Het eerste blok wordt geïnitialiseerd met een random Initialization Vector IV. Er volgt een XOR logische operatie waar datablok 1 wordt vergeleken met IV. Het resultaat is true wanneer datablok 1 gelijk is aan IV. Daarna wordt het datablok geëncrypt met een symmetrische key. Er volgen XOR logische operaties tussen het vorige cipherblok en het volgende datablok.

Een voorbeeld van Chain Block Chaining Mode (CBC) is het bitcoin-netwerk waar iedere gebruiker een kopie van alle transacties heeft. Chain Block Chaining werkt fraude tegen. Als iemand een transactie in een block wijzigt, komen zijn transacties niet meer overeen met die bij de andere gebruikers. We kunnen het sha256-algoritme in het terminalvenster als volgt gebruiken:

```
echo -n 'Alice stuurt Bob 1 bitcoin.' | openssl sha256
```

geeft als resultaat de volgende hash:

```
13ebb6e60780ae2e1d424f6917730d04b3c5a0261b0114ce8dc6d2b59e31bc2e
```

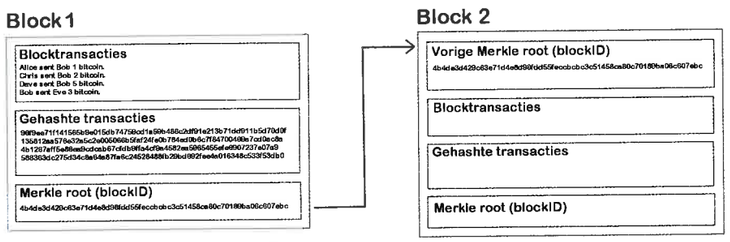
Maar als je de transactie verandert in:

```
echo -n 'Alice stuurt Bob 1.5 bitcoins.' | openssl sha256
```

geeft dat als resultaat een totaal andere hash:

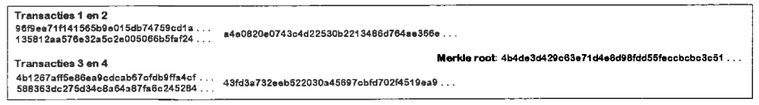
```
c0114cbedb2ad13c4fc635d97c6cbee2c0908bfdce67d99939bdecdac8038d60
```

Een block bestaat uit drie delen: de transacties, de hashes van elke transactie en een hash van het gehele blok. Zie de volgende figuur:



Figuur 3.14 Chain-Block-algoritme

De block-ID of Merkle root is de hash van alle hashes van de transacties. Bijvoorbeeld: je hasht de hash van transacties 1 en 2 en de hash van transacties 3 en 4 en je krijgt de block-ID. Zie de volgende figuur:



Figuur 3.15 De Merkle root

Wanneer je een tweede block wilt toevoegen dan doe je de (Chaining) met de block-ID.

3.2.3 Authenticated Encryption (AE)

Zoals eerder aangegeven: encryptie zorgt ervoor dat cipher text niet te herleiden is tot de plain text. We spreken dan over de eis van vertrouwelijkheid: alleen geverifieerde gebruikers mogen encrypten en decrypten.

Andere eisen die gesteld wordt aan encryptie-algoritmes:
Authenticatie: de identiteit van de verzender en ontvanger moet geverifieerd worden.
Integriteit: de informatie onderweg kan niet gecorrumpeerd worden.

Authenticated Encryption voorziet naast vertrouwelijkheid ook in integriteit en authenticatie.
Voor AE wordt gebruikgemaakt van een algoritme dat een Message Authentication Code, kortweg MAC, oplevert. Komt de MAC van de zender van het bericht overeen met de MAC van de ontvanger, dan is het bericht authentiek.

De componenten in het proces zijn:

Authenticated Encryptie
- Input: brontekst, key
- Output: cipher text en Message Authentication Code (MAC)

Authenticated Decryptie
- Input: cipher text, key, MAC
- Output: brontekst of een foutmelding wanneer de MAC niet hoort bij de cipher text.

De volgende figuur laat een Authenticated Encryption-proces zien.

Op beide begrippen gaan we hier niet verder in.
Hier hebben we een 128-bits key gegenereerd. Om 192-bits en 256-bits keys te genereren gebruiken we de volgende opties:

```
-aes-192-cbc
-aes-256-cbc
```

Stap 2: Maak het **plaintext.txt**-bestand met de tekst 'hello world'. Gebruik het **echo/echo**-commando als volgt:

```
$ echo "hello world" > plaintext.txt
$ cat plaintext.txt

hello world
```

Met het **cat/type**-commando tonen we de inhoud van het **plaintext.txt**-bestand.

Stap 3: In deze stap encrypt je het **plaintext.txt**-bestand met gebruik van de **key** en **iv** uit stap 1. Doe dit als volgt:

```
$ openssl enc -e -aes-128-cbc

-K 9D025DA55DC260E9C743121221713228

-iv 5DEF6BC571495841DC32F9B996F0D14

-in plaintext.txt -out streamcipher.enc
```

Voor encrypten gebruik je de **-e**-optie. De output is het **streamcipher.enc**-bestand. Het resultaat ziet er als volgt uit, waarbij regel 2 het resultaat is van regel 1:

```
$ cat streamcipher.enc
ı(??7?I??N??n??
```

Stap 4: In deze stap decrypt je het geëncrypte **streamcipher.enc**-bestand. Doe dit als volgt:

```
$ openssl enc -d -aes-128-cbc

-K 9D025DA55DC260E9C743121221713228

-iv 5DEF6BC571495841DC32F9B996F0D14

-in streamcipher.enc -out plaintext1.txt
```

Voor decrypten gebruik je de **-d**-optie. De output is het **plaintext1.txt**-bestand. Om de inhoud van het **plaintext1.txt**-bestand te tonen typ je het **cat**-commando als volgt in:


```
$ cat plaintext1.txt

hello world
```

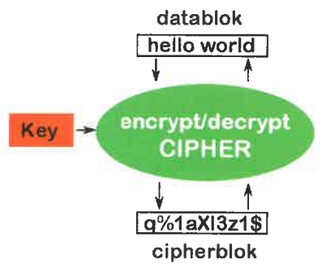
Toon een lijst met de aangemaakte bestanden. Gebruik het **ls/dir**-commando:

```
$ ls
plaintext.txt    plaintext1.txt    streamcipher.enc
```

3.2.2 Block ciphers

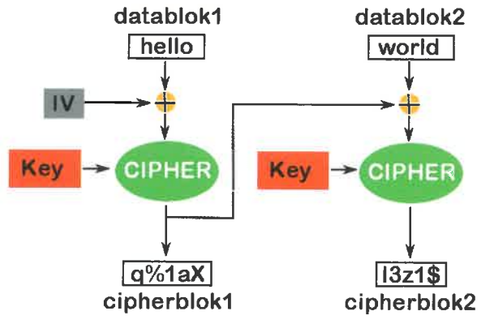
Block ciphers verdelen de data in blokken. Ieder blok wordt geëncrypt. Er zijn meerdere manieren (in het Engels: mode of operation) om te encrypten via block ciphers.

ECB (Electronic Codebook Mode): deze operatiemode leest en encrypt elk datablok afzonderlijk.



Figuur 3.12 Electronic Codebook Mode

CBC (Chain Block Chaining Mode): deze operatiemode gebruikt het vorige cipherblok bij de encryptie van het volgende datablok in een soort kettingproces.



Figuur 3.13 Chain Block Chaining Mode

Input	h							
Plain text byte	0	1	1	0	1	0	0	0
Key byte	1	1	0	0	1	0	1	1
Cipher text byte	1	0	1	0	0	0	1	1
Output	£							

Een link naar binaire, hexadecimale, decimale en ASCII tekstconversies vind je hier:

http://www.rapidtables.com/convert/number/ascii-hex-bin-dec-converter.htm

Voorbeeld 2
In het volgende voorbeeld encrypten we een stream van 12 bytes. Hier geven we de bytes hexadecimaal weer.

Plain text stream	h	e	l	l	o		w	o	r	l	d	!
Key stream	cb	2b	64	b2	b6	a1	59	4a	6e	13	6b	05
Cipher text stream	a3	4e	08	de	d9	81	2e	25	1c	7f	øf	24

Lab 3.2 OpenSSL-tool

In deze lab-opdracht oefenen we met de OpenSSL-tool. De syntaxis van het OpenSSL-commando is als volgt:

```
$ openssl enc -ciphername [options]
```

De opties zijn:

```
-in <file>     input file
-out <file>    output file
-pass <arg>    pass phrase source
-e             encrypt
-d             decrypt
-a/-base64     base64 encode/decode, depending on encryption flag
-k             passphrase is the next argument
-kfile         passphrase is the first line of the file argument
-md            the next argument is the md to use to create a key
               from a passphrase.  One of md2, md5, sha or sha1
-K/-iv         key/iv in hex is the next argument
-[pP]          print the iv/key (then exit if -P)
-bufsize <n>   buffer size
-engine e      use engine e, possibly a hardware device.
-seed-ofb
```



```
Cipher Types
-aes-128-cbc          -aes-128-cfb          -aes-128-cfb1
-aes-128-cfb8         -aes-128-ecb          -aes-128-ofb
-aes-192-cbc          -aes-192-cfb          -aes-192-cfb1
-aes-192-cfb8         -aes-192-ecb          -aes-192-ofb
-aes-256-cbc          -aes-256-cfb          -aes-256-cfb1
-aes-256-cfb8         -aes-256-ecb          -aes-256-ofb
-aes128               -aes192               -aes256
-bf                   -bf-cbc               -bf-cfb
-bf-ecb               -bf-ofb               -blowfish
-cast                 -cast-cbc             -cast5-cbc
-cast5-cfb            -cast5-ecb            -cast5-ofb
-des                  -des-cbc              -des-cfb
-des-cfb1             -des-cfb8             -des-ecb
-des-ede              -des-ede-cbc          -des-ede-cfb
-des-ede-ofb          -des-ede3             -des-ede3-cbc
-des-ede3-cfb         -des-ede3-cfb1        -des-ede3-cfb8
-des-ede3-ofb         -des-ofb              -des3
-desx                 -desx-cbc             -rc2
-rc2-40-cbc           -rc2-64-cbc           -rc2-cbc
-rc2-cfb              -rc2-ecb              -rc2-ofb
-rc4                  -rc4-40               -seed
-seed-cbc             -seed-cfb             -seed-ecb
-seed-ofb
```

De aes-128-cbc-cipher is een stream cipher die te vinden is in je OpenSSL-bibliotheek. In de volgende stappen genereren we een key en encrypten we de plain text 'hello world'. Dit doen we vanaf de commandoregel.

Waar nodig geven we de namen van de Linux- of Windows-commando's als volgt: **linux/windows**. Bijvoorbeeld: **ls/dir**.

Stap 1: Genereer een key, salt en een initialiserende vector (iv). Daarvoor open je het terminalvenster en typ je het volgende in:

```
$ openssl enc -aes-128-cbc -k secret -P -md sha1
```

De optie **-k** is voor het wachtwoord. Het resultaat zal zoiets zijn als:

```
salt=C80035F15F90738AE
key=9D025DA55DC260E9C74312122171322B
iv =25DEF6BC571495841DC32F9B996F0D14
```

'Salting' is een manier van compliceren – en daardoor extra beveiligen – van de hashwaarde van wachtwoorden. Aan de hashwaarde van het wachtwoord worden willekeurige bits toegevoegd, waarna de saltwaarde wordt opgeslagen. Op 'hashing' komen we in hoofdstuk 7 terug.

Een initialiserende vector is een willekeurige reeks die bij een key wordt gebruikt als start van het versleutelen.

de server geeft aan dat alle interactie met browsers plaatsvindt met veilige HTTPS-verbindingen en nooit met het onveilige HTTP. Dit wordt aan de browser doorgegeven via het response-header-veld **Strict-Transport-Security**. Bijvoorbeeld:

```
Strict-Transport-Security: max-age=31536000.
```

31536000 is gelijk aan een jaar, dat bestaat uit 31.536.000 seconden, de periode dat deze instructie bewaard wordt. Het resultaat is dat een link als

```
http://example.com/some/page/
```

automatisch wordt vervangen door de volgende veilige link:

```
https://example.com/some/page/
```

3.2 Encryptie-algoritmes (ciphers)

HTTPS zorgt voor de encryptie en decryptie van de verzonden informatie. Het maakt gebruik van de OpenSSL-bibliotheek van encryptie-algoritmes (ciphers). In deze paragraaf maken we kennis met een aantal encryptie-algoritmes in de OpenSSL-bibliotheek. Daarna kunnen we zelf gebruikmaken van deze algoritmes om bijvoorbeeld statische informatie, zoals bestanden met gevoelige informatie, te encrypten.

