

Certificados Digitais e Servidor Web

Prof. Kelvin Lopes Dias

Estagiária Maria Katarine S. Barbosa



Centro de
Informática
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Visão Geral HyperText Transfer Protocol (HTTP)

- Protocolo da camada de Aplicação;
- Arquitetura do tipo cliente-servidor;
 - Cliente - O Chrome e realiza solicitações, recebe e apresenta objetos web;
 - Servidor - Servidor Web envia objetos em respostas às solicitações.
- Utiliza o protocolo de Transporte TCP;
 - Criação de conexões para a realização da comunicação entre o cliente e o servidor.
- Códigos de status de respostas HTTP:
 - 404 Not Found, [\[Mais\]](#).
- Métodos de Requisição:
 - Get - Retorna um objeto;
 - Post - Envia informações para serem armazenadas no servidor, [\[Mais\]](#).

Hands-on: Servidor Web

- Instalação do Servidor Apache;
- Utilizando o telnet para visualizar o formato de uma resposta para uma solicitação HTTP;

```
katarine@katarine:~$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET / HTTP/1.1
Host: localhost

HTTP/1.1 200 OK
Date: Tue, 09 Aug 2022 13:23:35 GMT
Server: Apache/2.4.41 (Ubuntu)
Last-Modified: Tue, 09 Aug 2022 12:41:42 GMT
ETag: "2aa6-5e5ce4121f762"
Accept-Ranges: bytes
Content-Length: 10918
Vary: Accept-Encoding
Content-Type: text/html
```

Hands on 1: Web Server based on HTTP initial

```
from socket import socket, AF_INET, SOCK_STREAM
import htmlMessage
```

```
#1)Criar o socket servidor:
webServerSocket = socket(AF_INET, SOCK_STREAM)
webServerSocket.bind(('localhost', 9696))
webServerSocket.listen()
```

Criação do socket servidor TCP

```
#2)Aceitar as solicitações dos clientes:
for i in range(2):
    print('Esperando Solicitações ...')
    clientSocket, clientAddress = webServerSocket.accept()
    #Recebendo dados do cliente
    data = clientSocket.recv(2048)
    print(f'{data.decode()}')
    #Respondendo a solicitação
    msgHeader = 'HTTP/1.1 200 OK \r\n' \
                'Date: Tue, 09 Aug 2022 13:23:35 GMT\r\n' \
                'Server: MyServer/0.0.1 (Ubuntu)\r\n' \
                'Content-Type: text/html\r\n' \
                '\r\n'
    msgBody = '<html>' \
              '<head><title>Hello, World</title></head>' \
              '<body><h1> Your first web server!</h1>' \
              '<h3>Congratulation!!</h3>' \
              '</body>' \
              '</html>'

    msgHtml = msgHeader + msgBody
    #msgHtml = htmlMessage.sucesso()
    #msgHtml = htmlMessage.NaoEncontrado()

    clientSocket.send(msgHtml.encode())
    clientSocket.close()
```

Aceite das solicitações do navegador web

Geração da mensagem HTML utilizando o protocolo HTTP1.1

Desligando o servidor web

```
webServerSocket.close()
```

Hands on 2: Web Server based on HTTP

```
from socket import socket, AF_INET, SOCK_STREAM

with open('./index.html', 'r') as f:
    index_html = f.read()
with open('./style.css', 'r') as f:
    style_css = f.read()

ss = socket(AF_INET, SOCK_STREAM)
ss.bind(('localhost', 9999))
ss.listen()

while True:
    sc, _ = ss.accept()
    request = sc.recv(1024).decode()
    if request.startswith('GET / HTTP/1.1'):
        reply = 'HTTP/1.1 200 OK\n\n' + index_html
    elif request.startswith('GET /style.css HTTP/1.1'):
        reply = 'HTTP/1.1 200 OK\nContent-Type: text/css\n\n' + style_css
    else:
        reply = 'HTTP/1.1 404 Not Found\n\nPage not found.'

    sc.send(reply.encode())
    sc.close()

ss.close()
```

Conexão persistente ou não persistente?



