FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

# Beveiliging van netwerken en computers

Bert De Saffel

Master in de Industriële Wetenschappen: Informatica Academiejaar 2018–2019

Gecompileerd op 21 november 2018

UNIVERSITEIT
GENT

# Inhoudsopgave

# Hoofdstuk 1

# Basisconcepten

Computerbeveiliging

# Hoofdstuk 2

# Netwerk en communicatiebeveiliging

Voorlopig hebben we enkel de basisconcepten gezien zoals: symmetrische encryptie, asymmetrische encryptie, hashfuncties en message authentication codes. Dit hoofdstuk zal deze concepten toepassen om beveiligingsprotocollen te ontwikkelen voor netwerken.

## 2.1 SSH

SSH (Secure Shell) is een **applicatielaagprotocol**. Ondanks deze naamgeving bevat SSH ook een **transportlaagprotocol**, met als bedoeling een veilige connectie te maken tussen andere OSI applicatielaagprotocollen (bv HTTP) en de werkelijke OSI transportlaag. SSH can draaien op zowel workstations als routers en switches. Routers en switches volgen de OSI laag namelijk niet strict, zodat zij ook een applicatielaag hebben. Vroeger werd het programma **telnet** gebruikt om remote configuratie toe te passen. Deze manier van werken is **onveilig** aangezien verkeer tussen de local host en remote host niet geëncrypteerd wordt, zodat deze door eender wie kunnen bekeken worden. SSH verhelpt dit probleem door de informatie te encrypteren. Vrijwel alle UNIX en Linux distributies komen met een versie van SSH geinstalleerd. SSH laat onder andere toe om:

- een veilige remote verbinding op te zetten vanuit eender welke local host naar eender welke remote host. Het protocol maakt gebruik van erkende algoritme voor zowel encryptie (sleutels van ten minste 128 bits), data-integriteit, sleuteluitwisselingen en public key management. Tijdens het leggen van een connectie worden algoritmen tussen de local host en remote host afgesproken, op basis van welke ze al dan niet ondersteunen,
- TCP te tunnelen via een SSH connectie,
- bestanden te verplaatsen, gebruik makend van bijhorende protocols zoals SCP (Secure Copy) of SFTP (SSH File Transfer Protocol),
- zowel X-sessions als poorten te forwarden.

### 2.1.1 Architectuur

SSH kan opgesplitst worden in drie elementen: Transport Layer Protocol, User Authentication Protocol en het Connection Protocol.

#### Transport Layer Protocol

Dit protocol heeft onder andere de verantwoordelijkheid om: authenticeren van servers, sleuteluitwisselingen en perfect forward privacy implementeren. Perfect forward privacy wil zeggen dat, indien een sleutel gecomprimeerd wordt tijdens een sessie, deze geen invloed kan hebben op de beveiliging van vorige sessies. Dit protocol loopt uiteraard op de transportlaag, en in de meeste gevallen zal dit TCP zijn. Vooraleer de cliënt kan connecteren moet hij de **public host key** van de server bezitten. Hiervoor kunnen er twee modellen gebruikt worden (cfr. RFC 4521) waarbij:

1. de cliënt een lokale databank bevat die de mapping beschrijft van elke hostnaam naar de correspondeerde public host key. Op deze manier is er geen centrale entiteit nodig, maar het beheer van deze databank kan, indien deze groot genoeg wordt, lastig zijn om te onderhouden.
2. de mapping bevestigd wordt door een **CA (Certification Authority)**. De cliënt kent in dit geval enkel de CA root key waarmee de geldigheid van elke host key kan nagaan die getekend zijn door deze CA. In dit geval moet de cliënt slechts één of enkele CA root keys bevatten.

Wanneer er een connectie kan gelegd worden van een cliënt tussen een server, is het eerste proces altijd het **onderhandelen van de algoritmen**. Deze stap zal algoritmen selecteren die compatibel zijn met zowel de cliënt als de server. Wanneer de cliënt een TCP connectie heeft met de server worden volgende pakketten verstuurd:

- **Identification string exchange.** Dit is een string dat zowel het protocolversie als de softwareversie van SSH bevat. Deze string wordt ten eerste van de cliënt naar de server verstuurd, waarop de server dan antwoord met zijn protocol en softwareversie. Indien hieruit blijkt dat de machines niet compatibel zijn met elkaar, wordt de connectie onderbroken (SSH2 is bv niet compatibel met SSH) en worden volgend stappen bijgevolg niet meer uitgevoerd.
- **Algorithm Negotiation.** In deze fase sturen zowel de server als de cliënt een `SSH_MSG_KEXINIT` bericht, die een lijst van alle ondersteunde algoritmen bevat, in volgorde van voorkeur. Elk type van algoritme zoals sleuteluitwisseling, encryptie, MAC algoritme en compressiealgoritme heeft zo zijn eigen lijst. Elk type moet dan ook een algoritme toegekend krijgen en wordt bepaald door het eerste algoritme dat de cliënt goed vind dat ook beschikbaar is op de server.
- **Key Exchange.** Indien de vorige fase goed gelukt is, start nu de sleuteluitwisseling. Voor de sleuteluitwisseling worden er momenteel slechts twee versie van **Diffie-Helman** ondersteund (cfr. RFC 2409). Het einde van de sleuteluitwisseling wordt gesignaleerd door een `SSH_MSG_NEWKEYS` pakket, met als gevolg dat zowel de cliënt als de server de gegenereerde sleutels mag gebruiken.
- **Service Request.** De laatste stap, dat eigenlijk het begin is van een volgend proces, wordt gesignaleerd door een `SSH_MSH_SERVICE_REQUEST` pakket. Dit pakket vraagt ofwel de start van

het User Authentication Protocol of van het Connection Protocol. Alle verkeer tussen server en cliënt wordt op dit moment getransporteerd als de payload van een SSH Transport Layer pakket, beveiligd met encryptie en een MAC.

Een SSH Transport Layer pakket heeft de volgende vorm, waarbij elementen met een $\delta$-symbool geëncrypteerd en geauthenticeerd zijn en elementen met een $\Delta$-symbool ook optioneel gecompresseerd kunnen zijn:

$\delta$ **Pakketlengte (4 bytes).** De lengte van het pakket in bytes, zonder de lengte van de pakketlengte zelf en het MAC veld in beschouwing te nemen.

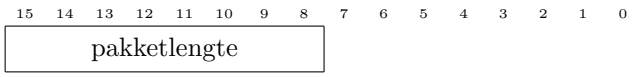
$\delta$ **Paddinglengte (1 byte).** Dit is de lengte van het random padding veld.

$\Delta$ **Payload**. De eigenlijke informatie van het pakket.