Er zijn twee soort ciphers: stream ciphers en block ciphers.

3.2.1 Stream ciphers

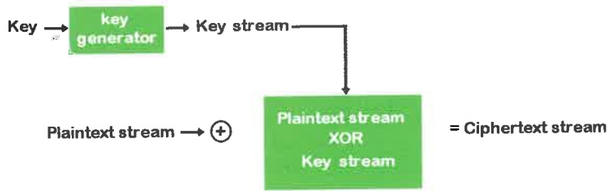
Stream ciphers zijn symmetrische key ciphers. Hiermee encrypten we streams van inputdata bit voor bit. De drie componenten van stream-ciphers zijn:

- Stream key generator
- Encryptfunctie
- Decryptfunctie

Stream ciphers werken met drie datastreams:

- Plain text stream (input)
- Gegenereerde key stream
- Cipher text stream (output)

De key stream wordt gegenereerd door de stream key generator met de symmetrische key als input. De volgende figuur schetst de stream cipher.



Figuur 3.11 Stream cipher

De encryptiefunctie combineert een byte (of algemener gesteld: een element; dit kan ook een bit of letter zijn) van de plain text input stream met een byte van de gegenereerde key stream met als resultaat een byte van de output cipher text stream. De functie is een simpele exclusive-or (XOR) logische operatie tussen de twee bytes.

De truth-tabel van een XOR-operatie zie je hieronder.

A XOR B = Output		
Input		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

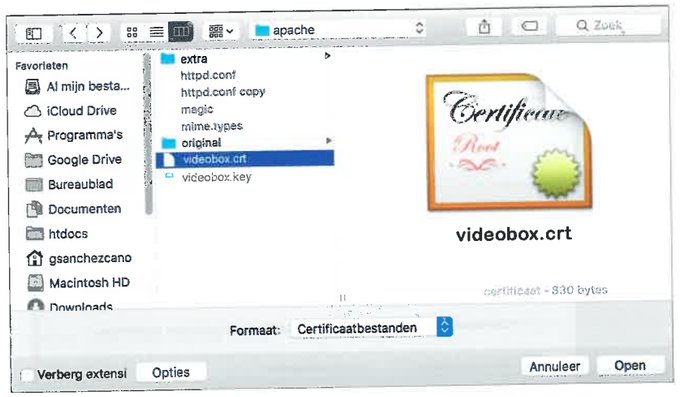
In deze tabel zien we dat als A gelijk is aan B dat de output false is (0).

Decryptie is het omgekeerde van encryptie. De functie combineert een byte van de cipher text stream met een byte van de key stream met als resultaat een byte van de plain text stream. De formule is als volgt:

Cipher text byte[0] = (Plain text byte[0] XOR Key byte[0])

Voorbeeld 1
In dit voorbeeld is de input de letter **h**. De letter h is binair 01101000.
De gegenereerde key stream byte is bijvoorbeeld 11001011. Dan is de cipher text (01101000 XOR 11001011) = 10100011 Dit is het £-teken.

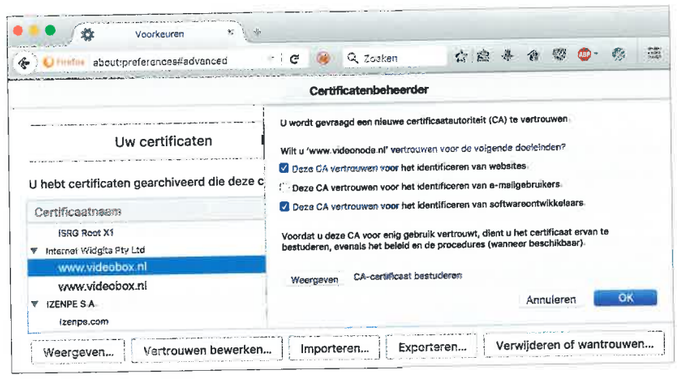
De volgende tabel geeft het encryptieproces weer.



Figuur 3.7 Certificaat importeren

In deze laatste stap laat je met de optie *Open* je browser je certificaat vertrouwen. Vink de volgende twee selectievakjes aan:

- *Deze CA vertrouwen voor het identificeren van websites.*
- *Deze CA vertrouwen voor het identificeren van softwareontwikkelaars.*



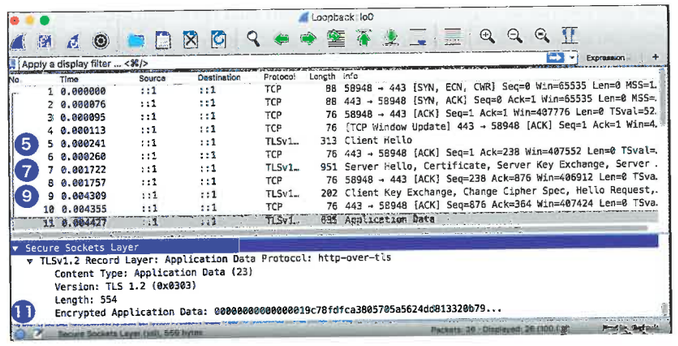
Figuur 3.8 Certificaat vertrouwen

We kunnen nu HTTPS activeren door vanuit je browser naar **https://www.videobox.nl** te gaan. Klik op het groene slotje om te bevestigen dat de verbinding veilig is.

Figuur 3.9 HTTPS activeren

Hier krijg je het bestand van Lab 2.5 *Applications/MAMP/Library/htdocs/videobox/index.php* te zien. Het is de library voor secure https-apps.

Om de encryptie van onze POST-gegevens te controleren open je Wireshark en vang je de data-packets af.



Figuur 3.10 Data-encryptie in Wireshark

- In de data-packets zie je hoe HTTPS correct is uitgevoerd.
- In data-packet 5 maakt de browser verbinding met de server.
- In data-packet 7 krijgt de browser het certificaat en de key van www.videobox.nl.
- In data-packet 9 wordt de asymmetrische key uitgewisseld met de server.
- In data-packet 11 zien **we** dat de data zijn encrypted.

3.1.4 HSTS

HTTP Strict Transport Security (HSTS) is een mechanisme om websites te beschermen tegen aanvallen en voorkomt het stelen van cookies. Dit werkt als volgt:

Verander in het **httpd.conf**-bestand de **Listen**-regel als volgt:

```
#Listen 8888
Listen 80
```

Verander de **ServerName**-regel:

```
#ServerName localhost:8888
ServerName localhost:80
```

Include het **httpd-ssl.conf**-bestand als volgt:

```
# Secure (SSL/TLS) connections
Include /Applications/MAMP/conf/apache/extra/httpd-ssl.conf
```

Stap 7: Wijzigingen in **httpd-ssl.conf**
Het **httpd-ssl.conf**-bestand moet ook geconfigureerd worden.

Verander de **Listen**-regel:

```
#Listen 443
Listen [::]:443
Listen 0.0.0.0:443
```

Verander de **<VirtualHost>**-regel:

```
## SSL Virtual Host Context
<VirtualHost *:443>
#  General setup for the virtual host
DocumentRoot "/Applications/MAMP/Library/htdocs"
ServerName www.videobox.nl:443
ServerAdmin admin@videobox.nl
```

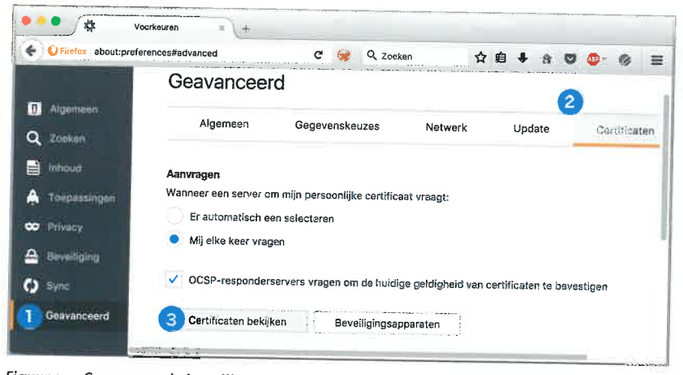
Set de SSLEngine aan en verwijs naar je certificaat en je key:

```
#   Enable/Disable SSL for this virtual host.
SSLEngine on

#   Server Certificate:
SSLCertificateFile "/Applications/MAMP/conf/apache/videobox.crt"

#   Server Private Key:
SSLCertificateKeyFile "/Applications/MAMP/conf/apache/videobox.key"
```

Stap 8: Browserinstellingen
Je certificaat is geen officieel maar een self signed certificaat en het is niet te vinden in de certificatenlijst in je browser. In deze stap:

- Ga je naar de geavanceerde instellingen van je browser.
- Open je de *Certificaten*-tab.
- Klik je op *Certificaten bekijken*.

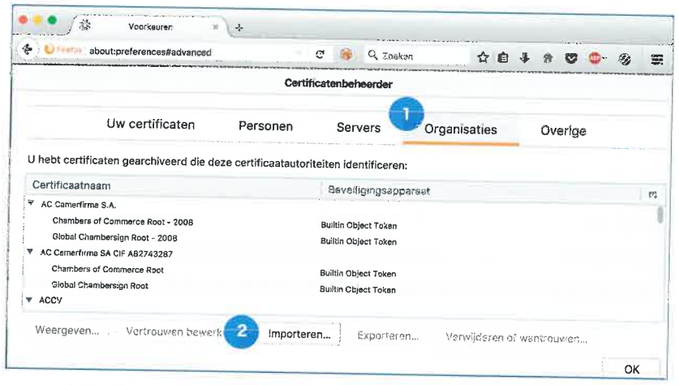
Dit zie je in de volgende figuur:



Figuur 3.5 Geavanceerde instellingen

Hierna importeer je je organisatiedomein (www.videobox.nl) in je browser:

- Klik op *Organisaties*.
- Klik op *Importeren*.



Figuur 3.6 Certificatenbeheerder

Dan importeer je je **videobox.crt**-certificaat vanuit je Apache-map:

Sla het bestand op met **Ctrl-O**.
Sluit af met **Ctrl-X**.

Stap 2: Maak backups voordat je wijzigingen aanbrengt in de volgende bestanden
In deze stap maak je backups van de volgende configuratiebestanden:
MAMP/conf/apache/**httpd.conf**
MAMP/conf/apache/extra/**httpd-ssl.conf**
Stop zo nodig de Apache-server.

Stap 3: Een self signed key genereren
Om HTTPS in je eigen localhost te implementeren genereren we een self signed key. Hiervoor gebruiken we het programma **openssl**. Het programma kan gedownload worden vanaf de website van Open SSL:

https://www.openssl.org/

Op je laptop is waarschijnlijk het programma **openssl** al geïnstalleerd. (Zo niet, ga naar **www.openssl.org** en installeer **openssl** alsnog.)

Open je terminalvenster en voer het volgende uit:

```
$ openssl genrsa -des3 -out videobox.key 1024
```

Typ tweemaal hetzelfde wachtwoord.
Het resultaat is de key **videobox.key** in je usersmap.

Stap 4: Genereer een certificaataanvraag
Een certificaataanvraag heet een Certificate Signing Request (CSR). Hier vullen we informatie in over het bedrijf dat het certificaat aanvraagt. In de volgende tabel zie je de nodige uitleg.