Hands on 3: Web Server based on HTTP simple

```
from http.server import HTTPServer, BaseHTTPRequestHandler

class handleRequest(BaseHTTPRequestHandler):
    def do_GET(self):
        self.send_response(200)
        self.send_header('content-type', 'text/html')
        self.end_headers()
        self.wfile.write(self.path.encode())

httpServer = HTTPServer(('localhost',9090), handleRequest)
print('O servidor está ativo!')
httpServer.serve_forever()
```

Segurança

Princípios de Segurança em Redes de Computadores

- **Confidencialidade;**
 - A informação só pode ser compreendida por usuários autorizados;
 - Uso de criptografia;
- **Autenticidade;**
 - Garante a veracidade e a autoria da informação;
 - Uso de assinatura e certificados digital;
- **Integridade;**
 - A informação só poderá ser modificada por usuários autorizados;
 - Uso de assinatura digital;
- **Disponibilidade;**
 - A informação estará sempre acessível e disponível para o usuário;
 - Realização de backups.

Princípios de Criptografia

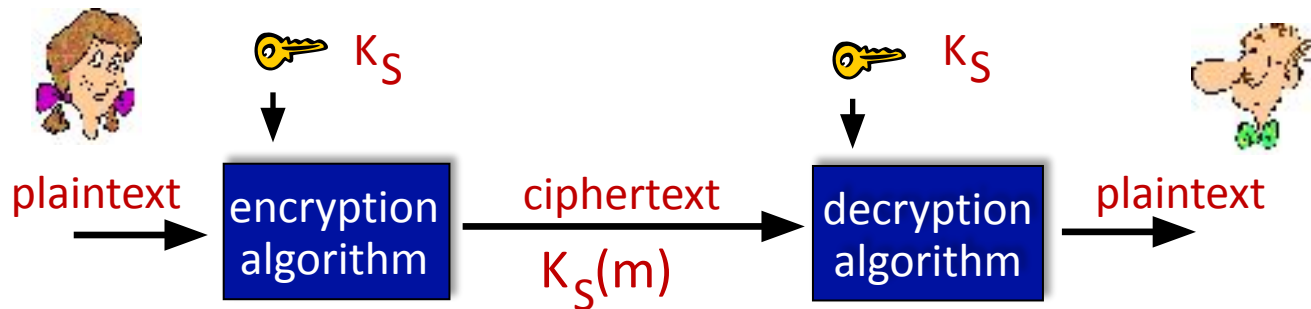
- Criptografia de chave Simétrica:

- Princípio baseado no esquema de chave **única**;
- O tamanho do texto criptografado é o mesmo ou **menor** que o texto original;
- O processo de criptografia é rápido;
- Fornece **confidencialidade**;
- Os tamanhos das chaves podem variar entre 128 a 256 bits;
- AES;

- Criptografia de chaves Assimétricas:

- Utiliza **duas** chaves, uma chave **pública** e outra **privada**;
- O tamanho do texto criptografado é o mesmo ou **maior** que o texto original;
- O processo de criptografia é **lento**;
- Fornece **confidencialidade, autenticidade e não repúdio**.
- O tamanho da chave depende do tipo do algoritmo utilizado;
- Diffie-Hellman, ECC e RSA