$\delta$ **Random padding.** Dit veld dient om cryptoanalyse moeilijk te maken. Kleine pakketten zijn op deze manier minder veel minder voorspelbaar en algoritmen, die vaak een kenmerkende vaste lengte hebben, worden ook moeilijker te achterhalen.

/ **MAC veld.** Dit veld wordt enkel gegenereerd indien dit zo in de onderhandeling besproken werd. Dit veld wordt berekent over het hele pakket en krijgt ook nog een bijhorend sequentienummer. Het sequentienummer start op 0, en wordt telkens met 1 geïncrementeerd voor elk pakket. Een aanvaller kan dit sequentienummer niet achterhalen aangezien een MAC een onomkeerbaar proces is.

packet




ToDo: juiste afbeelding

Het **sleuteluitwisselingsproces** vraagt wat enige uitleg, er is echter nog niet nagegaan of deze sleutels op een **veilige** manier uitgewisseld worden. Op het moment van sleuteluitwisseling is er helemaal nog geen SSH connectie. Het uitwisselingsproces verloopt in twee fasen: eerst wordt er een gedeelde sleutel gegenereerd met Diffie-Helman, daarna wordt deze gedeelde sleutel gesigneerd met de publieke sleutel van de cliënt voor authenticatie. Sleutels kunnen ook heruitwisseld worden, hierbij gelden volgende regels:

- Mogelijk op elk moment, behalve tijdens het sleuteluitwisselingsproces.
- Kan aangevraagd worden door beide partijen.
- Sessie identificaties blijven ongewijzigd.
- Cryptografische algoritmen kunnen gewijzigd worden.
- Sessiesleutels worden vervangen.

Meestal worden sleutels vervangen na het behalen van een bepaalde quota zoals een tijdslimiet of het aantal totaal verstuurde bytes.

### User Authentication Protocol

Dit protocol specifieerd **hoe** een cliënt zich moet authenticeren aan een server. Meerdere methoden zijn mogelijk waaronder de drie belangrijkste ervan: Public Key Authentication, Password Authentication en Host Based Authentication:

- **Public Key Authentication.** De implementatie van deze methode is afhankelijk van het gebruikte public-key algoritme. Een cliënt verstuurt berichten, dat gesigneerd is door de cliënt zijn private sleutel, naar de server, die de publieke sleutel van de cliënt bevat. De server gaat na of deze publieke sleutel nog geldig is en zoja, of dat de signatuur correct is.
- **Password Based Authentication.** De cliënt verstuurt zijn paswoord, dat geëncrypteerd wordt door het SSH Transport Layer Protocol naar de server, die nagaat of het paswoord correct is.
- **Host Based Authentication.** De authenticatie is gebaseerd op de host in plaats van een individuele gebruiker (cliënt) zelf. De cliënt verstuurt een handtekening, gemaakt met de private key van de host. De server verifieerd enkel de host, en dus niet de individuele gebruikers.

### Connection Protocol

Het connection protocol veronderstelt dat een beveiligide, geauthenticeerde verbinding tot stand gebracht is. Deze verbinding, **tunnel** genoemd, wordt gebruikt door het connection protocol om een aantal logische kanalen te multiplexen doorheen deze tunnel. Het connection protocol specifieert vier kanalen:

1. **Session.** Dit kanaal refereert naar het remote uitvoeren van een programma zoals een shell, een applicatie zoals e-mail of een systeemcommando.
2. **X11.** Het X Window System laat toe om applicaties te runnen op een server, maar dat ze eigenlijk op een desktop getoond worden.
3. **Direct-tcpip.** Dit kanaal dient voor local port forwarding.
4. **forwarded-tcpip.** Dit kanaal dient voor remote port forwarding.

De lifecycle van een kanaal verloopt in drie stages: een kanaal openen, uitwisselen van informatie en een kanaal sluiten. Wanneer de server of de cliënt (hier uitgewerkt voor de cliënt) een kanaal wil openen, wordt er een lokaal nummer bijgehouden voor dat kanaal en wordt er een bericht verstuurt naar de server met volgende informatie:

- Eén van de vier kanaaltypes.
- Het lokale kanaalnummer van de cliënt.
- De initiële window size. Dit is het aantal bytes dat kan verstuurt worden naar de verzender zonder dat het venster moet aangepast worden.
- De maximum pakketgrootte specifieert hoeveel bytes een individueel pakket mag bevatten.

Als de server het kanaal kan openen, zal het een `SSH_MSG_CHANNEL_OPEN_CONFIRMATION` bericht versturen. Dit bericht bevat het kanaalnummer van de cliënt, het kanaalnummer van de server, en de waarde voor de window-en pakketgrootten voor inkomend verkeer. Als de server het kanaal niet kan openen, zal het een `SSH_MSG_CHANNEL_OPEN_FAILURE` versturen, met een foutmelding. Bij een open kanaal worden er voortdurend `SSG_MSG_CHANNEL_DATA` berichten verstuurd, die het kanaalnummer van de ontvanger en een datablok bevat. Deze berichten kunnen in twee richtingen voorkomen. Tot slot kan er nog een `SSH_MSG_CHANNEL_CLOSE` bericht verstuurd worden, die het kanaalnumer bevat dat gesloten moet worden.

### 2.1.2 Port forwarding

Port forwarding of port mapping is een toepassing van **NAT (Network Address Translation)** dat een een communicatieaanvraag van een specifiek adres en poort naar een ander zal omleiden. Op deze manier kan elke onveilige TCP verbindingen in een veilige SSH verbinding gemaakt worden. Een klein woordje uitleg over een poort. Een poort is een identificatie van een TCP-gebruiker. Elke applicatie dat bovenop TCP draait, heeft een poortnummer. Het **SMTP (Simple Mail Transfer Protocol)** luistert algemeen naar poort 25. Inkomende SMTP berichten zullen dus deze poortnummer bevatten. TCP Herkent dit poortnummer en zal het verkeer routen naar de SMTP server applicatie. SSH ondersteunt twee types port forwarding: local forwarding en remote forwarding:

- Local forwarding.
- Remote forwarding.

## 2.2 Sleuteluitwisselingen

Gegeven twee partijen $A$ en $B$. Een sleuteluitwisseling kan op verschillende manieren gebeuren:

1. $A$ kan het fysiek afleveren aan $B$.
2. Een betrouwbare partij geeft de sleutel fysiek aan $A$ en $B$.
3. Als $A$ en $B$ vroeger al gecommuniceerd hebben, kunnen ze de vorige sleutel gebruiken om de nieuwe te encrypteren.
4. Als $A$ en $B$ een beveiligide verbinding hebben met een derde partij $C$, dan kan $C$ de sleutel doorverwijzen naar $A$ en $B$.

### 2.2.1 Out-of-band sleuteluitwisseling

Sleutels niet-digitaal uitwisselen wordt out-of-band sleuteluitwisseling genoemd. Dit is toelaatbaar voor kleine organisaties maar voor grotere organisaties is dit niet voldoende.