Veld	Betekenis	Voorbeeld
City/Town/Locality	Vestigingsplaats van de organisatie	Amsterdam
State/Province	Vestigingsprovincie	Noord-Holland
Organization	Organisatienaam	VideoBox B.V.
Unit	Afdeling	ICT
Common Name	Hostnaam	www.videobox.nl
Country Name	Tweeletterige (ISO) landcode	NL
E-mail	Van serveradministrator	admin@videobox.nl
Password	Leeg laten	
Extra Name	Leeg laten	
Bit Length	Minimaal 2048	2048

Tabel 10 SSL-certificaataanvraag

Een officieel certificaat kun je bij een Certificate Authority online aanvragen. Bijvoorbeeld via de volgende link:

https://www.sslcertificaten.nl/support/FAQ/Overzicht_CSR-velden

In ons geval willen we een self signed certificaat genereren voor testdoeleinden. Dat betekent dat we alle informatie in het formulier leeg kunnen laten, behalve Common Name. Hier typen we onze domeinnaam: **www.videobox.nl**. Genereer een self signed certificaat **als** volgt:

Open het terminalvenster en voer het volgende uit:

```
$ openssl req -new -key videobox.key -out videobox.csr
```

Gebruik hetzelfde wachtwoord als in stap 3.
Het resultaat is de aanvraag **videobox.csr** in je usersmap.

Je verwijdert het wachtwoord in **videobox.key** als volgt:

```
$ cp videobox.key videobox.tmp
```

en daarna:

```
$ openssl rsa -in videobox.tmp -out videobox.key
```

Stap 5: Genereer een certificaat
Hier maken we de stap van request (**videobox.csr**) naar certificaat (**videobox.crt**).
Om het certificaat te genereren open je je terminalvenster en voer je het volgende uit:

```
$ openssl x509 -req -days 365 -in videobox.csr -signkey videobox.key -out
videobox.crt
```

Gebruik hetzelfde wachtwoord als in stap 3.
Het resultaat is het certificaat **videobox.crt** in je usersmap.

Verplaats key en certificaat naar de MAMP-configuratie:

```
$ cp videobox.crt /applications/mamp/conf/apache/videobox.crt
$ cp videobox.key /applications/mamp/conf/apache/videobox.key
```

Je kunt deze bestanden ook gewoon kopiëren en plakken. De certificaat- en key-bestanden moet je op een veilige plek opslaan (die niet toegankelijk is voor hackers).

Stap 6: Wijzigingen in httpd.conf
De laatste stap is het veranderen van een aantal regels in de configuratiebestanden.

1 Je surft naar je Gmail.

2 Gmail-server stuurt het Gmail-certificaat met daarin Gmails (rode) publieke key.

3 Je browser verifieert Gmails certificaat en (rode) publieke key.
   - Je browser genereert een unieke (groene) symmetrische key en encrypt deze vervolgens met gebruik van Gmails (rode) publieke key. Je browser stuurt de geëncrypte symmetrische key naar Gmail.
   - Gmail decrypt de symmetrische key met gebruik van Gmails (zwarte) privé-key. Resultaat is dat je browser en Gmail nu allebei dezelfde unieke (groene) symmetrische key bezitten.

4 Einde van handshake. Begin veilige verbinding.

5 Alle communicatie tussen je browser en Gmail wordt vanaf nu geëncrypt en gedecrypt met de (groene) symmetrische key.

Bij stap 2 van de SSL-handshake krijgt je browser het certificaat van de server en de browser verifieert het certificaat door in de eigen lijst van betrouwbare certificaten te zoeken. Ga naar de geavanceerde instellingen van je browser en klik op de *Certificaten*-tab om een lijst van certificaten in je browser te zien. Het certificaat van Gmail zie je onderaan in de volgende figuur:

Figuur 3.3 Certificaten in je browser

Een SSL-certificaat is een digitaal bestand met een cryptografische key en informatie over de gecertificeerde organisatie. Het activeert het HTTPS-protocol via poort 443 voor een veilige verbinding tussen de webserver en de browser.

3.1.3 OpenSSL

OpenSSL is een open source cryptografie-toolkit met een cryptografie-library. De website van OpenSSL vind je hier:

https://www.openssl.org/

Servers en browsers maken gebruik van de OpenSSL cryptografie-library met actuele ciphers. Ciphers zijn encryptie-algoritmes. Tijdens de SSL-handshake bepaalt de server welke cipher voor de encryptie gebruikt wordt.

Lab 3.1 HTTPS-protocol activeren

In deze lab-opdracht voeren we de benodigde stappen uit voor het activeren van het HTTPS-protocol voor je eigen localhost Apache-server.

Stap 1: Localhost DNS-record
Bij elk domein (webapplicatie) staat een record ingesteld met de naam *localhost* en het IP-adres *127.0.0.1*. Localhost verwijst naar de huidige server in een netwerk. Dit record is een loopback-interface voor localhost en kan gebruikt worden door applicatieontwikkelaars om hun applicaties in de eigen server te testen zonder echt publiek IP-adres.

Open je terminalvenster en voer het volgende uit:

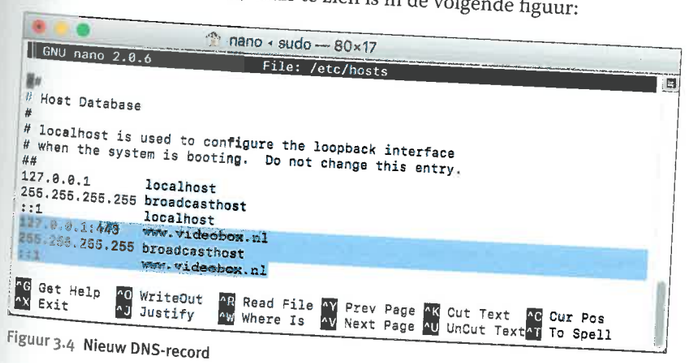
```
$sudo nano /etc/hosts
```

Er is een gratis Sudo voor Windows softwareprogramma met alle functionaliteit van de Sudo-commando's van Linux. Sudo voor Windows kun je installeren vanaf de volgende link:

http://blog.lukesampson.com/sudo-for-windows

Met de **nano**-editor openen we het bestand **hosts**. Hier zien we een DNS-record met het IP-adres voor de domein-naam **localhost**.

Voeg er een nieuw DNS-record aan toe met de nieuwe testdomeinnaam **videobox**. De drie gemarkeerde regels vormen het nieuwe DNS-record. VideoBox is de naam van de webapplicatie die we in het laatste hoofdstuk gaan ontwikkelen. Maak het nieuwe DNS-record aan, zoals te zien is in de volgende figuur:



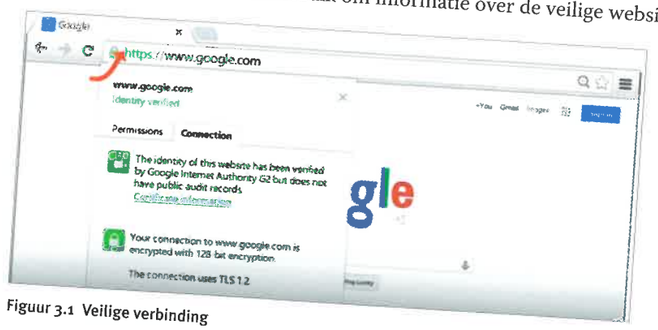
Figuur 3.4 Nieuw DNS-record

van de publieke key van de ontvanger. De versleutelde tekst kan nu alleen met behulp van de geheime key van de ontvanger ontcijferd worden. De geheime keys worden nooit gedeeld.

3.1 HTTPS en SSL

In paragraaf 2.5 hebben we bij de behandeling van de OSI-lagen ook verschillende protocollen behandeld, waaronder Hypertext Transfer Protocol Secure (HTTPS). De S in het protocol betekent dat een veilige laag met TLS/SSL is toegevoegd aan HTTP. HTTPS maakt gebruik van asymmetrische **en** symmetrische cryptografie.

In de volgende figuur zien we een HTTPS-verbinding met **google.com**. Hier kun je klikken op HTTPS in de adresbalk om informatie over de veilige website te zien.



Figuur 3.1 Veilige verbinding

We gaan eerst dieper in op TLS/SSL. De afkorting TLS staat voor Transport Layer Security (en dit is de oude naam). SSL staat voor Secure Socket Layer. SSL is de nieuwe naam, die we verder gebruiken.

3.1.1 SSL

SSL is de extra beveiligingslaag die een HTTPS-verbinding veilig maakt. Om SSL goed te laten functioneren hebben we een aantal technologieën nodig.

- TCP/IP-netwerkverbinding tussen server en client.
- HTTP voor het versturen van bestanden.
- Certificate Authority (CA) voor het leveren van digitale certificaten.

Het bepalen van de authenticiteit van de server loopt via deze digitale certificaten. Je organisatie vraagt een certificaat aan bij een Certificate Authority zoals Verisign. Een Certificate Authority is een betrouwbare organisatie die digitale certificaten uitreikt voor het identificeren van geregistreerde internetservers. Feitelijk: dat certificaat getuigt dat de publieke sleutel, vervat in het certificaat, toebehoort aan de persoon/organisatie/server die in het certificaat vermeld wordt.

Voor een SSL-verbinding heb je een SSL-certificaat nodig. Een SSL-certificaat bevestigt dat de content onderweg tussen je server en de browser afkomstig is van een gecertificeerd domein. Het zegt dat de pagina geëncrypt en veilig is en onderweg niet gedecrypt kan worden door een tussenpersoon.

Je kunt een SSL-certificaat kopen bij onder andere de volgende autoriteiten:

- **comodo.com**
- **gogetssl.com**

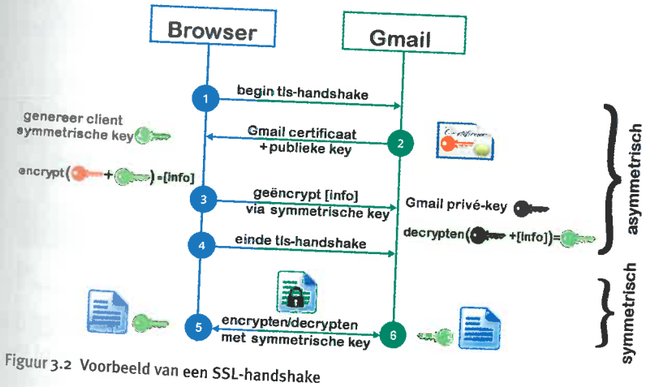
Door je certificaat op je server te plaatsen kun je het TLS-protocol gebruiken. Dit protocol is veiliger want de content wordt geëncrypt en bij aankomst in de browser gedecrypt.

Samenvattend: We gebruiken SSL voor authenticatie tussen de server en de cliënt en verder om dataverkeer te encrypten. Het SSL-protocol zorgt voor een veilige HTTPS-verbinding tussen de webserver en de browser en gebruikt daarbij poort 443.

Het protocol maakt handig gebruik van een TLS/SSL-handshake. De handshake wordt uitgevoerd met asymmetrische encryptie. Na de handshake wordt er een veilige verbinding gemaakt met symmetrische encryptie.

3.1.2 SSL-handshake

In de volgende figuur zien we hoe HTTPS gebruikmaakt van een SSL-handshake.



Figuur 3.2 Voorbeeld van een SSL-handshake

We gebruiken gekleurde keys om het verschil uit te leggen. De kleur heeft verder geen inhoudelijke betekenis. Hier volgt een uitleg van een voorbeeld van een SSL-handshake en een veilige verbinding:

nummers 1 tot en met 65535. Zo gebruikt TCP bijvoorbeeld poort 80 voor HTTP en poort 25 voor SMTP. UPD gebruikt bijvoorbeeld poort 53 voor DNS.

De Network Layer gebruikt IP om de data-packets van verzender naar ontvanger te sturen en over de verschillende netwerken te routeren.

De Data Link en Physical Layers worden gebruikt om de data-packets over het fysieke medium (koper, glasvezel of draadloos) te verzenden en zijn belangrijk voor de netwerkbeheerder, maar minder belangrijk voor de programmeur.

Hoofdstuk 3
Cryptografie

De term cryptografie komt uit het Grieks en betekent geheimschrift. Cryptografie is het geheim maken en houden van informatie door deze te versleutelen. Dit versleutelen noemen we *encryptie*.

Encryptie zorgt ervoor dat normale tekst (plain text), wanneer je deze verstuurt, voor de (al of niet bedoelde) ontvanger onleesbaar is. De versleutelde versie van de plain text noemen we cipher text.