Symmetric key cryptography

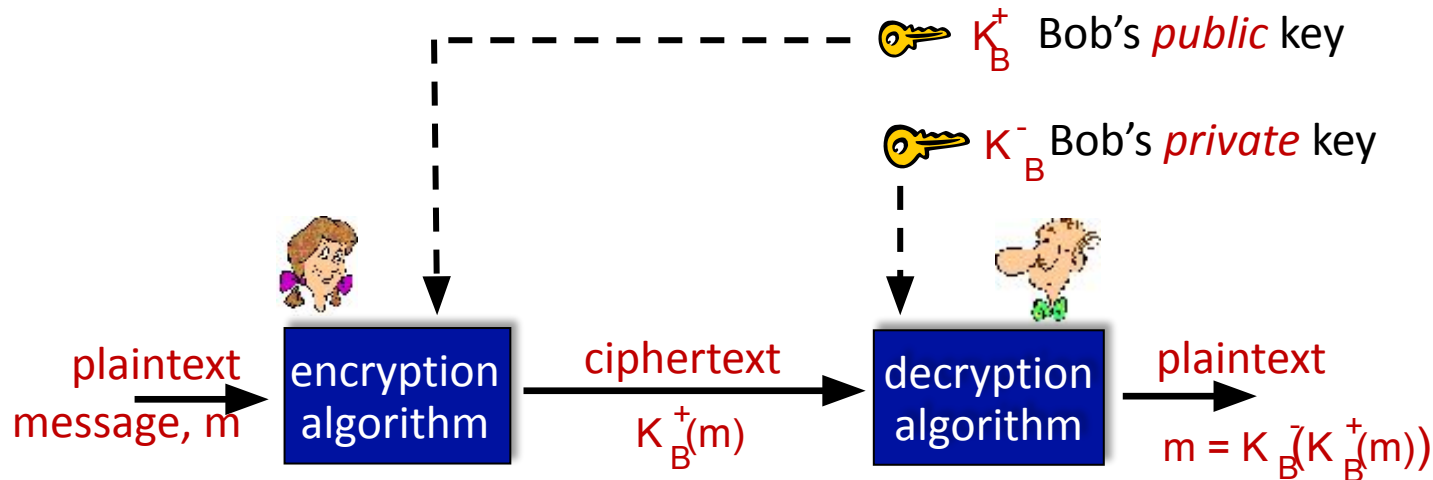


symmetric key crypto: Bob and Alice share same (symmetric) key: K

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher

Q: how do Bob and Alice agree on key value?

Public Key Cryptography



Security: 8- 11

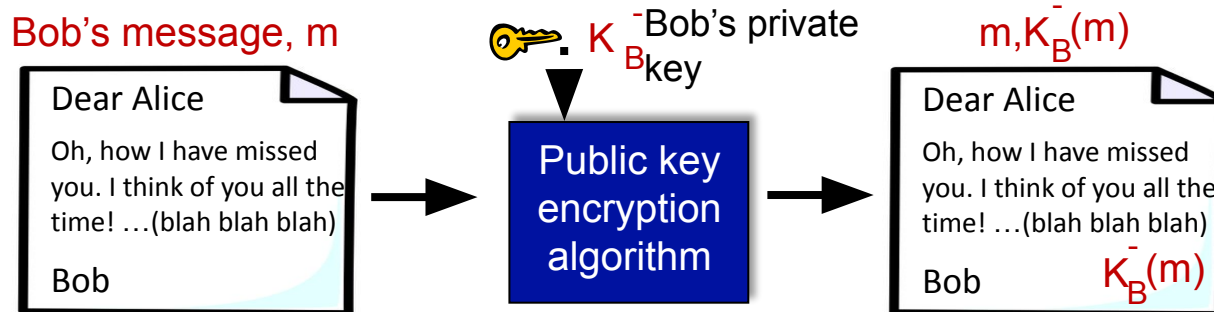
Wow - public key cryptography revolutionized 2000-year-old (previously only symmetric key) cryptography!

- similar ideas emerged at roughly same time, independently in US and UK (classified)

Digital signatures

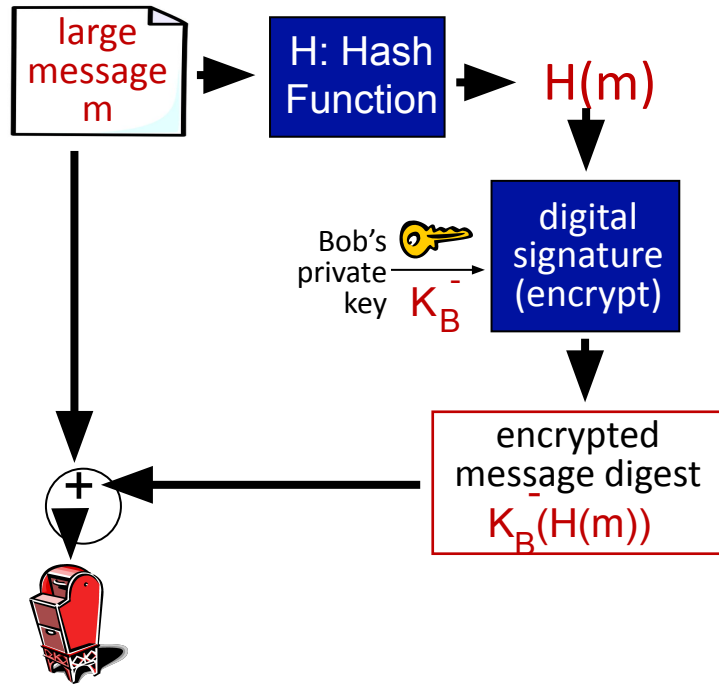
cryptographic technique analogous to hand-written signatures:

- sender (Bob) digitally signs document: he is document owner/creator.
- *verifiable, nonforgeable*: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document
- simple digital signature for message m :
 - Bob signs m by encrypting with his private key K_B^- , creating “signed” message, $K_B^-(m)$

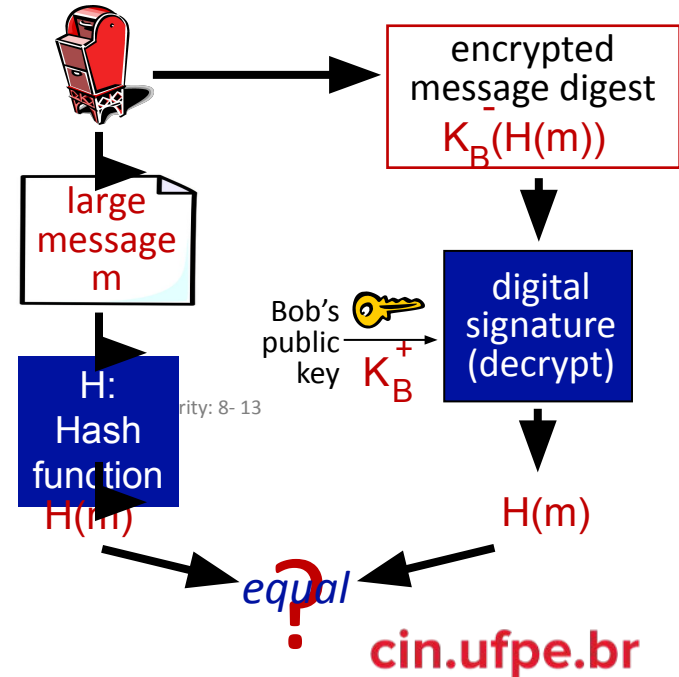


Digital signature = signed message digest

Bob sends digitally signed message:

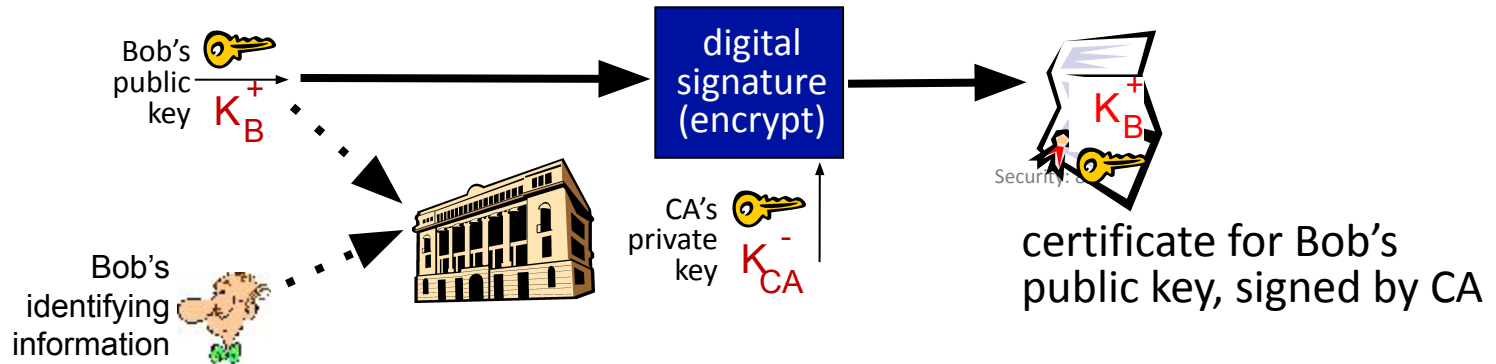


Alice verifies signature, integrity of digitally signed message:



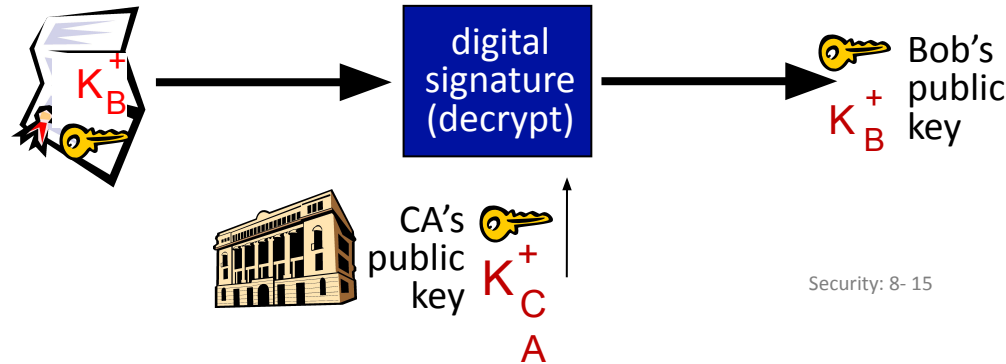
Public key Certification Authorities (CA)

- **certification authority (CA):** binds public key to particular entity, E
- entity (person, website, router) registers its public key with CE provides “proof of identity” to CA
 - CA creates certificate binding identity E to E’s public key
 - certificate containing E’s public key digitally signed by CA: CA says “this is E’s public key”



Public key Certification Authorities (CA)

- when Alice wants Bob's public key:
 - gets Bob's certificate (Bob or elsewhere)
 - apply CA's public key to Bob's certificate, get Bob's public key



Hands on 4: Key Generator

- Chave Simétrica:

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

data = b'hello world'

key = get_random_bytes(16)

print(f'Key: {key}')

cipher = AES.new(key, AES.MODE_EAX)
ciphertext, tag = cipher.encrypt_and_digest(data)

print(f'Cipher = {ciphertext} - size is {len(ciphertext)} bytes')

file_out = open("encrypted.bin", "wb")
[ file_out.write(x) for x in (cipher.nonce, tag, ciphertext) ]
file_out.close()

file_in = open("encrypted.bin", "rb")
nonce, tag, ciphertext = [ file_in.read(x) for x in (16, 16, -1) ]
file_in.close()

# let's assume that the key is somehow available again
cipher = AES.new(key, AES.MODE_EAX, nonce)
data = cipher.decrypt_and_verify(ciphertext, tag)

print(f'Data is back {data}')
```

```
katarina@katarina: ~$ python3 1.py
Key: b'+\xeb\xeb\x1d\xbc\xdb\x86\xaf\x12?d\x13 '
Cipher = b'\xcdIKy\xc1\x02\xe3'\xba8\xd7' - size is 11 bytes
Data is back b'hello world'
```

- Chave Assimétrica:

```
import rsa

pubkey, privkey = rsa.newkeys(512)

str1 = "hello world"

enctex = rsa.encrypt(str1.encode(), pubkey)
dectex = rsa.decrypt(enctex, privkey).decode()
print("The primordial string: ", str1)
print("The Encrypted message: ", enctex)
print("The Decrypted message: ", dectex)