### 2.2.2 Diffie-Hellman

Dit protocol was de eerste praktische methode om een gedeelde sleutel over een onbeveiligide verbinding te versturen, gebaseerd op discrete logaritmen. Het maakt gebruik van de eigenschap

$$(g^b \ mod \ p)^a \ mod \ p = (g^a \ mod \ p)^b \ mod \ p$$

indien $p$ een priemgetal en $g$ de primitieve wortel modula $p$ is. Een voorbeeld is $p = 7$ en $g = 3$.

$$3^1 = 3 = 3 \bmod 7$$
$$3^2 = 9 = 2 \bmod 7$$
$$3^3 = 27 = 6 \bmod 7$$
$$3^4 = 81 = 4 \bmod 7$$
$$3^5 = 243 = 5 \bmod 7$$
$$3^6 = 729 = 1 \bmod 7$$

Het algoritme verloopt in volgende stappen:

1. Alice en Bob zoeken een priemgetal $p$, en een daarbijhorende primitieve wortel $g$. Deze $p$ en $g$ mogen publiek gekend zijn.
2. Alice kiest een geheim getal $a < p$. Haar publieke sleutel is $y_A = g^a \bmod p$.
3. Bob kiest een geheim getal $b < p$. Zijn publieke sleutel is $y_B = g^b \bmod p$.
4. Alice en Bob bereken respectievelijk $y_B^a \bmod p$ en $y_A^b \bmod p$. Deze zullen beiden dezelfde uitkomst bekomen, en wordt dan hun sleutel.

Aangezien $g^x$ elk mogelijk getal zal genereren modulo $p$ voor alle $x < p$, is het moeilijk voor een aanvaller, die de publieke sleutels toch kent, de geheime sleutel te achterhalen. Ze moeten hierbij het discrete logaritme probleem oplossen.

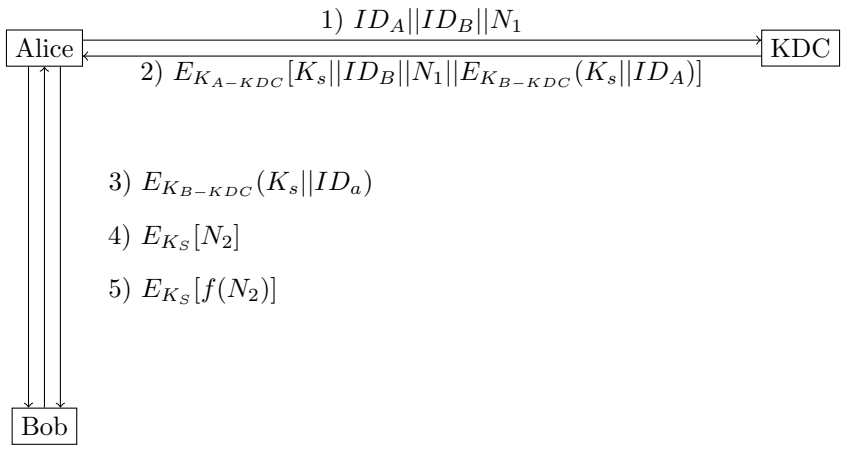
Stel $p = 23$ en $g = 5$. Alice neemt $a = 6$ en Bob neemt $b = 15$, dan wordt $y_A = 5^6 \bmod 23 = 8$ en $y_B = 5^{15} \bmod 23 = 19$. Alice berekent $19^6 \bmod 23 = 2$ en Bob berekent $8^{15} \bmod 23 = 2$.

Normaal wordt $p$ een priemgetal van ongeveer 300 cijfers genomen. De getallen $a$ en $b$ zijn dan weer minstens 100 cijfers lang. De geheime sleutel bepalen, indien $g$, $p$, $y_A$ en $y_B$ gekend zijn duurt quasi oneindig lang.

#### Nadelen Diffie-Hellman

### 2.2.3 Sleuteldistributiecentrum

Het sleuteldistributiecentrum vermijdt dat gebruikers zelf de sleutel moeten overhandigen. Elke gebruiker heeft een geheime sleutel die kan gebruikt worden om met het **KDC (Key Distribution Centre)** te communiceren. Er zijn twee soorten sleutels: de sessiesleutels zijn een tijdelijke sleutel die gebruikt worden voor één enkele sessie. Deze sessiesleutel wordt gebruikt om data te encrypteren



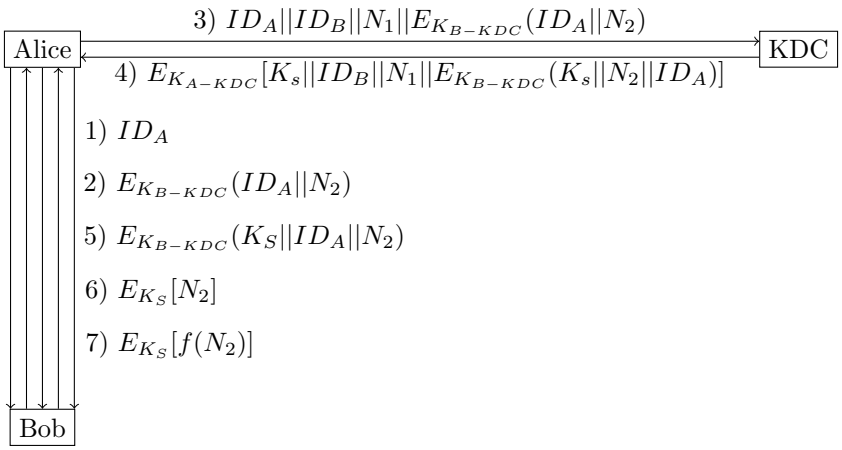

Figuur 2.1: Basiswerking KDC.

tussen verschillende gebruikers. De ander sleutel is de mastersleutel. Deze wordt gebruikt om sessiesleutels te encrypteren en wordt gedeeld tussen een gebruiker en de KDC. De basiswerking van de KDC (figuur 2.1) is als volgt:

1. Alice vraagt aan de KDC om een verbinding op te zetten met Bob. Alice stuurt ook een "nonce" $N_1$. Dit is een willekeurig getal dat slechts éénmaal gebruikt wordt om replay aanvallen te voorkomen.
2. De KDC geeft volgende informatie terug, dat geëncrypteerd wordt door Alice haar sleutel $K_{A-KDC}$: De sessiesleutel, de identiteit van Bob, dezelfde nonce dat Alice stuurde naar de KDC, en geëncrypteerde data (met Bob zijn sleutel $K_{B-KDC}$).
3. Alice verstuurt de geëncrypteerde data naar Bob.
4. Bob antwoordt met een tweede nonce $N_2$, geëncrypteerd met de sessiesleutel, zodat Bob geauthenticeerd is naar Alice toe.
5. Alice antwoordt terug naar Bob met een bewerkte nonce (een bepaalde functie, bv 1 toevoegen), geëncrypteerd met de sessiesleutel, zodat Alice geauthenticeerd is naar Bob toe.