Alleen de persoon voor wie de tekst bedoeld is, weet hoe de tekst weer leesbaar gemaakt moet worden. Dit laatste, het weer leesbaar maken, noemen we *decryptie*. In plaats van de termen encrypten en decrypten gebruiken we ook de tegenstelling coderen en decoderen.

Encryptie in zijn simpelste vorm kan het afspreken van codewoorden zijn. De code kan een eenvoudig woord zijn of een algoritme. De code wordt sleutel of key genoemd.

Leerdoelen

Aan het einde van dit hoofdstuk zou je de kennis en vaardigheden over de volgende onderwerpen moeten beheersen.

- specialistische kennis van cryptografie;
- cryptografie toepassingen om een applicatie te beveiligen;
- OpenSSL encryptiebibliotheek;
- encryptie-algoritmes of ciphers;
- HTTPS.

Er zijn twee soorten cryptografie.

Symmetrische cryptografie: de verzender en de ontvanger spreken af welke geheime key (sleutel) gebruikt wordt voor encryptie en decryptie. Omdat de verzender en de ontvanger dezelfde key moeten delen, is het mogelijk dat de key tijdens deze communicatie onderschept wordt door hackers.

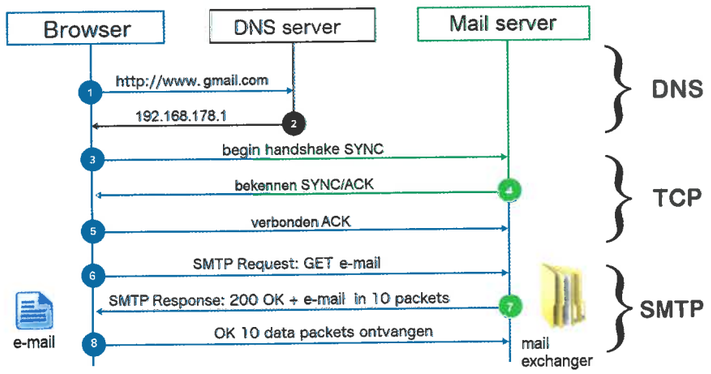
Asymmetrische cryptografie: maakt gebruik van publieke en geheime keys. Encrypties met de publieke key kunnen alleen met de geheime key gedecrypt worden. Hier delen dus de verzender en de ontvanger hun publieke keys met elkaar, maar niet hun geheime keys. Bijvoorbeeld, de verzender encrypt een tekst met behulp

In hoofdstuk 6 gaan we met echte MiM-aanvallen oefenen met de Burp-pentesten-tool. De Burp-tool speelt de rol van de Man in the Middle.

2.5 SMTP-protocol

Het acroniem SMTP staat voor Simple Mail Transfer Protocol. Het is het standaardprotocol voor het versturen van e-mails van de verzender naar de mailserver van de ontvanger. Het maakt gebruik van de volgende drie poorten:

- **TCP Port 25** – de standaard niet-versleutelde SMTP-poort;
- **Port 2525** - de alternatieve niet-versleutelde SMTP-poort;
- **Port 465** – de versleutelde SMTP-poort.



Figuur 2.12

Wanneer de TCP-verbinding succesvol tot stand is gebracht, voeren de twee SMTP-processen een request/response-dialoog uit, die bepaald wordt door het SMTP-protocol.

2.6 OSI- en TCP/IP-modellen

De OSI- en TCP/IP-modellen beschrijven de verschillende taken in lagen (layers) die nodig zijn voor het verbinden en communiceren met netwerken.

Het OSI-model staat voor: Open Systems Interconnection. Het OSI-model is een referentiemodel en beschrijft de functies die op een bepaalde laag uitgevoerd moeten worden.

TCP/IP (Transmission Control Protocol/Internet Protocol) is een suite van protocollen voor het internet en voor intranetten. TCP/IP gebruikt het client/server-model waar een computergebruiker (client) vraagt om een service zoals een webpagina die door een server wordt geserveerd. Programmeurs gebruiken het TCP/IP-model om te zien wat voor protocollen hun programma's gaan draaien. Netwerkbeheerders gebruiken het OSI-model voor *trouble shooting* (netwerkproblemen oplossen). Hieronder worden de OSI- en TCP/IP-modellen schematisch weergegeven.

OSI model layers	OSI- en TCP/IP-modellen		TCP/IP model layers
	Functie	*Voorbeeld*	
(7) Application	Services voor applicaties	HTTP, SMTP, DNS, TelNet	Application
(6) Presentation	Dataformattering Encryptie en decryptie	HTTPS, VPN, SSL	
(5) Session	Opbouwen/verbreken van de verbinding	Class of Service	
(4) Transport	Data-packets samenstellen en beheren	TCP, UDP	Transport
(3) Network	Routeren tussen netwerken	IP, RIP, OSPF	Network
(2)	Verzending gereedmaken van data-packets	Ethernet, P2P	Network access
(1) Physical	Verbinding tussen de apparaten	UTP, Glasvezel, draadloos	

Figuur 2.13 **OSI-model versus TCP/IP-model**

Hier volgt een uitleg over de OSI- en TCP/IP-modellen:

De Application Layer is de interface tussen de gebruiker en de applicaties en gebruikt applicaties zoals HTTP, SMTP en DNS.

De Presentation Layer wordt vooral gebruikt voor het encrypten en decrypten van de data en gebruikt hiervoor HTTPS en SSL (Secure Socket Layer) protocollen voor respectievelijk het encrypten van webpagina's en datatransport.

De Sessions Layer begint, beheert en beëindigt de verbinding tussen client en server. Hij gebruikt hierbij de three-way handshake om de verbinding op te bouwen. Tijdens het uitwisselen van de data-packets controleert de Sessions Layer of beide hosts nog functioneren, als bijvoorbeeld de webserver niet meer reageert zorgt deze voor een foutmelding naar de applicatie (browser).

De Transport Layer deelt de data op in kleinere blokken (data-packets) om ze te kunnen versturen. Deze gebruikt UPD en TCP, die gebruikmaken van de poort-

2.4.1 HTTP-methodes

HTTP implementeert de volgende methodes:

- GET
- POST
- PATCH
- PUT
- DELETE

Lab 2.5 HTTP-headers

Typ het volgende over en sla het op als **lab2.5.php**. Kopieer daarna **lab2.5.php** als **index.php** in de map *Applications/MAMP/Library/htdocs/videobox*.
In het volgende voorbeeld versturen we het volgende formulier naar onze localhost-server met de POST-methode:

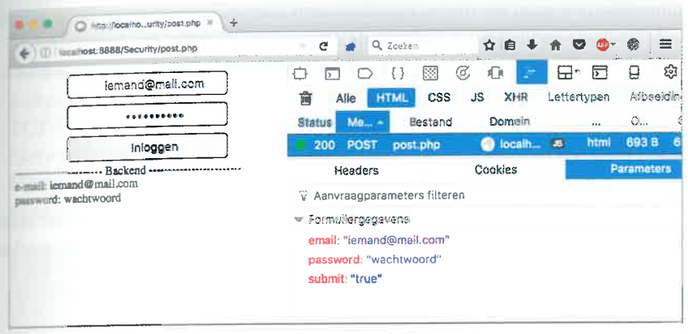
```
<form name="inloggen"  action="" method="POST">
   <input type="text" name="email" value="iemand@mail.com" />
   <input type="password" name="password"value="wachtwoord" />
   <input type="submit" name="submit" value=" Inloggen " />
</form>
<?php
if(isset($_POST['submit'])) {
    echo "------------------ Backend ----------------------";
    $email = htmlspecialchars($_POST['email']);
    $password = htmlspecialchars($_POST['password']);
    echo "<br>e-mail: $email";
    echo "<br>wachtwoord: $password";
}
?>
```

Verstuur de formuliergegevens naar je localhost-server. Om het netwerkverkeer in je browser te zien doe je het volgende:

Typ **CTRL-Alt-Q** om de *Netwerk*-tab van het ontwikkelaarsmenu te openen.

In de volgende figuur zien we de HTTP-header van bovenstaand script. Klik op de *HTML*-tab en daarna op de *Method*-tab.

- Selecteer de POST-methode zoals hieronder.
- Klik op de *Headers*-tab om de HTTP-header te zien.
- Klik op *Parameters* om de data in de HTTP-header te zien.



Figuur 2.11 HTTP-POST-methode

Lab 2.6 Simulatie Man in The Middle

HTTP is niet veilig voor het versturen van gevoelige informatie zoals wachtwoorden of creditcardnummers. In de vorige figuur zien we dat de formuliergegevens te vinden zijn in het parameterdeel van de HTTP-header. Deze gegevens kunnen gezien en gestolen worden door een 'man in het midden'-aanval. Een MiM-aanval wordt uitgevoerd door een onbevoegde gebruiker die naar het netwerkverkeer van anderen luistert. Dit kan gebeuren in open netwerken zoals bij Starbucks of op het vliegveld. We gaan nu een MiM-aanval simuleren.

Stap 1: Verstuur het formulier uit Lab 2.5 opnieuw.

Stap 2: Open de *Netwerk*-tab.

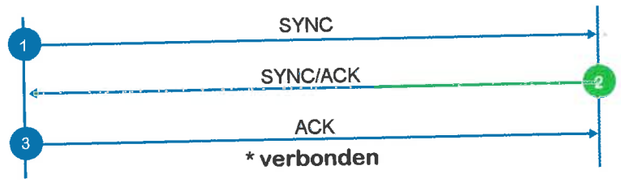
Stap 3: Open de *POST*-tab.

Stap 4: Klik in de *Headers*-tab op de *Edit en verzend*-knop.

Stap 5: Nu ga je de onderschepte parameters editen. Edit bijvoorbeeld de wachtwoord-parameter en klik op verzenden.

In PHP kunnen we header-informatie over de request terugvinden in de `$_SERVER`-array, bijvoorbeeld:

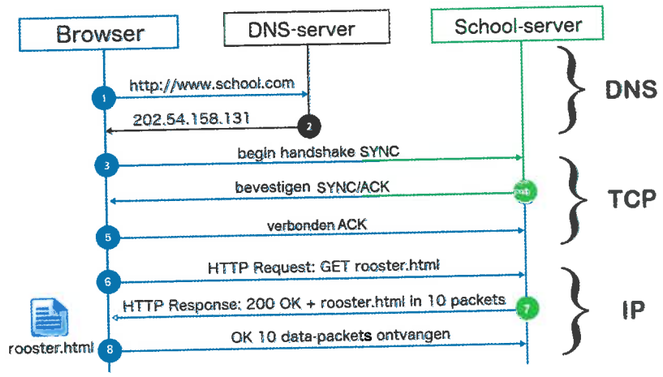
```
Domain name $_SERVER['HTTP_HOST']
Browser $_SERVER['HTTP_USER_AGENT']
Language $_SERVER['HTTP_ACCEPT_LANGUAGE']
Encoding $_SERVER['HTTP_ACCEPT_ENCODING']
IP $_SERVER['REMOTE_ADDR']
```



Figuur 2.8 Transmission Control Protocol handshake

De browser en de server moeten allereerst SYNC- en ACK-packets uitwisselen voordat een verbinding gemaakt kan worden.

Het volgende UML-sequentiediagram schetst hoe HTTP werkt:



Figuur 2.9 Sequentiediagram van HTTP

Hier volgt een uitleg van HTTP.

- Je typt de URL van het rooster in je school-server in, bijvoorbeeld **http://www.school.com**.
- De DNS-server zoekt en geeft je het corresponderende IP-adres van de hostnaam **www.school.com**, bijvoorbeeld **202.54.158.131**.
- TCP maakt een verbinding tussen het IP-adres van je computer en het IP-adres van de school-server door middel van de TCP-handshake.
- HTTP stuurt een GET-aanvraag voor resource **rooster.html**.
- Je school-server zoekt het aangevraagde document **www.school.com/rooster.html**
- Als het document is gevonden, stuurt de school-server een response **200** (dit betekent dat het document was gevonden) en het document **rooster.html** wordt verstuurd.
- Als het document niet gevonden is, stuurt de school-server een response **400** (dit betekent dat het document niet gevonden is).