print(f'0 tamanho da criptografia é {len(enctex)} bytes')
```

```
1 The primordial string: hello world
2 The Encrypted message: b"! \xeb\xdb\xfd\xe2Ll0\xcc
3 The Decrypted message: hello world
4 0 tamanho da criptografia é 64 bytes|
```


Transport-layer security (TLS)

- Pesquisa inicial, implementação: programação de rede segura, soquetes seguros:
 - Secure Socket Layer (SSL) obsoleta [2015]
 - TLS 1.3: RFC 8846 [2018]
- Fornece:
 - Confidencialidade - por meio de criptografia simétrica;
 - Integridade - por meio de criptografia hashing;
 - Autenticação - por meio da criptografia baseada em chaves públicas.
- Cipher suite:
 - Algoritmos que podem ser usados para a geração das chaves criptográficas e assinaturas digitais.
- Procedimentos:
 - Handshake - Utilização dos certificados, chaves privadas para se autenticarem e criarem um segredo compartilhado.
 - Key derivation - Utilizam o segredo compartilhado para gerarem as chaves;
 - Data transfer - Fluxo de dados;
 - Connection closure - Mensagens especializadas para garantir que a conexão será encerrada de maneira segura.

TLS - Handshake

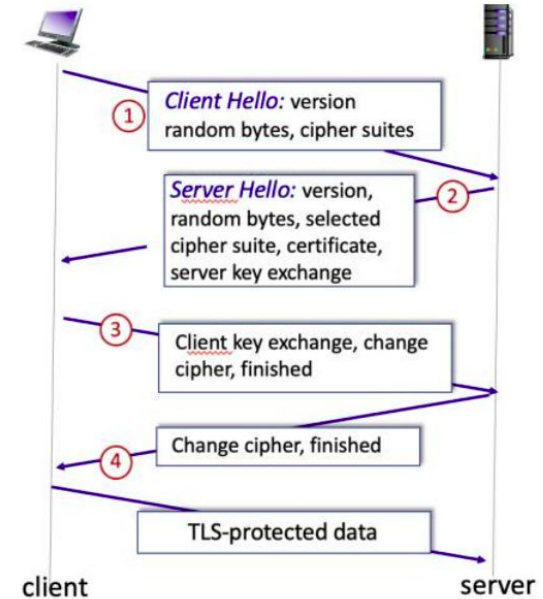
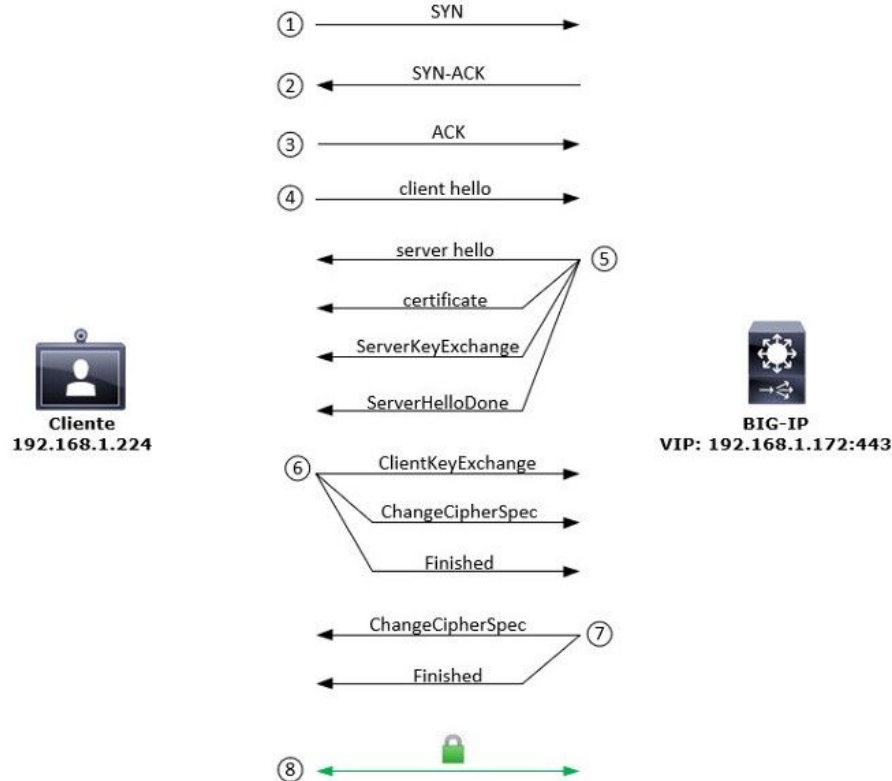
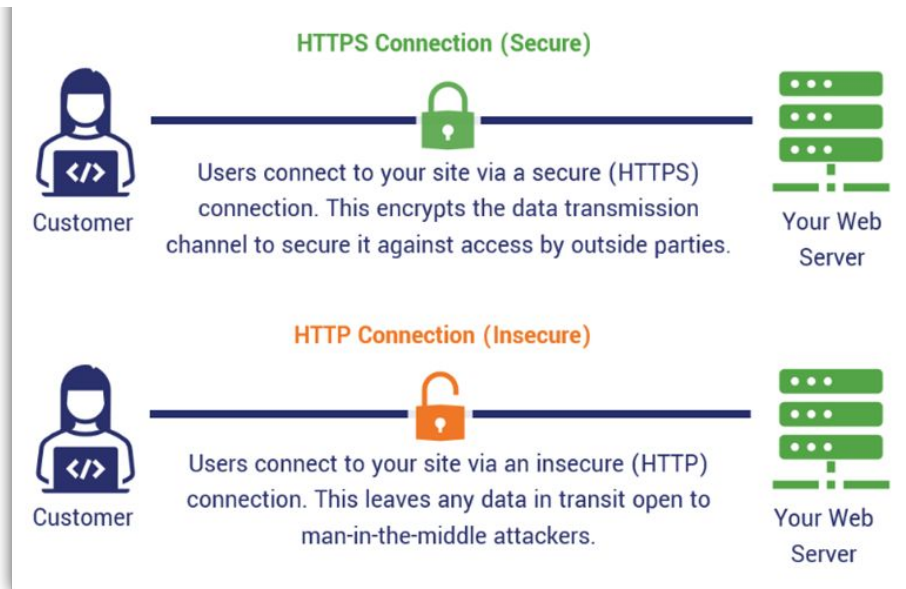
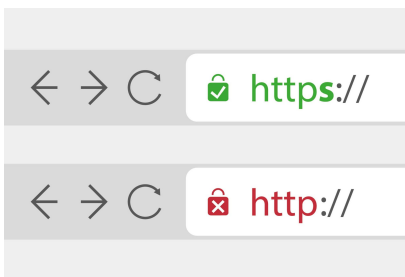