Een aanvaller kan, indien hij $K_s$ bemachtigd, stap 3 herhalen, zodat de aanvaller nu de Bob zijn response kan beantwoorden. Bob is dan aan het praten met de aanvaller, terwijl hij denkt dat het Alice is. Een beter algoritme:

1. Alice informeert Bob dat ze wil communiceren.
2. Bob antwoordt met een geëncrypteerde nonce $N_2$.
3. Alice vraagt aan de KDC om communicatie te starten met Bob. Alice verstuurt een andere nonce $N_1$, samen met de geëncrypteerde nonce $N_2$.




Figuur 2.2: Verbeterde werking KDC.

4. De KDC beantwoordt dit met een geëncrypteerd bericht, gebruikmakend van Alice haar sleutel $E_{K_{A-KDC}}$ (De KDC authenticeerd zich op die manier naar Alice toe, en bereikt confidentialiteit). De inhoud van dit is: de sessiesleutel, Bob zijn identiteit, $N_1$, en een geëncrypteerd bericht met de sessiesleutel, $N_2$ en Alice haar identiteit.
5. Alice verstuurt de sessiesleutel, $N_2$ en haar eigen identiteit geëncrypteerd met de sleutel van Bob naar Bob toe.
6. Bob antwoordt met een tweede nonce $N_2$, geëncrypteerd met de sessiesleutel, zodat Bob geauthenticeerd is naar Alice toe.
7. Alice antwoordt terug naar Bob met een bewerkte nonce (een bepaalde functie, bv 1 toevoegen), geëncrypteerd met de sessiesleutel, zodat Alice geauthenticeerd is naar Bob toe.

ToDo: PKI

## 2.3 Beveiligen van netwerkprotocollen

### 2.3.1 Beveiligen van de transportlaag

Het belangrijkste protocol is **SSL (Secure Socket Layer)**. Het **TLS (Transport Layer Security)** protocol is de opvolger van SSL, zodat TLS dus ook alle features heeft van de laatste SSL versie. Hedendaags wordt enkel nog TLS gebruikt.

In eerste instantie kan SSL/TLS vergeleken worden met SSH, aangezien SSH ook de transportlaag beveiligd. De verschillen tussen SSH en SSL/TLS zijn:

- SSL/TLS is ontworpen om generiek verkeer over de transportlaag te beveiligen.
- SSH bevat ook nog multiplexing, user authenticatie en terminal management.

- SSL gebruikt X.509 certificaten, SSH gebruikt een eigen formaat.
- Verschillende optimalisaties: SSH voor shell applicaties en TLS voor betere performantie bij https.

Verder wordt enkel nog TLS behandelt, SSL kan gegenegeerd worden.

De verschillen tussen SSH en TLS zijn:

- **TLS server authenticatie is optioneel.**
  - Anonieme operaties zijn toegelaten.
  - Hierdoor is een man-in-the-middle attack een zeer eenvoudige aanval.
  - Server authenticatie in SSH is verplicht.
- **TLS heeft geen user authenticatie.**
  - TLS moet enkel de twee interfaces authenticeren (SSH kan dit ook).
  - User authentication wordt behandeld door een bovenliggende laag.
- **TLS gebruikt X.509 public-key certificaten om te authenticeren.**
  - SSH heeft een eigen certificaatformaat.
  - Vereist een werkend PKI systeem.
  - Een voordeel van een PKI systeem is dat het schaalbaar is op vlak van sleutelbeheer, iets wat SSH nog niet kan.
- **TLS kent enkel public-key authenticatie**
  - SSH heeft ook nog host-based, password, ...
- **TLS heeft niet de extra features die het SSH connection protocol wel heeft**

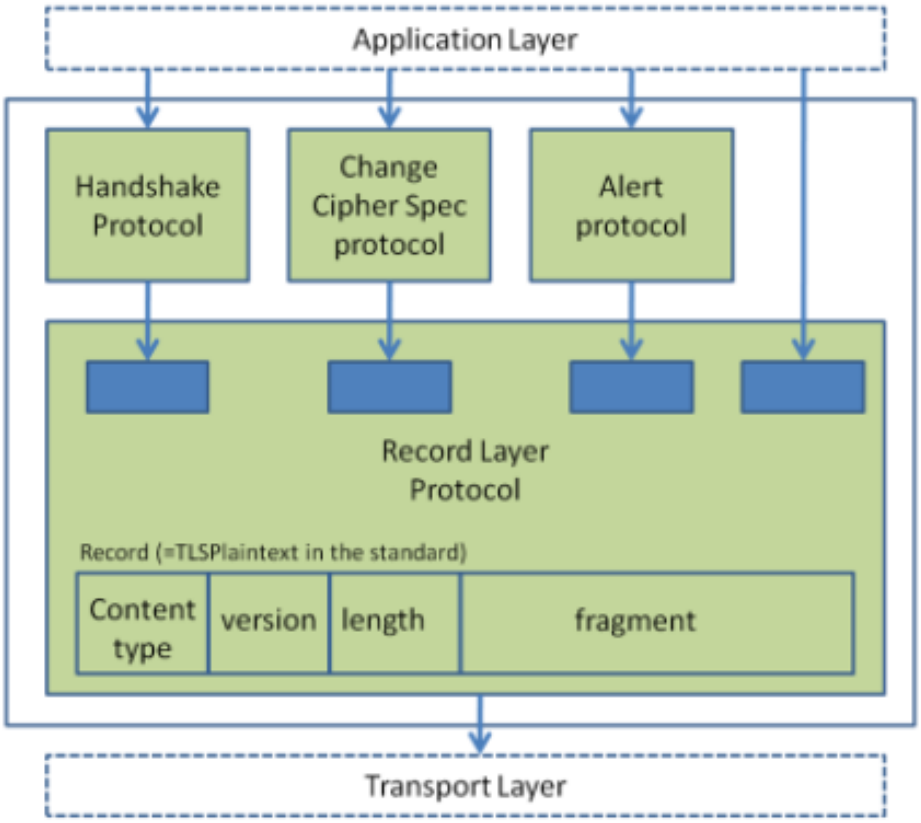
### Architectuur

- Een **TLS connectie** is een communicatiekanaal tussen een cliënt en een server. Deze connecties zijn vaak van korte levensduur en servers zullen een connectie zelf vernietigen na een bepaalde tijdsduur indien de connectie te lang op idle modus staat.
- Een **TLS sessie** wordt gebruikt om de server een bepaalde staat te geven. De connectie kan gesloten worden, maar kan een sessie behouden. Deze sessie kan dan verdergezet worden met een nieuwe connectie. Een sessie wordt op zowel de client als de server bewaart. Er kan ook een nieuwe sessie aangemaakt worden tijdens een connectie. Een sessie wordt gedefinieerd aan de hand van een aantal parameters:
  - De sessieidentifier, een random byte sequentie, gegenereerd door de server.
  - De optionele X.509 certificaten.
  - De optionele compressiemethode.

