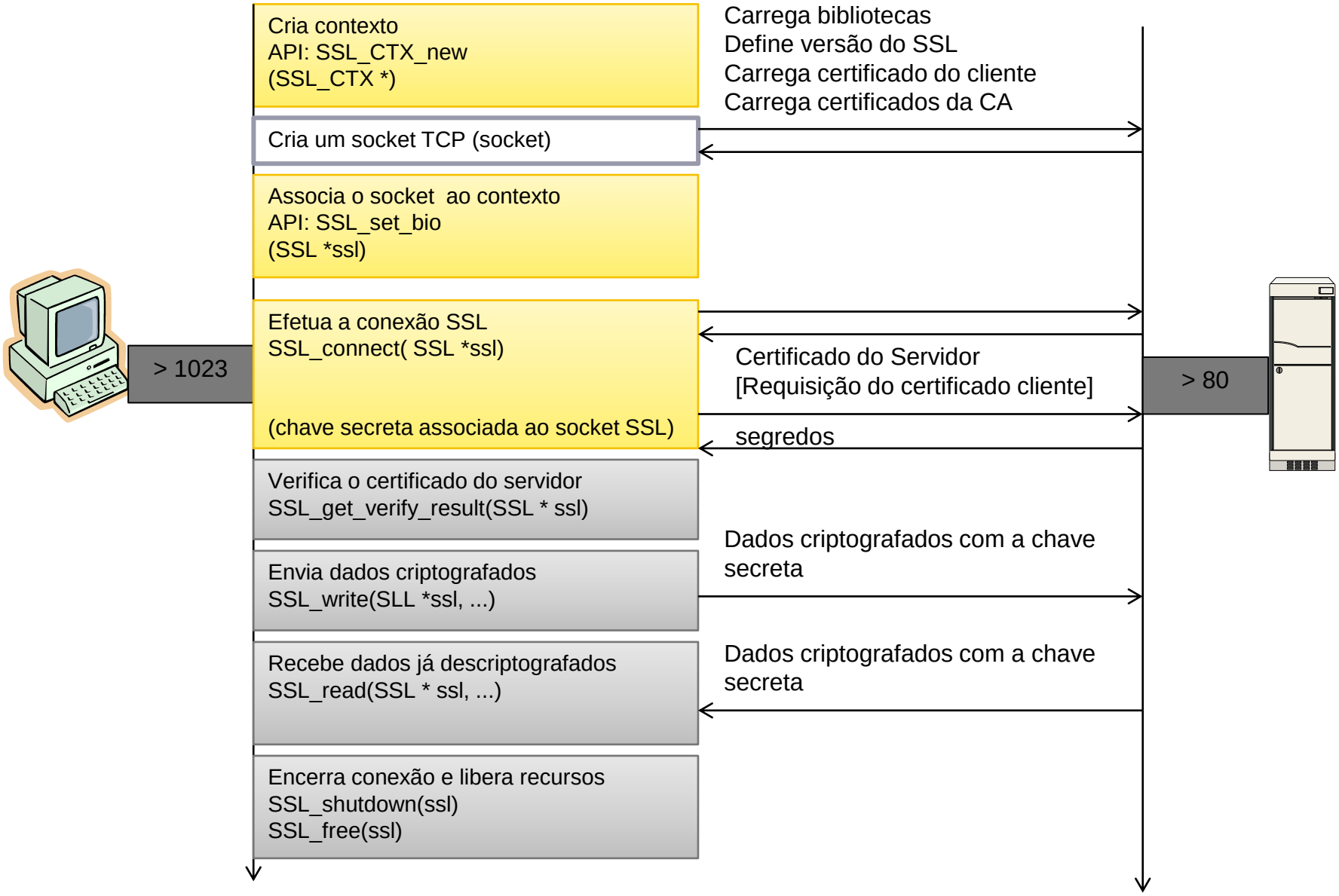
A chave secreta é derivada do segredo



TLS HANDSHAKE: RESUMO

1. TLS é executado sobre uma conexão TCP já estabelecida
2. O cliente envia a versão de TLS e a lista de ciphersuites suportados para o servidor.
3. O servidor envia para o cliente o ciphersuite escolhido e seu certificado. Opcionalmente solicita o certificado do cliente.
4. O cliente inicia uma troca de chaves do tipo RSA ou Diffie-Hellman para estabelecer a sessão segura baseada em chave secreta
5. O servidor envia uma mensagem criptografada com a chave secreta
6. O cliente descriptografa a mensagem gerada com a chave secreta.

API SSL


Cria contexto
API: SSL_CTX_new
(SSL_CTX *)
Carrega bibliotecas
Define versão do SSL
Carrega certificado do cliente
Carrega certificados da CA
Cria um socket TCP (socket)
Associa o socket ao contexto
API: SSL_set_bio
(SSL *ssl)
> 1023
Efetua a conexão SSL
SSL_connect(SSL *ssl)
(chave secreta associada ao socket SSL)
Certificado do Servidor
[Requisição do certificado cliente]
segredos
> 80
Verifica o certificado do servidor
SSL_get_verify_result(SSL * ssl)
Envia dados criptografados
SSL_write(SLL *ssl, ...)
Dados criptografados com a chave secreta
Recebe dados já descriptografados
SSL_read(SSL * ssl, ...)
Dados criptografados com a chave secreta
Encerra conexão e libera recursos
SSL_shutdown(ssl)
SSL_free(ssl)

Conclusão

TLS: protocolo de negociação

- Funciona ao nível da aplicação
- Protege apenas TCP
- Permite negociar os algoritmos usados para proteger as mensagens
- Define o formato das mensagens criptografadas
- Usado para proteger o HTTPS: HTTP over TLS
- Define uma API para criar aplicações seguras baseadas em TCP