Figure 2: TLS handshaking

HTTP vs. HTTPS

- Hypertext Transfer Protocol Secure (HTTPS);
- HTTPS, (utiliza a porta 443 como padrão);
- É mais lento que o protocolo http;
- Precisa de certificados digitais.



HTTPs Errors



Your connection is not private

Attackers might be trying to steal your information from [redacted] (for example, passwords, messages, or credit cards). [Learn more](#)

NET::ERR_CERT_AUTHORITY_INVALID

☐ Automatically send some [system information and page content](#) to Google to help detect dangerous apps and sites. [Privacy policy](#)

HIDE ADVANCED

Back to safety

This server could not prove that it is [redacted]; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

Proceed to 172.16.16.16 (unsafe)



Your connection is not private

Attackers might be trying to steal your information from [redacted] (for example, passwords, messages or credit cards). [Learn more](#)

NET::ERR_CERT_DATE_INVALID

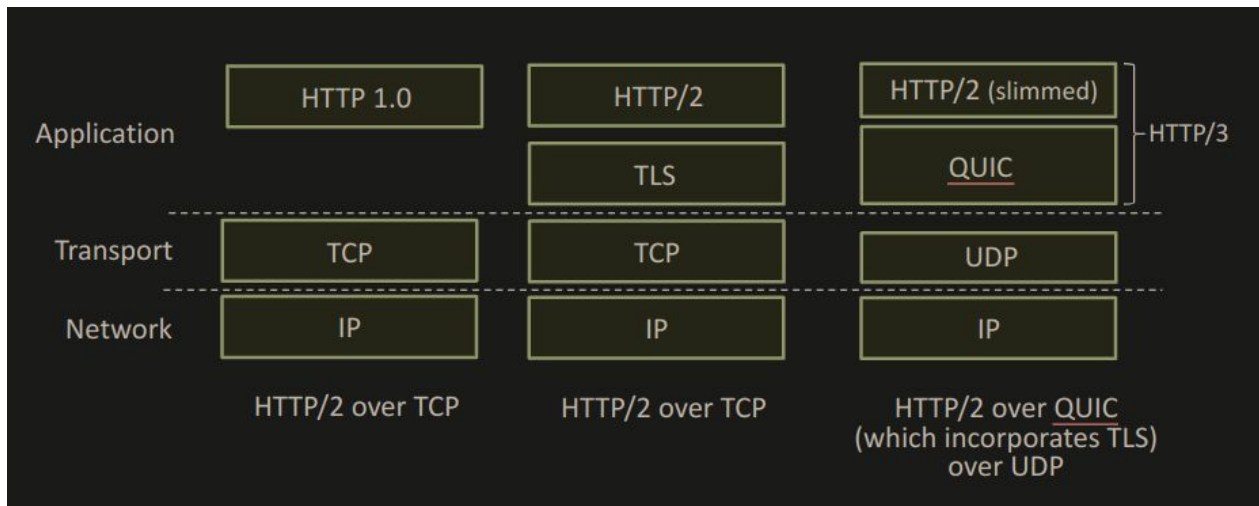
☐ Automatically send some [system information and page content](#) to Google to help detect dangerous apps and sites. [Privacy Policy](#)

ADVANCED

Back to safety

TLS API

- TLS fornece uma API que qualquer aplicativo pode usar
- Uma visão HTTP do TLS:



Hands on 5: HTTPS + Certificados