- De *master secret*, een gedeelde sleutel van 48 bytes die gekend is door de cliënt en de server.
- Een bit *is resumable*, die aangeeft of dat de sessie kan gebruikt worden om nieuwe connecties aan te maken.

Figuur 2.3 toont aan dat TLS gebruik maakt van een tweelagen architectuur met

1. het TLS record layer protocol en,
2. het TLS-specifieke protocollen: TLS Handshake protocol, TLS Change Cipher Protocol en TLS Alert Protocol.




Figuur 2.3: Tweelagen architectuur van TLS

- Het **Handshake protocol** laat toe om de cliënt en server te authenticeren aan elkaar. Een HP bericht heeft de volgende structuur:
  - 1 byte voor de berichtsoort (10 typen gedefinieerd).
  - 3 bytes voor de berichtgrootte.
  - variabel aantal bytes voor de eigenlijke informatie, soms zelfs leeg.

Het handshake protocol verloopt in 4 fasen:

1. Eerst moet een TCP sessie aangemaakt worden via het normale TCP process: SYN → SYN-ACK → ACK.
2. Daarna verstuurt de clënt een **client_hello** bericht naar de server met volgende parameters:
   - **Version**: De hoogste TLS versie waarover de cliënt beschikt.
   - **Random**: Een timestamp (32 bits) en 28 bytes gegenereerd door een veilige randomgenerator, dat gebruikt wordt als nonce.
   - **SessionID**: Een identifier dat gebruikt wordt om de sessie te identificeren. Wordt onder andere gebruikt om een sessie verder te zetten met een andere connectie.
   - **CipherSuite**: Een lijst van combinaties van cryptografische algoritmen die de cliënt ondersteunt. Een combinatie bestaat uit een sleuteluitwisselingsmethode, de encryptiespecificatie, de MAC functie en de pseudorandomfunctie dat gebruikt wordt.
   - **CompressionMethod**: Een lijst van compressiemthoden waarover de cliënt beschikt.

   De server reageert met een **server_hello** bericht die dezelfde structuur bevat als de client_hello variant. De server neemt uiteraard de versie en cryptografische algoritmen dat hijzelf ook ondersteund.
3. Indien authenticatie vereist is, stuurt de server een X.509 certificaat naar de cliënt. **Optioneel** zal de server ook een certificaat vragen aan de cliënt, indien authenticatie vereist is. De server eindigt met een **server_done** bericht. De cliënt antwoordt dan indien nodig met zijn certificaat. De cliënt stuurt ook een **client_key_exchange** bericht, met de nodige informatie om een sessiesleutel te genereren.
4. De cliënt verstuurt nu een **change_cipher_spec** bericht (met het Change Cipher Protocol), die de juiste combinatie van cryptografische algoritmen bevat. Tot slot stuurt de cliënt een **finished** bericht. De server verstuurt dezelfde berichten naar de cliënt. Nu zijn de cliënt en server in staat om te communiceren met elkaar via het SSL Record protocol.

- Het **Change Cipher protocol** kan slechts één bericht verstaan: 1 byte met 1 waarde. Het zorgt ervoor dat de tijdelijke keuze van cryptografische algoritmen de werkelijke keuze wordt. Het kan ook gebruikt worden om de encryptiealgoritmen te veranderen gedurende een connectie.

- Het **Alert protocol** bevat berichten met foutmeldingen en waarchuwingen. Een bericht bestaat enerzijds uit één byte, die "fatalöf "warningäangeeft, en een andere byte die meer uitleg geeft over de fout.

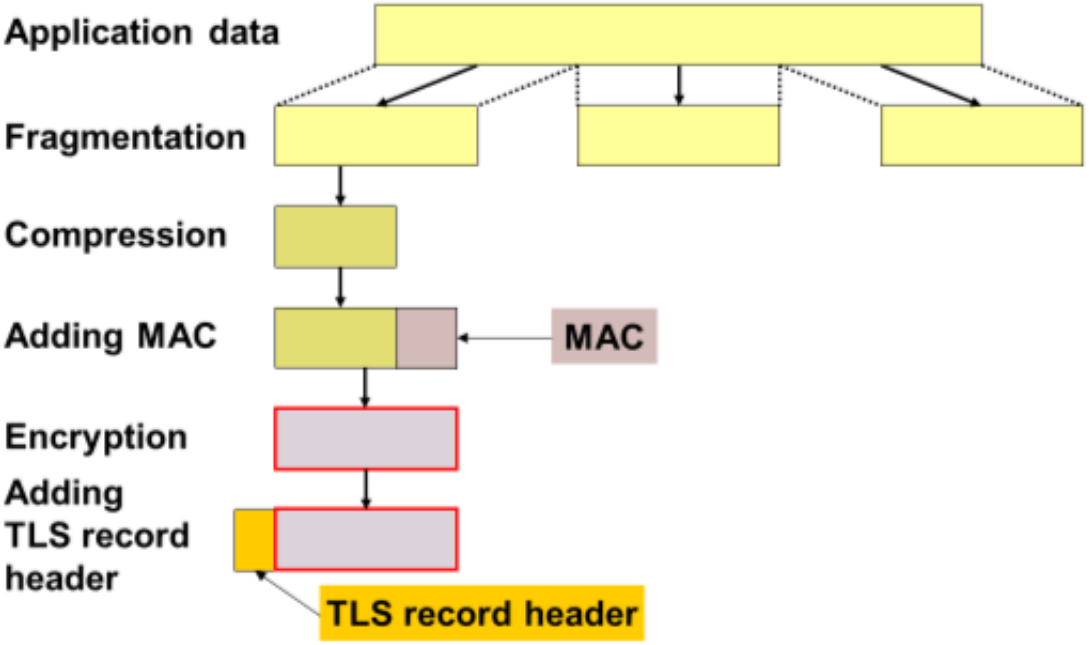
  Mogelijke waarschuwingen:

  - Problemen met certificaten.
  - De connectie wordt verbroken door één van de partijen.

  Mogelijke fouten:

  - Problemen met het decompresseren.
  - Een fout tijdens het handshake protocol (geen geldige beveiligingsparameters gevonden).
  - Fout ingevulde parameters tijdens het handshake protocol.