HTTP-headers

Bij iedere request en iedere response maakt HTTP gebruik van headers. Er zijn vier HTTP-headers:

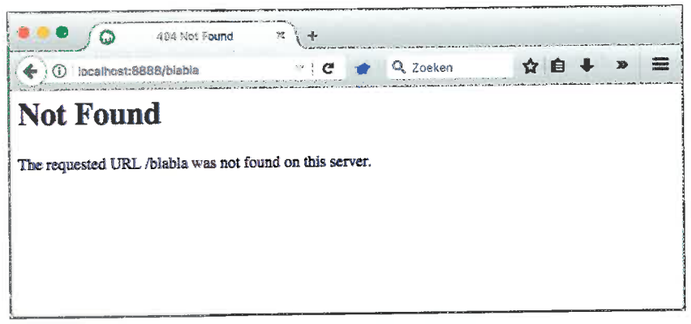
- General: algemene informatie over request en response.
- Request: informatie over het op te halen document.
- Response: informatie over de server en het af te leveren document.
- Entity: bijvoorbeeld content en lengte van het afgeleverde document.

HTTP-statuscode

Hieronder zie je de mogelijke statuscodes van een HTTP-request:

- 200: OK request succesvol
- 201: gecreëerd
- 300: redirection
  - 301: permanent verplaatst
  - 302: tijdelijk verplaatst
  - 304: niet gewijzigd
- 400: probleem met request
  - 401: onbevoegd request
  - 403: verboden, geen toegang
  - 404: document niet gevonden
  - 405: methode niet toegestaan
- 500: probleem met server

In de volgende figuur zien we een HTTP-error 401 als we bijvoorbeeld een verkeerde URL intikken.



Figuur 2.10 HTTP-error 401

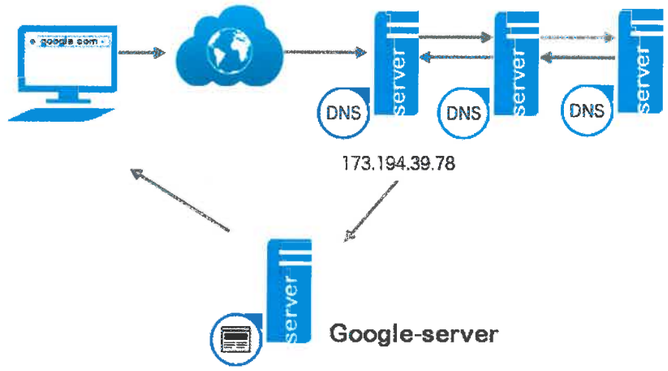
2.3 Domain Name System (DNS)

DNS heeft de primaire functie om domeinnamen te vertalen naar corresponderende publieke IP-adressen. Het is een essentiële component van de functionaliteit van het internet. De DNS-servers hebben een database zoals de volgende tabel:

Domeinnaam	Publiek IP-adres
www.kpn.nl	145.7.170.135
www.tmobile.nl	80.79.204.8
www.google.com	173.194.39.78

Tabel 8 DNS-database

Er zijn DNS-servers voor de domeinen .com, .net, .edu, .org enzovoort. Op dit moment zijn er 53.895 DNS-servers in 205 landen. In de volgende figuur zien we hoe DSN-servers de domeinnaam **google.com** vertalen naar het juiste IP-adres.



Figuur 2.6 DNS-servers

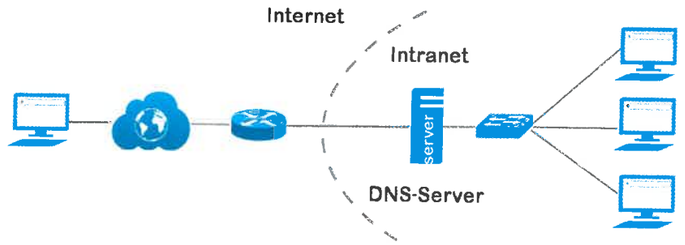
Wanneer je **google.com** typt in je browser maak je verbinding met een van de DNS-servers. De server vertaalt **google.com** naar **173.194.39.78** en maakt een verbinding met **google.com**. Je mag ook 173.194.39.78 zelf intypen, maar **google.com** is makkelijker te onthouden.

DNS spoofing

DNS spoofing is computer hacking met gebruikmaking van de DNS-resolvers-cache van de DNS-server. De hacker breekt in een DNS-server in en corrumpeert de DNS-resolvers-cache met een incorrect IP-adres van de hacker. Het netwerkverkeer wordt dan omgeleid naar de computer van de hacker.

Intranet Domain Name System (DNS)

Een netwerk, bijvoorbeeld een intranet, moet een eigen DNS-register hebben om de domeinnamen (webapplicaties) te kunnen vertalen naar IP-adressen. In de volgende figuur zien we een intranetserver met een DNS-server.



Figuur 2.7 Intranet DNS-server

2.4 HTTP

Het HyperText Transfer Protocol, afgekort HTTP, is een asymmetrisch request-response client-server protocol voor het transporteren van documenten, bijvoorbeeld webpagina's en video's binnen het internet. HTTP maakt gebruik van IP en TCP voor het transporteren van documenten. We kunnen ook zeggen: HTTP (over TCP/IP). TCP en IP hebben we al in paragraaf 1.3 en 1.4 besproken. Hier is weer de tabel met de protocollen:

Applicatie	Protocol	Transport
e-mail	SMTP	TCP
Web	HTTP/HTTPS	TCP
File Transfer	FTP	TCP
Domain Name Server	DNS	UPD

Tabel 9 Protocollen

Uniform Resource Locator (URL)

HTTP maakt gebruik van een URL (Uniform Resource Locator). URL is het adresseringsschema voor documenten in het internet en heeft het volgende formaat:

protocol://hostnaam/documentnaam

Bijvoorbeeld, wanneer je je schoolrooster aanvraagt typ je:

http://www.school.com/rooster.html

HTTP begint met een driestaps handdruk (TCP handshake) die als volgt een verbinding maakt tussen de browser en de server:

2.2.1 Dynamic Host Control Protocol (DHCP)

Routers zijn uitgerust met een DHCP-server. DHCP in de router zorgt voor het uitdelen van statische en dynamische IP-adressen aan de computers in je netwerk. Een **dynamisch IP-adres** is een adres dat tijdelijk is geleased aan een computer in je netwerk. De duur van de lease kan uren of dagen zijn. De DHCP-server kent de reeks van mogelijke dynamische IP-adressen in je netwerk. Een **statisch adres** is bijvoorbeeld het adres van je router (gateway). Alle routers hebben een statisch adres.

Dynamische IP-adressen

De dynamische adressen worden geleased binnen een netwerk aan apparatuur zoals computers, servers en printers zodat ze met elkaar kunnen communiceren. Voor de dynamische IP-adressen per netwerk of subnetwerk gebruiken we drie zogenaamde privé-adresreeksen:

Dynamische privé-IP-adressen		
Reeks	**Van**	**Tot**
1	10.0.0.0	10.255.255.255
2	172.16.0.0	172.31.255.255
3	192.168.0.0	192.168.255.255

2.2.2 Firewalls

Een firewall maakt onderdeel uit van je router. Firewalls opereren tussen privénetwerken en het internet. Firewalls kunnen door middel van regels geconfigureerd worden om communicatie met andere netwerken te blokkeren of toe te staan, bijvoorbeeld tweerichtingcommunicatie via poort 80 toestaan. Een andere regel kan bijvoorbeeld een SQL-server vanaf een extern IP-adres met een specifiek intern IP-adres via poort 1433 laten communiceren. Nog een andere regel kan bijvoorbeeld communicatie blokkeren via risicovolle poorten.

Voorbeelden van firewalls zijn:

- CheckPoint
- Cisco PIX
- SonicWall
- Contivity
- Linksys (thuisnetwerk)

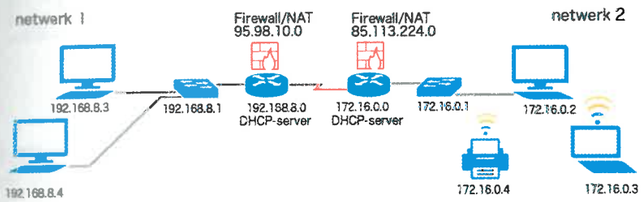
Firewalls voeren ook Network Address Translation (NAT) uit.

2.2.3 Network Address Translation (NAT)

Iedere computer van een privénetwerk krijgt een dynamisch adres door DHCP toegewezen. Alle apparatuur binnen het netwerk communiceert onderling door

middel van deze dynamische privé-IP-adressen. Maar om te communiceren met het internet heb je een publiek IP-adres nodig. NAT is het proces waar een, honderd of duizend computers van een privénetwerk een uniek publiek IP-adres (het adres van de router) gebruiken om toegang te krijgen tot het internet.

In de volgende figuur zien we dat op het moment dat je laptop in netwerk 2 een verbinding maakte met je router, DHCP van je router het dynamische IP-adres 172.16.0.3 voor je laptop maakte.



Figuur 2.5 DHCP en NAT

Wanneer je laptop 172.16.0.3 in netwerk 2 een request naar netwerk 1 stuurt vindt er een Network Address Translation (NAT) plaats, zodat de request via je router 172.16.0.0 verloopt. Wanneer de response van netwerk 1 terug naar je router komt, weet het NAT-protocol dat de response voor je laptop 172.16.0.3 is bedoeld. De rol van het NAT-protocol is om de response door te sturen naar de juiste aanvrager.

Lab 2.4 Firewalls

Maak de topografie van Lab 2.2 af door er firewalls aan toe te voegen.

2.2.4 Gereserveerde publieke IP-adressen

Internet-providers beschikken over reeds gereserveerde publieke IP-adressen. Providers leasen deze adressen aan hun klanten. In de volgende tabel zien we voorbeelden van gereserveerde adressen van twee providers.

Gereserveerde publieke IP-adressen				
Van	**Tot**	**Aantal**	**Provider**	**Datum**
85.113.224.0	85.113.255.255	8192	KPN	15-12-2005
95.98.0.0	95.99.255.255	131072	T-Mobile	12-01-2009

Tabel 7 Gereserveerde publieke IP-adressen

Opgave 5
Schrijf de volgende IPv6-adressen uit:

`3ffe:9:1:1000:0:cafe:ef0:1`

`ffe:3e:1a:1abc:face:0:ef0:1`

IPv6 regel 5
Aaneengesloten hextetten met nullen schrijven we met twee dubbelepunten. Bijvoorbeeld bij het volgende IP-adres:

`ff02:0000:0000:0000:0000:06a0:0500:0001`

kunnen we het volgende doen:

`ff02:0000:0000:0000:0000:06a0:0500:0001`

en als volgt schrijven:

`ff02::6a0:500:1`

Opgave 6
Optimaliseer het volgende IPv6-adres:

`3ffe:0404:0000:0000:0000:0000:0ef0:bc00`

2.1.8 Netwerk- en host-ID's
In IPv4 kunnen we de netwerk-ID door het subnetmasker of het aantal bits identificeren, bijvoorbeeld:

`255.255.255.0 of /24`

In IPv6 identificeren we de netwerk-ID alleen door het aantal bits, bijvoorbeeld:

`3ffe:1944:100:a::1/64`

Het adres kunnen we als volgt uitschrijven:

`3ffe:1944:0100:000a:0000:0000:0000:0001`

In dit geval bestaat de netwerk-ID uit de eerste 64 bits en de host-ID uit de tweede 64 bits:

`3ffe:1944:0100:000a` is de netwerk-ID
`0000:0000:0000:0001` is de host-ID

Hier is een tweede voorbeeld:

`2001::1/96`

In dit geval is de netwerk-ID 96 bits lang en de host-ID 32 bits lang:

`2001:0:0:0:0:0` is de netwerk-ID
`0:1` is de host-ID

Lab 2.3 IPv6
Vind de netwerk-ID en de host-ID van de gegeven IPv6-adressen.