```
import http.server
import ssl

httpd = http.server.HTTPServer(('localhost', 443),
http.server.SimpleHTTPRequestHandler)
httpd.socket = ssl.wrap_socket (httpd.socket,
certfile='./certificate.pem', server_side=True, ssl_version=ssl.PROTOCOL_TLS)
httpd.serve_forever()
```

-----BEGIN CERTIFICATE-----

```
MIIDXzCCAkcCFGcDmoMDQBL3a0U+8XuDMczSm+BMA0GCSqGSIb3DQEBCwUAMGwx
CzAJBgNVBAYTAKJSMRMwEQYDVQQIDApQZXJuYVlidlWNvMQ8wDQYDVQQHDAZSZWNp
ZmUxYjAUBgNVBAoMDUF1bGEgZGUgUmVhZDZlZmVhZDZlZmVhZDZlZmVhZDZlZmVh
Y2luLnVmcGUUyYnIwHhcnMjMwMjE0MjAwNjQ3Wjc0MjAwNjQ3WjB5MjAwNjQ3Wj
CQYDVQQGEwJCUjE0MjAwNjQ3WjB5MjAwNjQ3WjB5MjAwNjQ3WjB5MjAwNjQ3WjB5
MRYwFAYDVQQKDA1BdWxhIGRlIFJlZGVzMR8wHQYJKoZIhvcNAQkBFhBta3NiQGnp
bi51ZnB1LmJyMIIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAzYtIlbj9
ArymVSA5G2zVWZtxXjl0hj0rTiD1CbBjCMHTukrAXvHCHFBMseBOHnJKvpNbUQUc
HTHziv0gYpXlzug4ZZGjPJ9jWcoYlhvovW9wFQ9U+/1bHLH3aj8fGjp2fAlQtRk2
35Ak/PAuWVI8Hw+HyKdZQWC5dlc5igT2/NdgF6ec2H5Dy0IKi5SU5TxkVUHakHIk
IvcwSGjK2EBwZgn3CjTRNvzZggo9pMVSVcGXXbbTdlEjh2IzUdr5oRdWY5LSHIDW
CJw50oyqstf6wNp8FoPKBcwrLSUPfBFX0uVfQBbMGYd8AasJHKMPHNLrWU3BeYME
PzV8XzBTrqaIwwIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQB5Vg24SDzbcIwWZpZG
96cx6NhrUo4TP0nlywji3KkxP8Lnc0v1/syCNRq0sIkXTUdjR6YhWDJ0/6/zZTV
ewL9jme7H0ddbBHUQFy6ic/iq/8GmkpFAPqm5i0fZgk60Z+WQkARI+tC9dR8Q/
GqUqkmj4o9GgT/0XMMXhrwqtCqTHInf/UyjcQiQf5qn+HPIdtHH4EblxedRlCpF
Hj8KUQ8wYZxzcIzt2/Vlz+1v1uiKs1+7djP7EM0QI0PqlXp+ez0tCtCBBYUsP3S6
YTIh30dra9GlgfiTR8K+j4BKcpx05xCadnSfCb0oLLJUIHWarQP7qmLR8Y2AbDd
+Z27
```

-----END CERTIFICATE-----

Referências

[1] KUROSE, J. F. e ROSS, K. - Computer Networking- A Top-Down Approach - 8ª Ed., chapters 2 e 8, Pearson, 2022.

Dúvidas ?



Certificados Digitais e Servidor Web

Prof. Kelvin Lopes Dias

Estagiária Maria Katarine S. Barbosa