- Het **Record protocol** is een gelaagd protocol dat informatie zo zal bewerken dat: het bruikbare blokken worden van ten hoogste $2^{14}$ bytes, optioneel de informatie zal compresseren, een MAC zal toevoegen voor integriteit, de informatie zal encrypteren en het resultaat doorsturen. Ontvangen data wordt dan in omgekeerde volgorde bewerkt. De typische workflow gaat als volgt (figuur 2.4):




Figuur 2.4: Werking van het TLS record protocol.

1. De informatie wordt verdeeld in fragmenten van hoogstens $2^{14}$ bytes ($\equiv$ 16 KB).
2. De informatie wordt eventueel gecompresseerd. De compressie is optioneel, moet terugkeerbaar zijn, en mag de lengte van de data niet meer dan 1024 bytes vergroten.
3. De MAC wordt als volgt berekend: MAC($K_{MAC}||Seq\#||Type||Length||Data$).
4. De informatie wordt geëncrypteerd met een symmetrisch encryptiealgoritme.
5. De TLS record header wordt toegevoegd.

Het verschil met SSH:

- SSH zal eerst encrypteren, en dan pas de MAC toevoegen. Geen van beide methoden is beter dan de andere, en komt met zijn eigen problemen.

De TLS record header heeft volgende structuur (figuur 2.5):

- 1 byte voor het contenttype. Dit is het applicatielaagprotocol dat gebruikt wordt.
- 2 bytes voor de versie (voor TLS 1.2 heeft dit de waarden 3 en 3).


Figuur 2.5: De TLS record header.

  - Lengte van de data.

- De **master secret**, die nodig is bij een TLS sessie, wordt ofwel gegenereerd door de cliënt met RSA encryptie en verzonden naar de server, ofwel uitgewisseld met Diffie-Helman. De grootte van de sleutel bedraagt 384 bits. Sleutels kunnen niet hergebruikt worden aangezien ze afhankelijk zijn van de nonces die gebruikt werden tijdens het handshake protocol.
- Het **heartbeat protocol** draait bovenop het record protocol en heeft voornamelijk als doel om huidige connecties te testen, en om ze in leven te houden zonder dat er opnieuw moet onderhandeld worden tussen de cliënt en de server. Het protocol ondersteund twee berichten:
  - **HeartbeatRequest**: De cliënt of server verstuurt naar de ander een bericht met daarin een payload (typisch een string) en de lengte van de payload.
  - **HeartbeatResponse**: Een ontvanger van een HeartbeatRequest moet antwoorden met dezelfde payload als een HeartbeatResponse.

_ToDo: P20 printout C03c

### 2.3.2 Beveiligen van de netwerklaag

### 2.3.3 Beveiligen van de datalinklaag

# Hoofdstuk 3

# Encryptiealgoritmen

## 3.1 Geschiedenis

Encryptiemethoden worden al eeuwenlang gebruikt om informatie onleesbaar te maken voor partijen die deze informatie niet mogen achterhalen. Deze inleiding bespreekt de basismethoden om encryptie toe te passen.

Zogenaamde **substitution ciphers** zijn de meest eenvoudigste vorm van encryptie. De originele versie, ook wel Caesar cipher genoemd, vervangt elke letter van het alfabet met een voorgedefinieerde shift in positie. Een shift van 3 zal het originele alfabet (in kleine letters) afbeelden op het verschoven alfabet (in grote letters): Het duidelijke nadeel is dat er slechts 25 verschillende encryp-

a b c d e f g h i j k l m n o p q r s t u v w x y z
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

tiemogelijkheden zijn. Een verbeterde versie mapt elke letter met een willekeurige andere (nog niet gebruikte) letter: waardoor er nu 26! $\approx 4 \cdot 10^{26}$ mogelijkheden zijn. Deze manier is daarom niet meer

a b c d e f g h i j k l m n o p q r s t u v w x y z
D K V Q F I B J W P E S C X H T M Y A U O L R G Z N

secuur, aangezien de onderliggende frequentie van letters nog steeds dezelfde is. Een eenvoudige dencryptiemethode is om de relatieve frequenties van de ciphertekst te tellen, waardoor er statistisch kan achterhaald worden met welke gedecrypteerde letter elke geëncrypteerde letter overeenkomt.

Een encryptiealgoritme wil ook de relatieve frequenties verbergen. Een methode dat dit implementeert is de Vigenère Cipher. Deze bevat een sleutel $K = k_1 k_2 ... k_d$ waarbij $k_i$ het $i$-de alfabet specifieert dat gebruikt moet worden. Na $d$ letters start ment terug vanaf $k_i$.

Een laatste voorbeeld van een substitutiecipher is de Playfair cipher. Hier wordt er een $5x5$ matrix opgesteld, waarbij eerst het sleutelwoord ingevuld wordt, gevolgd door de niet gebruikte letters in alfabetische volgorde. Dit wordt uitgewerkt met de sleutel `MONARCHY` in Tabel 3.1. Een bericht wordt in paren van letters geëncrypteerd. Stel dat we het woord ballon willen encrypteren, de paren letters

| | | | | |
|---|---|---|---|---|
| M | O | N | A | R |
| C | H | Y | B | D |
| E | F | G | I/J | K |
| L | P | Q | S | T |
| U | V | W | X | Z |

Tabel 3.1: Playfair cipher.

worden: ba lx lo nx. Er wordt een x toegevoegd indien er twee dezelfde letters achter elkaar komen, en als het eindpaar uit slechts één letter bestaat. Er kunnen zich drie gevallen voordoen bij elk paar:

1. Als beide letters in dezelfde rij voorkomen, wordt elke letter vervangen door de letter die er rechts van ligt: (on → na).
2. Als beide letters in dezelfde kolom voorkomen, wordt elke letter vervangen door de letter die er onder van ligt: (ba → ib).
3. Anders wordt elke letter vervangen door de letter van de kolom van de andere letter: (lx → su).

Er zijn $26 * 26 = 676$ mogelijke diagrammen die gemaakt kunnen worden.

Na de **substitution ciphers** zijn er ook **transposition ciphers**. Zulke ciphers gaan letters niet vervangen, maar gaan ze echter verplaatsen. Dit heeft natuurlijk als nadeel dat de relatieve frequentie van de letters behouden wordt. Twee eenvoudige voorbeelden zijn:

1. Keer elke letter om:
   `A SIMPLE EXAMPLE` → `ELPMAXE ELPMIS A`
2. Keer de woordvolgorde om, en elk woord keert ook de lettervolgorde om:
   `A SIMPLE EXAMPLE` → `A ELPMIS ELPMAXE`

Een iets beter voorbeeld is de Rail Fence cipher. Deze heeft als private sleutel het aantal rijen dat gebruikt wordt. Elke letter zal diagonaal geschreven worden, uitgewerkt in Tabel 3.2 op de zin `DEFEND THE EAST WALL`: Geëncrypteerd is dit dus `DNETLEEDHESWLXFTAAX`.