`IPv6 2001::1/80`

In dit geval bestaat de netwerk-ID uit ____ bits en de host-ID uit ____ bits.
Netwerk-ID is:
Host-ID is:

`IPv6 2001::1/16`

In dit geval bestaat de netwerk-ID uit ____ bits en de host-ID uit ____ bits.
Netwerk-ID is:
Host-ID is:

2.2 Routercomponenten

De router van een netwerk voert verschillende functies uit. In de volgende onderdelen bekijken we de volgende routercomponenten:

- Dynamic Host Control Protocol (DHCP)
- Firewalls
- Network Address Translation (NAT)

Opgave 2

Optimaliseer de volgende IPv6-adressen:

```
2001:0ef0:0db8:3ffe:0000:0000:00fe:0001
```

```
3ff2:00ff:0d10:4323:0001:0010:0afe:0001
```

IPv6 regel 2

De tweede regel zegt:

'Een hextet bestaande uit nullen hoef je niet op te schrijven.'

Je schrijft **::** in plaats van de nullen. Bijvoorbeeld bij het volgende adres:

```
2001:db8:aaaa:1111:0000:1100:ef0:0100
```

kun je het volgende doen:

```
2001:db8:aaaa:1111:0000:1100:ef0:100
```

De optimalisering ziet er als volgt uit:

```
2001:db8:aaaa:1111::1100:ef0:100
```

Opgave 3

Optimaliseer de volgende IPv6-adressen:

```
3ffe:0404:001:1000:0010:0000:0ef0:bc00
```

```
3ffe:04dd:1111:0000:00ca:0100:0ef0:0001
```

IPv6 regel 3

We kunnen maar één keer **::** gebruiken per IP-adres. Het volgende IP-adres:

```
2001:0d02:00a0:0000:eda1:00ef:0000:0095
```


kunnen we als volgt schrijven:

```
2001:0d02:00a0:0000:eda1:00ef:0000:0095
2001:d02:a0::       eda1:ef:0:95
```

Of als

```
2001:0d02:00a0:0000:eda1:00ef:0000:0095
2001:d02:a0:0:eda1:ef::         95
```

Opgave 4

Optimaliseer de volgende IPv6-adressen met twee dubbelepunten.

```
3ffe:1111:0000:0000:1111:cafe:ef0:1
```

```
ffe:0000:0000:1111:0000:0000:ef0:1
```

Schrijf de volgende IPv6-adressen uit:

```
ffe:2001:0:101::10:ef:1
```

```
ffe:2001:0:1a::afe:ef0:1
```

IPv6 regel 4

Een geoptimaliseerd IPv6-adres kunnen we uitschrijven. Bijvoorbeeld het volgende adres:

```
2001:0:10:100:1000:cd:ad:101
```

kunnen we als volgt uitschrijven:

```
2001:0000:0010:0100:1000:00ca:00ad:0101
```

Decimaal	Hexadecimaal	Binair
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Tabel 6 Decimale, hexadecimale en binaire getallen

Een hexadecimaal getal bestaat uit 4 binaire bits. Een hexadecimaal is niet hoofdlettergevoelig. Dus `3FFE` is hetzelfde als `3ffe`.

Hexadecimaal	3	f	f	e
Binair	0011	1111	1111	1110

Een hextet bestaat uit vier hexadecimalen of 16 bits, bijvoorbeeld:

hextet			
ødb8			
0000	1101	1011	1000

Opgave 1

Schrijf het volgende hextet in het binaire stelsel:

3ffe			

Schrijf het volgende hextet in het binaire stelsel:

ød2			

Een IPv6-adres is een 128-bits adres en bestaat uit 8 hextetten gescheiden door dubbelepunten (:). Bijvoorbeeld:

```
2001:0db8:aaaa:1111:0000:0000:0000:0100
```

Hieronder zien we een volledig IPv6-adres in 8 hextetten.

1				2				3				4			
2001:				ødb8:				aaaa:				1111:			
0010	0000	0000	0001	0000	1101	1011	1000	1010	1010	1010	1010	0001	0001	0001	0001

5				6				7				8			
0000:				0000:				0000:				0100			
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0001	0000	0000

IPv6 regel 1

Er zijn regels bij het opschrijven van IPv6-adressen die ze vereenvoudigen. De eerste regel zegt:

'Voorloopnullen hoeven we niet op te schrijven.'

Bijvoorbeeld bij het volgende adres:

```
3ffe:0404:0001:1000:0000:0000:0ef0:bc00
```

kunnen we de voorloopnullen als volgt weglaten:

```
3ffe:0404:0001:1000:0000:0000:0ef0:bc00
```

Na optimalisering ziet het er als volgt uit:

```
3ffe:404:1:1000:0:0:ef0:bc00
```

Aantal subnetwerken

We kunnen trouwens ook het aantal subnetwerken controleren door de volgende formule te gebruiken:

Subnetwerken = 2^n,
waar *n* het aantal enen is in het laatste octet van het subnetmasker.

In ons geval is dit: subnetwerken = $2^2 = 4$.
Dat zijn inderdaad vier subnetwerken.

In de volgende tabel zien we dat het eerste subnetwerk 64 beschikbare IP-adressen heeft, waar 0 (het eerste) het subnet-adres is en 63 (het laatste) het broadcast-adres.

Reeks-nummer (63)	Subnet-adres (laagste in reeks)	Beschikbare host-adressen van . . . tot	Broadcast-adres (hoogste in reeks)
0...63	192.168.100.0	192.168.100.1...192.168.100.62	192.168.100.63
64...127	192.168.100.64	192.168.100.65...92.168.100.126	192.168.100.127
128...191	192.168.100.128	192.168.100.129...192.168.100.190	192.168.100.191
192...255	192.168.100.192	192.168.100.193...192.168.100.244	192.168.100.255

Tabel 5 Reeksen per subnetwerk

Ons IP-adres 192.168.100.97 valt in de tweede reeks en behoort tot host 97.

- Subnet-adres is : 192.168.100.64
- IP-adres is :192.168.100.97
- Broadcast-adres is : 192.168.100.127

Lab 2.1 Adres berekenen

In deze lab-opdracht bereken je de juiste adressering voor IP-adres 192.168.100.50 /25.

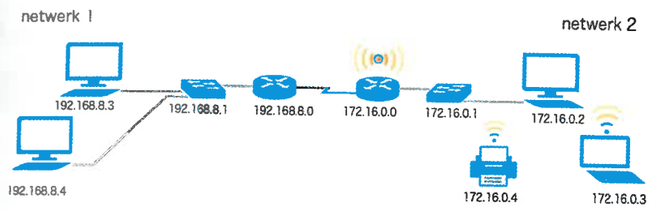
- Subnet-adres is: ?
- Subnetmasker is: ?
- Broadcast-adres is: ?

2.1.5 Standaard-gateway

Onze computer 192.168.100.97 kan rechtstreeks communiceren met alle computers in de reeks 64...127. Maar als onze computer wil communiceren met een computer in een ander netwerk of subnetwerk, bijvoorbeeld 172.16.0.2, hebben we een router of gateway nodig. Onze lokale router is de verbinding met externe routers van andere netwerken. De lokale router wordt de default gateway (uitgang) genoemd en

wordt bij de configuratie van een host (automatisch) opgegeven. De routers zullen de informatie naar het juiste netwerk en de juiste host doorsturen.

In de volgende topologie verbinden we computers binnen een netwerk met switches en we verbinden netwerken met routers.



Figuur 2.4 Routers zijn de gateways

Een router wordt door de netwerk-administrator ingesteld met het juiste subnetmasker en het juiste gateway-adres. Deze verschillen van netwerk tot netwerk.

Lab 2.2 Routers en switches

In deze lab-opdracht teken je de topografie van Lab 1.2 met de Packet Tracer-tool of met Lucidcharts. De routers en switches moeten met de juiste IP-adressen ingesteld worden.

2.1.6 IPv6

IPv6, Internet Protocol versie 6, is het antwoord op het toenemende aantal gebruikers en apparatuur dat gebruikmaakt van het internet. Een IPv6-adres bestaat uit 128 bits en zorgt voor 340 sextiljoen IP-adressen. In deze sectie behandelen we het volgende:

- IPv6-adresnotatie
- Regel 1: voorloopnullen
- Regel 2: twee dubbelepunten ::
- Netwerkprefix

2.1.7 Hexadecimale notatie

We schrijven IPv6-adressen in hexadecimale notatie. Hexadecimaal betekent letterlijk 16-tallig. Er wordt gewerkt met de cijfers 0 t/m 9 en de letters A t/m F.
In de volgende tabel zien we decimale en de corresponderende hexadecimale en binaire getallen:

In ons IP-adres 192.168.10.10 is het eerste octet 192. Dat maakt dit een klasse-C-adres met een CIDR-prefix van 24. Het CIDR-prefix zegt dat de eerste 24 bits van het subnetmasker enen moeten zijn.
In het subnetmasker zet je de eerste 24 bits om in enen en de rest zet je om in nullen. Dit geven we als volgt weer:

	Netwerk-ID			Host-ID
IPv4	192.	168.	10.	10
	11000000	10101000	00001010	00001010
subnetmasker	255.	255.	255.	0
	11111111	11111111	11111111	00000000

Ons subnetmasker is: 255.255.255.0
Zo maskeren we het netwerk-adres af als: 192.168.10.10.
De host-component zegt dat we 1 tot 255 computers kunnen hosten.

Aantal hosts

De beschikbare IP-adressen voor hosts per netwerk rekenen we uit met de volgende formule:

hosts = 255 – de waarde in laatste subnetmasker-octet

In ons geval is dit: 255 – 0 = 255 hosts.

Aantal subnetwerken

We kunnen trouwens ook het aantal subnetwerken controleren door de volgende formule te gebruiken:

Subnetwerken = 2^n,
waar *n* het aantal enen is in het laatste octet van het subnetmasker.

In ons geval is dit: subnetwerken = 2^0 = 1 subnetwerk.

De volgende tabel is een samenvatting van deze sectie:

Netwerkadres	Beschikbare host-adressen van . . . tot	Broadcast-adres (hoogste in reeks)
192.168.10.0	192.168.10.1 . . . 192.168.10.254	192.168.10.255

Tabel 4

- Netwerkadres is: 192.168.10.0
- IP-adres is: 192.168.10.10
- Broadcast-adres is: 192.168.10.255

Het hoogste adres in een subnet is het broadcast-adres. Het bestaat uit het netwerkadres plus allemaal 1'en als host-adres, in het voorbeeld dus 255. Het broadcast-adres wordt door een host gebruikt om naar alle hosts in het subnet tegelijk hetzelfde bericht te sturen. Veel protocollen uit de TCP/IP-protocolsuite maken hiervan gebruik.

2.1.4 Subnetten

Bij subnetten verdelen we het host-deel van het IP-adres in twee delen: een subnet-deel en een host-deel.
Neem bijvoorbeeld het volgende IP-adres:
192.168.100.97/26

De CIDR-prefix /26 zegt dat de eerste 26 bits van het subnetmasker enen moeten zijn.
In het subnetmasker zet je de eerste 26 bits om in enen en de rest zet je om in nullen. Dit geven we in de volgende tabel weer:

	Netwerk-ID			Subnet-ID	Host-ID
IPv4	192.	168.	100.	97	
	11000000	10101000	01100100	01	100001
subnetmasker	255.	255.	255.	192	
	11111111	11111111	11111111	11	000000

IP-adres is: 192.168.100.97
Subnetmasker is: 255.255.255.192

Aantal hosts per subnetwerk

De beschikbare IP-adressen voor hosts per subnetwerk rekenen we uit met de volgende formule:

hosts = 255 – de waarde in laatste subnetmasker-octet

In ons geval is dit: 256 – 192 = 64
Dat zijn 64 mogelijke hosts per subnetwerk.

Dit geeft ons de volgende vier subnetwerken:
Subnetwerk 1 0 tot 63
Subnetwerk 2 64 tot 127
Subnetwerk 3 128 tot 191
Subnetwerk 4 192 tot 255

ieder apparaat een eigen IP-adres krijgt, moet het internet de nieuwe IPv6-infrastructuur implementeren. Een IPv6-adres bestaat uit 128 bits en levert 340 sextiljoen adressen (340 met 36 nullen) op. Verderop behandelen we IPv6-adressen. Voorbeeld van een IPv4-adres is: 192.168.10.10

Een IPv4-adres bestaat uit vier getallen (tussen 0 en 255), gescheiden door punten. Een deel van dit adres identificeert het netwerk en een deel identificeert een unieke host (computer, printer of tablet) in het netwerk. De computer geeft IPv4-adressen weer met vier binaire octetten (bytes). Een octet (byte) bestaat uit acht getallen (bits). Elk bit mag aan of uit zijn (0 = uit, 1 = aan) De waarde van een bit is afhankelijk van de positie in het octet (byte).