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D | | | | N | | | | E | | | | T | | | | L | | |
| | E | | E | | D | | H | | E | | S | | W | | L | | X | |
| | | F | | | | T | | | | A | | | | A | | | | X |

Tabel 3.2: Rail Fence cipher.

De laatste methode die besproken wordt is de columnar transposition cipher. Een bericht wordt in rijen geschreven over een bepaald aantal kolommen. Hierna worden de kolommen gesorteerd op basis van de geheime sleutel $K = k_i k_j ... k_n$. Stel $K = k_3 k_4 k_2 k_1 k_5 k_6 k_7$ en plaintext `ATTACK POSTPONED UNTIL TWO AM`: Op Tabel 3.3 wordt deze plaintext uitgeschreven in rijen en kolommen. De ciphertext wordt dan `TTNAAPTMTSUOAODWCOIXKNLYPETZ`.

| | | | | | | |
|---|---|---|---|---|---|---|
| A | T | T | A | C | K | P |
| O | S | T | P | O | N | E |
| D | U | N | T | I | L | T |
| W | O | A | M | X | X | X |

Tabel 3.3: Rail Fence cipher.

Uiteindelijk kan men ook de combinatie maken van **substitution** en **transposition** ciphers, wat de basis vormt van moderne cryptografie.

## 3.2 Symmetrische algoritmen

Vooraleer er in detail kan ingegaan worden op symmetrische algoritmen, moet eerst de term **block cipher** besproken worden. Een block cipher wil zeggen dat informatie in blokken zullen verstuurd worden en kan best vergeleken worden met een substitutiecipher, maar dan toegepast op blokken. Een typische blokgrootte varieert van 8 tot 128 bytes en wordt door de meeste algoritmen gebruikt. De **Feistel Cipher encryptie** vormt de basis van moderne blockciphers en maakt gebruik van twee primitieve encryptieoperaties:

- Substitutie (de S-box)
- Permutatie (de P-box)

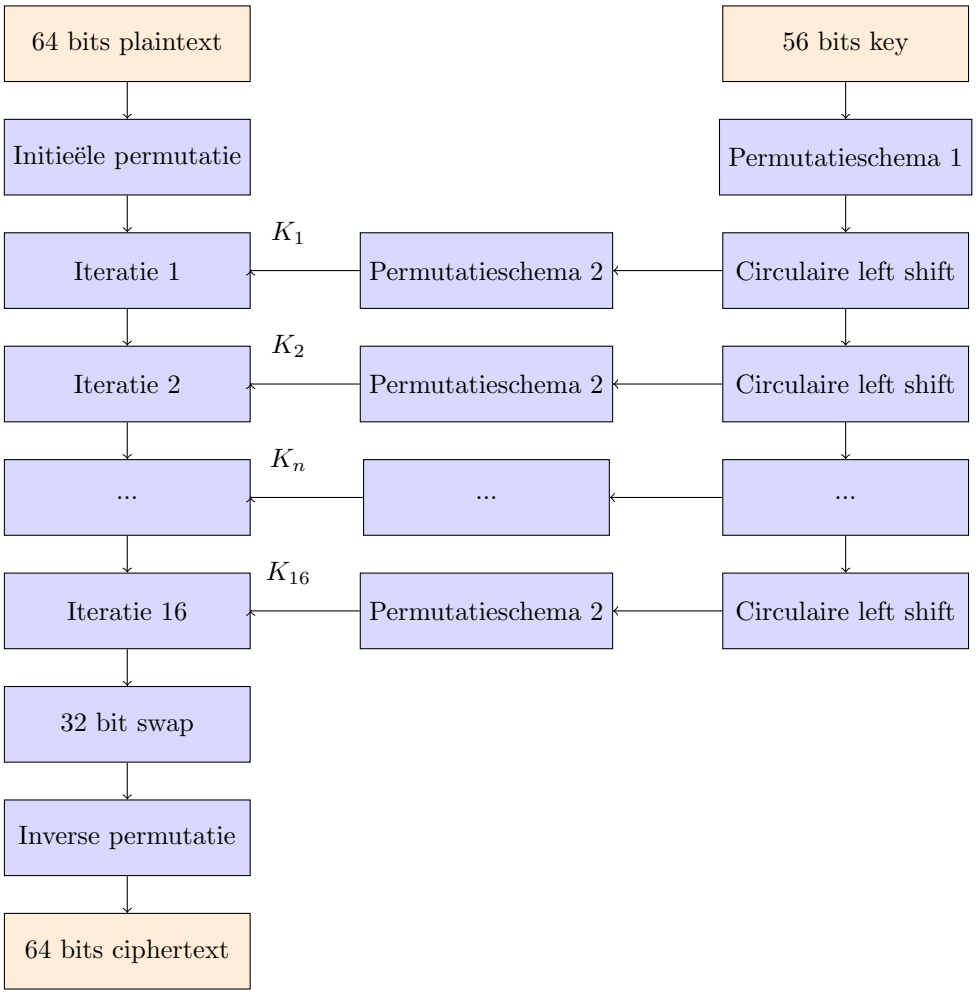
Het Feistel encryptieschema maakt gebruik van $n$ ronden, een functie $F$ en de XOR operator. ToDo: slide 36 uitleggen

### 3.2.1 DES & 3-DES

Het **DES (Data Encryption Standard)** algoritme maakt gebruik van het reeds vermelde Feistel schema waarbij de **blokgrootte 64 bits** en **sleutelgrootte 56 bits** bedraagt. Deze methode is redelijk traag, ondanks de kleine blokgrootte, en wordt door de kleine sleutelgrootte ook als onveilig beschouwd. Een 56 bit sleutel kan namelijk binnen de 10 uur gekraakt worden indien het aantal operatiers per **micro**seconde $10^6$ bedraagt.

Het algoritme maakt gebruik van 16 ronden in het Feistel schema. De 56 bit sleutel wordt gebruikt om iteratiesleutels $K_1, K_2, ...K_{16}$ te bepalen, die elk 48 bits groot zijn. Voor de eerste ronde en na de laatste ronde wordt er een extra permutatie uitgevoerd, waarbij de laatste permutatie het inverse is van de eerste permutatie. Het resultaat van dit algoritme is één van de $2^{64}$ uitvoermogelijkheden van de 64 bits.