In de volgende byte (octet) staan het 8e en 2e bit aan. Dat betekent dat: 8+2= 10.

10							
128	64	32	16	8	4	2	1
0	0	0	0	1	0	1	0
							Bit
Byte							
Octet							

In het volgende byte (octet) staan alle bits aan. Dat betekent dat: 128+64+32+16+8+4+2+1= 255.

255							
128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1

In de volgende tabel zien we een schematische weergave van het IPv4-adres 10.168.10.10, opgesplitst in een netwerkdeel en een hostdeel.

	Netwerk-ID	**Host-ID**		
IPv4	10.	168.	10.	10
	00001010	10101000	00001010	00001010
	stad	huisnummer		

Stel dat het netwerkdeel de stad is en het hostdeel het huisnummer. Dat zijn heel veel huisnummers, veel meer dan we nodig hebben. In de volgende tabel lenen we een octet van het hostdeel en geven dat door aan het netwerkdeel. Dit kun je je voorstellen als een octet voor straten in de stad. Dan krijgen we een stad met straten en huisnummers.

	Netwerk-ID		**Host-ID**	
IPv4	172.	168.	10.	10
	10101100	10101000	00001010	00001010
	stad	straat	huisnummer	

Met twee octetten met 16 bits kunnen we 2^{16} = 65.536 mogelijk hosts hebben. Dit zijn veel hosts (computers) voor een enkel netwerk. In de volgende tabel lenen we nog een octet van het hostdeel. Dit zien we schematisch hieronder weergegeven:

	Netwerk-ID			**Host-ID**
IPv4	192.	168.	10.	10
	11000000	10101000	00001010	00001010

Met een octet met 8 bits hebben we 2^8 = 256 mogelijk hostadressen (computers), waarvan er 254 in ons netwerk te gebruiken zijn.

2.1.3 Subnetmasker

Een computer moet het netwerk- en het host-deel van het IP-adres kunnen scheiden, hiervoor wordt het zogenaamde subnetmasker gebruikt. Om het standaard-subnetmasker van een IP-adres te vinden gebruiken we tabel 3.

Klas	Reeks	Standaard CIDR prefix
A	1-126	/8
B	128-191	/16
C	192-223	/24
D	224-239	multitasking
E	240-255	experimenteel

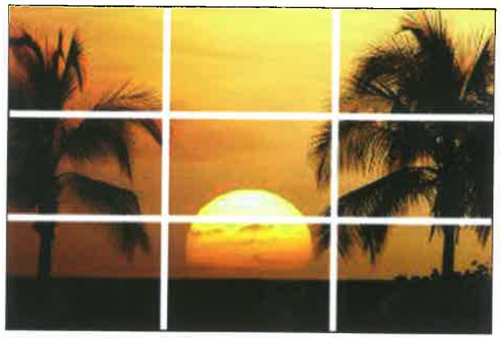
Tabel 3 IP-adresklassen

LET OP

Je ziet dat 127 niet wordt gegeven in de reekskolom. 127 is de loopback. 127.0.0.0 t/m 127.255.255.255 gebruiken we voor de loopback om onze applicaties en routers te testen. We gebruiken bijvoorbeeld 127.0.0.1 als onze localhost-webserver om onze applicaties te testen.

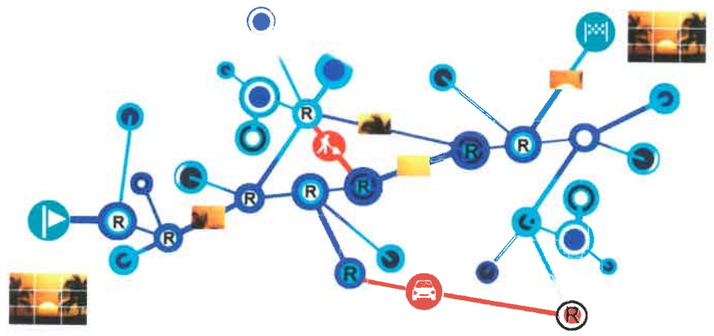
In tabel 3 zien we klassen verdeeld in reeksen bepaald door het eerste octet van elk IP-adres. In de derde kolom zien we de standaard-CIDR (Classless Inter Domain Routing) prefix. We gebruiken de CIDR-prefix om het **subnetmasker** te bepalen. Dit doen we als volgt.

van de data-packets. Stel dat de foto erg groot is, dan wordt de foto door IP in delen verstuurd. Deze delen zijn de data-packets.



Figuur 2.1 Data-packets

De data-packets worden via verschillende routes verstuurd door de routers die deel uitmaken van het netwerk. Ieder packet bevat de IP-adressen van de verzender en de ontvanger. Uiteindelijk arriveren alle packets via verschillende routes op hun bestemming en worden ze weer omgezet in de oorspronkelijke foto. In de volgende figuur wordt het Internet Protocol schematisch weergegeven.



Figuur 2.2 Internet Protocol

In de figuur zien we dat er alternatieve routes zijn gebruikt tussen het vertrekpunt en aankomstpunt die de werkzaamheden en de verkeersproblemen vermijden.

IP controleert ook de snelheid van de data-packets: geluid-packets worden bijvoorbeeld sneller getransporteerd dan video- of tekst-packets.

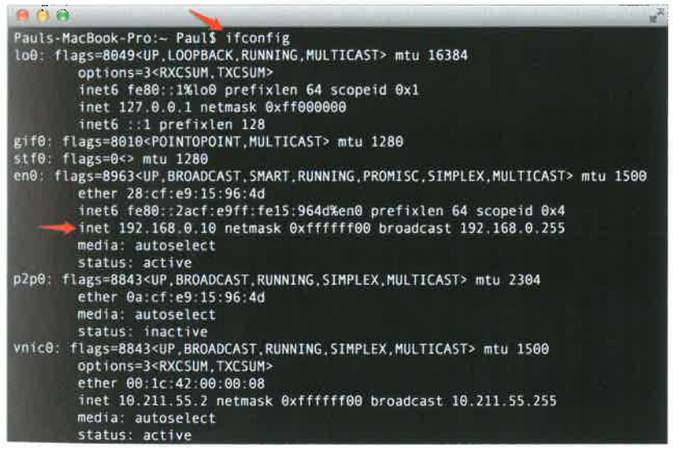
Het Internet Protocol (IP) maakt gebruik van TCP en wordt ook aangeduid als de TCP/IP-suite. IP specificeert het formaat van de data-packets (datagrams) en het adresseringsschema binnen het netwerk. Volgens het adresseringsschema krijgt ieder apparaat dat verbonden is met het internet een uniek adres. Dit adres heet het IP-adres. IP-adressen maken het mogelijk om computers met elkaar te verbinden. In de volgende secties bespreken we:

- IP-adressen
- Subnetmaskers
- Standaard-gateway

Een netwerk is een verzameling hosts (computers) met een eigen domein (adres). Alle hosts binnen een netwerk delen hetzelfde netwerkadres, hetzelfde subnetmasker en dezelfde standaard-gateway. Netwerkadressen, subnetmasker en standaard-gateway bespreken we verderop in dit hoofdstuk.

2.1.1 IP-adressen

Als je naar de opdrachtprompt in je computer gaat en vervolgens **ipconfig** (of **ifconfig** voor MAC en Linux) intypt krijg je je IP-adres te zien zoals hieronder:



Figuur 2.3 IP-configuratie

Hier zien we als IPv4-adres **192.168.0.10**.

2.1.2 IPv4

IPv4 (Internet Protocol versie 4) is de huidige adresseringsinfrastructuur van internet. Een IPv4-adres bestaat uit 32 bits met maximaal iets meer dan vier miljard IP-adressen. Omdat het Internet of Things zich steeds sneller ontwikkelt, waar

Hoofdstuk 2
Internetarchitectuur

Het internet is een architectuur die geregeld wordt door een aantal protocollen. Internetprotocollen, zoals netwerkprotocollen, zijn regels en standaarden die gebruikt worden om het internetverkeer te reguleren. Een protocol controleert ook dat alles volgens de regels wordt uitgevoerd.

Leerdoelen

Aan het einde van dit hoofdstuk moet je de volgende kennis en vaardigheden beheersen.

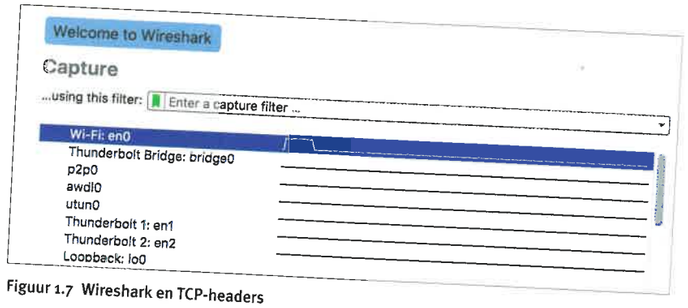
- Internet Protocol (IP)
  - IPv4
  - IPv6
- Routercomponenten
  - Dynamic Host Protocol (DHCP)
  - firewalls
  - Network Address Translation (NAT)
- Domain Name System (DNS)
- Hyper Text Transfer Protocol (HTTP)
- Man-in-the-Middle-simulatie (MiM)

2.1 Internet Protocol (IP)

Het internet werkt zoals het landelijke postsysteem. Je stuurt post met het adres van de verzender en de geadresseerde. In het postkantoor wordt de post gesorteerd en via de juiste route op het juiste adres afgeleverd.

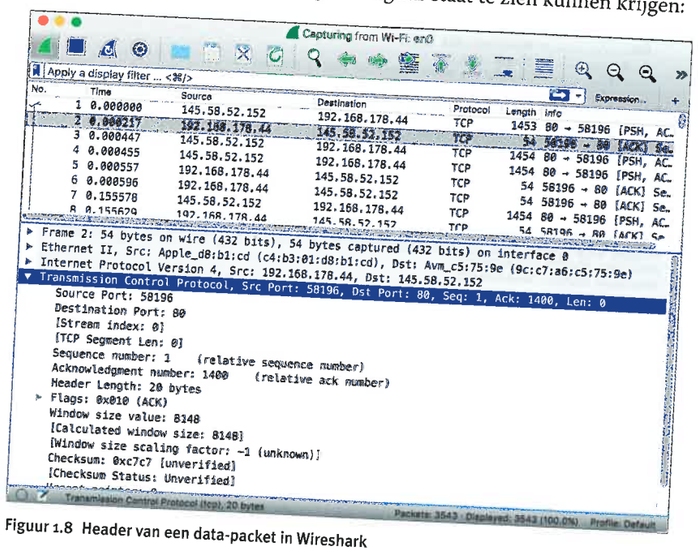
Het internet is het netwerk van netwerken waar wereldwijd netwerken met elkaar communiceren. Internet wordt ook de 'digitale snelweg' genoemd. Denk aan de Nederlandse en Belgische wegennetten en hoe die aansluiten bij de rest van het Europese wegennet en daarna bij Oost-Europese wegennetten. Je hebt een vertrekpunt, je hebt snelheidscontroles en een aankomstpunt. Je hebt een navigatiesysteem dat alternatieve routes uitrekent bij verkeersproblemen. Internet werkt precies op dezelfde manier.

Stel je wilt een vakantiefoto vanaf je smarttelefoon versturen naar de telefoon van je collega. IP controleert de adressering, de verpakking van de foto en de snelheid



Figuur 1.7 Wireshark en TCP-headers

Selecteer *Wi-Fi: enø* en scan het dataverkeer tussen jouw PC en **google.com** bijvoorbeeld. Je zou zoiets als in de volgende figuur staat te zien kunnen krijgen:

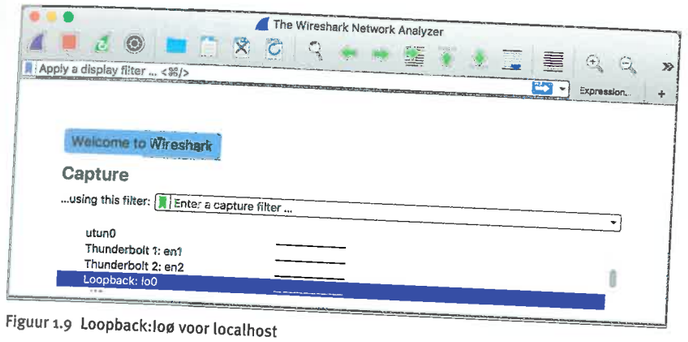


Figuur 1.8 Header van een data-packet in Wireshark

In de volgende link vind je meer documentatie over Wireshark:

https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html

Start een eigen applicatie in localhost. Selecteer de *Loopback*-optie in Wireshark om het dataverkeer tussen je laptop en je localhost te scannen.



Figuur 1.9 Loopback:lø voor localhost

Kijk om je verder in Wireshark te verdiepen naar de volgende YouTube-tutorial over Wireshark:

https://www.youtube.com/watch?v=TkCSr3øUojM

1.3.2 File Transfer Protocol (FTP)

FTP is een protocol voor het overbrengen van computerbestanden tussen een client en een server in een computernetwerk. Het protocol voert de volgende stappen uit:

1. FTP-client maakt contact met de FTP-server via poort 21 en specificeert het TCP transportprotocol.
2. Client krijgt verbinding.
3. Client stuurt bestandsaanvraag naar de file-server.
4. File-server opent een TCP-verbinding met de client en voert de bestandsoverdracht uit.
5. Na de bestandsoverdracht wordt de verbinding beëindigd door de file-server.

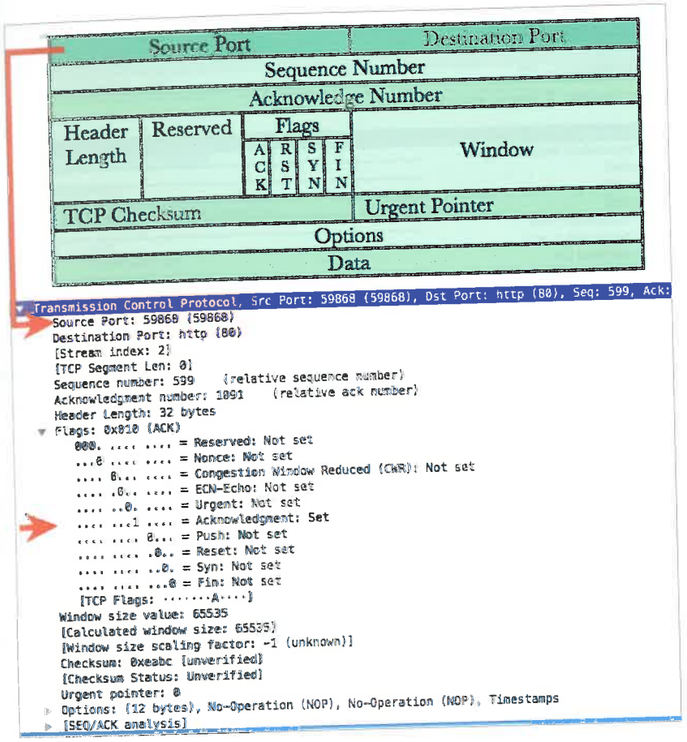
Source port					Destination port
Sequence number					
Acknowledge number					
Header length	Reserved	Flags			Window
		ACK	RST	SYN	FIN	
TCP checksum					Urgent pointer
Options					
Data					

Tabel 2 Structuur van een TCP-header

Hier volgt een uitleg over de structuur van headers:

- Source port: is de poort van de applicatie of het apparaat dat het packet verzendt.
- Destination port: is de poort van de applicatie of het apparaat dat het packet ontvangt.
- Sequence number: is het volgordenummer van het packet.
- Acknowledge number: geeft het nummer van het volgende packet aan.
- Flags
  - ACK: packet acknowledgment, bevestigt of het vorige packet was ontvangen
  - RST: weigert packet/herstart verbinding
  - SYN: verbinding gemaakt
  - FIN: verbinding beëindigd

In de volgende figuur zien we in Wireshark de header van een packet met de ACK-flag aan.



Figuur 1.6 Flags in data-packet

Lab 1.2 Wireshark

In deze lab-opdracht maken we kennis met Wireshark. Wireshark is een tool voor het scannen of 'snuiven' van data-packets in netwerkverkeer en het analyseren van de gebruikte protocollen in netwerkcommunicatie. Ga om Wireshark te downloaden naar de volgende link:

https://www.wireshark.org

of

https://www.wireshark.org/download.html

Installeer en start Wireshark. Dubbelklik op *Wi-Fi: enø* om het dataverkeer tussen je PC en het internet te scannen. Maak een scan. Voor het dataverkeer tussen je PC en je localhost selecteer je *Loopback: IoO*. Maak een scan.

1.3 Netwerkprotocollen

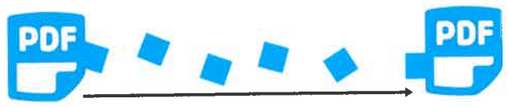
Protocollen zijn regels en standaarden die gebruikt worden om twee of meer computers te verbinden en om informatie en bestanden te delen. Een protocol controleert dat alles volgens de regels wordt uitgevoerd.
In deze paragraaf bespreken we de volgende protocollen:

Applicatie	Protocol	Transport
e-mail	SMTP	TCP
Web	HTTP/HTTPS	TCP
File Transfer	FTP	TCP
Domain Name Server	DNS	UPD

Tabel 1 Protocollen

1.3.1 Transmission Control Protocol (TCP)

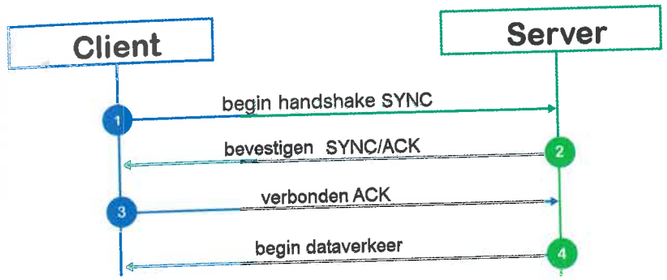
Het Transmission Control Protocol oftewel TCP controleert de netwerkverbindingen in een local area netwerk (LAN). Hier wordt geregeld hoe computers binnen een netwerk communiceren. TCP regelt het verkeer van data-packets. Een data-packet is een data-eenheid. Stel je voor dat je een groot PDF-document wilt sturen van jouw computer naar een andere computer. TCP breekt het document op in kleinere data-packets. Deze packets worden via een netwerkverbinding verstuurd naar de ontvangende computer.



Figuur 1.4 PDF verdeeld in data-packets

TCP handshake

In de volgende figuur gebruiken we het UML-sequentiediagram om de three-way handshake van TCP weer te geven. UML is de modelleringstaal die software-ontwikkelaars het meest gebruiken om complexe systemen en processen in kaart te brengen.



Figuur 1.5 Three-way handshake bij TCP

In dit sequentiediagram maken de client en de server een verbinding (handshake). TCP zorgt ervoor dat de packets in de juiste volgorde op de bestemde computer worden ontvangen.

Poorten

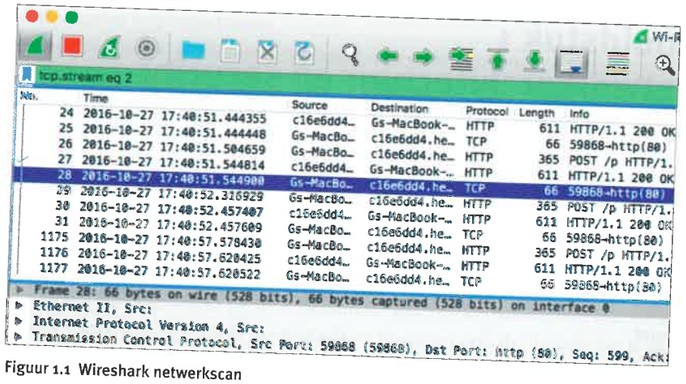
TCP gebruikt een adresseringsschema om computers binnen een LAN te laten communiceren. Dit schema maakt gebruik van poortnummers waarbij de zendende computer een poortnummer krijgt en de ontvangende computer een tweede poortnummer krijgt. De poortnummers lopen van 0 tot 65535 en zijn als volgt verdeeld:

Systeempoorten	0 – 1023
Gebruikerspoorten	1024 – 49151
Privé/Dynamische poorten	49152 – 65535

Het protocol zorgt ervoor dat als een data-packet niet is ontvangen het data-packet opnieuw wordt verzonden.

Header

TCP verstuurt elk data-packet met een eigen data-packet-header. Een header bevat twee soorten informatie. Ten eerste meta-informatie over het desbetreffende data-packet zoals verzendingsgegevens. Ten tweede de content of *payload* van het packet. In de volgende tabel staat een weergave van de structuur van een TCP-header.



Figuur 1.1 Wireshark netwerkscan

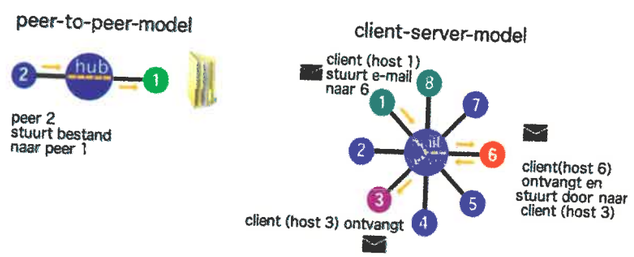
Op het eerste gezicht lijkt het allemaal raadselachtig. In de volgende hoofdstukken gaan we deze geheimzinnige codes ontcijferen. Om deze scan te kunnen analyseren moeten we eerst kennismaken met een aantal basisbegrippen over netwerken zoals:

- netwerkmodellen
- netwerkprotocollen
- data-packets
- headers
- poorten
- IP-adressen

Zoals eerder gezegd, om een gebouw te kunnen beveiligen moet je eerst de blauwdruk van het gebouw bestuderen om de kwetsbare punten te kunnen identificeren. Een netwerkblauwdruk noemen we een netwerkmodel.

1.2.1 Netwerkmodellen

Hieronder zien we twee netwerkmodellen: het **peer-to-peer**-model, verkort tot **P2P,** en het **client/server**-model.



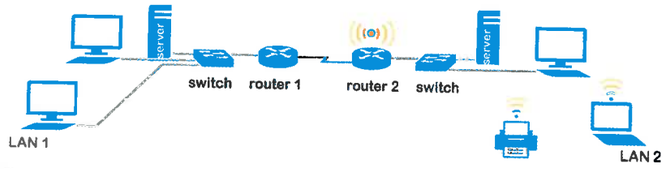
Figuur 1.2 Netwerkmodellen

In een **peer-to-peer**-model hebben we twee computers die bestanden met elkaar delen. Voorbeelden van P2P-netwerken zijn bestanden delen met Bit Torrent of instant messaging met WhatsApp. In dit model zijn beide computers **servers**, want beide computers serveren bestanden.

In een **client/server**-model is één computer de **server** en de andere zijn **clients**. De server serveert bestanden en andere data aan de clients. Voorbeelden van client/server-netwerken zijn webwinkels en internetbankieren.

1.2.2 Routers en switches

Met **switches** kunnen we computers fysiek met elkaar verbinden binnen een local area network (LAN). Een **router** verbindt twee of meer netwerken. In de volgende figuur zien we een netwerktopologie met netwerk 1 met twee computers verbonden via de router met netwerk 2.



Figuur 1.3 Netwerktopologie

Een server kan bestanden delen (serveren) met de computers in het eigen netwerk. Maar deze kan ook bestanden serveren via de router naar computers in een ander netwerk. Dit doet de router met behulp van protocollen.

Lab 1.1 Netwerkdiagram

In deze lab-opdracht maak je kennis met een van de volgende twee softwareprogramma's:

Cisco Packet Tracer: https://www.netacad.com/campaign/ptdt-1/

of

Lucidcharts: https://www.lucidchart.com

Stap 1: Teken de topologie in Packet Tracer of Lucidchart zoals te zien is in de netwerkconfiguratie in figuur 1.3.

Stap 2: Voeg een nieuw LAN met drie pc's eraan toe.