De 16 vereiste iteratiesleutels worden in het begin van het algoritme aangemaakt. Eerst en vooral wordt de 56 bit sleutel gepermuteerd, afhankelijk van het gekozen permutatieschema, op Figuur 3.1 permutatieschema 1 genoemd. Een permutatieschema mapt een bit van de oorspronkelijke sleutel op een bit van de gepermuteerde sleutel, zodat er $2^{64}$ mogelijke schemas bestaan. De gepermuteerde sleutel wordt hierna beschouwd als twee resultaatsleutels $C_n$ en $D_n$, met $n$ het huidig rondenummer, van elk 28 bits groot. Beide resultaatsleutels worden onafhankelijk van elkaar verwerkt door een




Figuur 3.1: DES schema.

circulaire left shift van 1 of 2 bits, afhankelijk van de huidige ronde, dat eenvoudig door de conditionele operator kan beschreven worden: $n \,\%\, == 0 \;?\; 2 \;:\; 1$. Indien $C_2$ en $D_2$ de resultaatsleutels zijn van ronde 2, dan zijn $C_3$ en $D_3$ de resultaatsleutels van ronde 3 door respectievelijk $C_2$ en $D_2$ 2 bits naar links te permuteren. Voor elke ronde wordt er een iteratiesleutels $K_n$ gegenereerd, door een tweede permutatieschema, op Figuur 3.1 permutatieschema 2 genoemd, toe te passen op $C_n D_n$, waarbij de 8 meest significante bits van $C_n$ genegeert worden. Het eindresultaat zijn 16 verschillende iteratiesleutels, die zullen toegepast worden bij elke DES ronde.

Bij elke ronde wordt het 64-bit blok opgedeeld in twee 32-bit blokken $L_n$ en $R_n$ waarbij de 32 meest significante bits in $L_n$ komen, en de 32 minst significante bits in $R_n$. Vooraleer dit proces plaatsvindt is er eerst een initiele permutatie van het hele 64-bit blok. Deze permutatie wordt uitgevoerd met behulp van een permutatieschema, zoals bij de sleutelgeneratie. Het gepermuteerde blok wordt hierna opgedeeld in de blokken $L_n$ en $R_n$, en kan ronde 1 van start gaan. Bij elke iteratie wordt de functie $F$ gebruikt, die twee blokken manipuleert: $R_{n-1}$ en de sleutel van 48-bit groot,

die op voorhand al gegenereerd werd. Omdat hij elke iteratie wordt gebruikt wordt het ook wel de iteratiefunctie genoemt. De uiteindelijke uitvoer van de iteratiefunctie is een blok 32 bits. $L$ en $R$ kan voor een volgende iteratie berekent worden als:

$$\begin{aligned} L_{n+1} &= R_n \\ R_{n+1} &= L_n \oplus F(R_n, K_{n+1}) \end{aligned} \tag{3.1}$$

Hoe werkt de iteratiefunctie? Eerst moet het blok $R_n$ uitgebreidt worden via een selectietabel. Het gebruik van deze selectietabel wordt hier voorgesteld als de functie $E(R_n)$, die als input een blok van 32-bit en zal als output een blok van 48-bit genereren waarbij de bits gegroepeerd per 6 zitten. Sommige bits van $R_n$ worden zullen dus meerdere keren gemapt moeten worden. Na deze uitbreiding wordt het resultaat van $E(R_n)$ en de sleutel $K_n$ met een XOR operatie bewerkt (XOR geeft 0 indien beide bits gelijk zijn, en 1 als ze verschillend zijn). Elke groep van 6 bits stelt een adres voor in een welbepaalde substitutie tabel $S_i$, voor de $i$-de groep van 6 bits: er zijn dus 8 substitutietabellen. Elk adres wijst naar een 4-bit getal, dat de originele 6 bits zal vervangen, zodat het eindresultaat terug een blok van 32 bits is. Hoe worden deze 4 bits bepaald? De substitutietabel neemt een blok van 6 bits $B$ als input, de bits $b_5b_4b_3b_2b_1b_0$ van $B$ doen het volgende:

- $b_5$ en $b_0$ representeren tesamen een getal van 0 tot en met 3. Stel dit getal gelijk aan $i$.
- De tussenliggende bits representeren tesamen een getal van 0 tot en met 15. Stel dit getal gelijk aan $j$.
- Zoek in de tabel het getal op de $i$-de rij en $j$-de kolom. Dit getal heeft een waarde tussen 0 en 15, en bevat dus slechts 4 bits. Deze bitrepresentatie van dat getal vervangt de 6 bits.

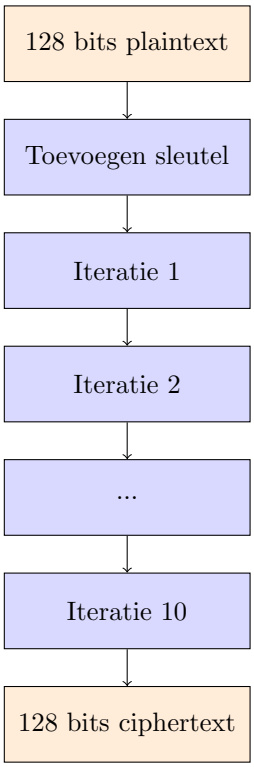
Tot slot moet er nog een permutatie $P$ uitgevoerd worden op de 32 bits verkregen van de substitutiestap. Met behulp van de iteratiefunctie worden opeenvolgende $L_i$ en $R_i$ bepaalt (formule 3.1).

### Block cipher modi

Er zijn verschillende implementaties van het DES-algoritme, en deze hangen dan weer af van de gebruikte block cipher modus. Verschillende mogelijkheden zijn:

- **ECB (Electronic Code Book).** Elk 64-bit blok van oorspronkelijke informatie wordt individueel geëncrypteerd
- **CBC (Chain Block Chaning).** Elk cipherblok is nu afhankelijk van elkaar, door een initiële XOR-operatie uit te voeren met het vorige geëncrypteerde blok, en het te encrypteren blok.
- **CFB (Cipher Feedback).** Gelijkaardig aan CBC, maar ToDo: wat?

ToDo: rest van block cipher mody en 3-DES




Figuur 3.2: AES schema.

### 3.2.2 AES

_ToDo: no clue what w[x, y] wil zeggen + DECRYPTIE (SLIDE 96)

Het **AES (Advanced Encryption Standard)** algoritme maakt is ook een symmetrisch encryptiealgoritme waarbij de **blokgrootte 128 bits** en **sleutelgrootte 128, 192 of 256 bits** bedraagt. Het werd speciaal ontworpen om cryptoanalyse moeilijk te maken en is zelfs sneller dan 3DES op 32-bit toestellen. Op Figuur 3.2 wordt het schema getoond voor een blokgrootte van 128 en een sleutelgrootte van 128 bits. Het schema is gelijkaardig voor een sleutelgrootte van 192 of 256 bits, maar hebben dan respectievelijk 12 en 14 iteraties. Het algoritme is ongeveer 2,5 maal sneller dan het DES algoritme (200 Mbit/s versus 80 Mbit/s) en werkt uitsluitend voor CBC of CTR block cipher modi. Bovendien is het heel goed **paralleliseerbaar**.

## 3.3 Asymmetrische algoritmen

# Hoofdstuk 4

# Software- en systeembeveiliging