**H1 Introductie Cryptografie**

* 1. **Classificatie Cryptografische systemen p basis van 3 eigenschappen**
* Operaties om van X => Y te gaan
  + Substitutie
    - Vervanging van tekens
    - Caesar
      * (X + #plaatsen\_te\_verschuiven) % 26 is de sleutel
      * 26 sleutels
      * Gevoelig aan frequentie-analyse
    - Vigenère
      * Met een sleutelwoord als key
      * Tabel met combinaties
  + Transpositie
    - Veranderen van de volgorde
    - Sleutelwoord is key
      * Aan de hand van het sleutelwoord de positie bepalen
* Aantal gebruikte sleutels
  + Symmetrisch – secret key – dezelfde sleutel
  + Asymmetrisch - public key – verschillende sleutel
* Operaties om van input => X te gaan
  + Blokvercijfering
  + Stroomvercijfering
  1. **Stenografie**

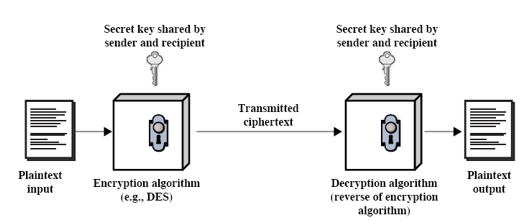
Het verbergen van informatie in afbeeldingen en/of audiofragmenten. Dit kan in combinatie met cryptografie.

* Afbeeldingen
  + Least Significant Bit (LSB)
  + RGB waarden van een pixel (8bits, 8bits, 8bits)
  + Laatste bit van iedere kleur vervangen door bit van de te verbergen data
* Geluid
  + Menselijk gehoor kan maar bepaalde frequenties waarnemen
  + Niet waarneembare frequenties gebruiken om data in te verstoppen

**H2 Symmetrische Cryptografie**

**2.1 Wat is symmetrische cryptografie**

Bij symmetrische cryptografie wordt dezelfde sleutel gebruikt voor zowel de encryptie als de decryptie.



Symmetrische cryptografie heeft twee basisbewerkingen:

* Substitutie
  + X vervangen door Y
* Transpositie
  + X op een andere plaats zetten

Voordelen:

* Snel
* Avalanche effect
  + Door vele iteraties worden kleine foutjes grote fouten en weet de hacker niet of die dicht bij de sleutel zit of niet.

Nadelen:

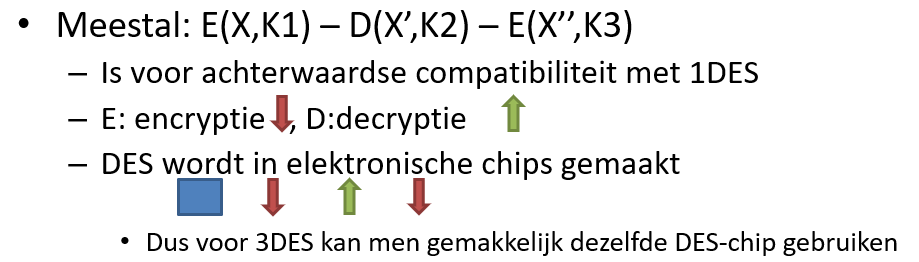
* Sleutelbeheer
  + Sleutel veilig communiceren tussen persoon A en persoon B
  + Aantal sleutels
    - Neemt exponentieel toe naarmate er meer personen bijkomen
    - => Aanleiding tot asymmetrische cryptografie (RSA)

Voorbeelden:

* DES – Data Encryption Standard
* 3DES – 3x Data Encryption Standard
* AES – Advanced Encryption Standard
* IDEA – International Data Encryption Algorithm

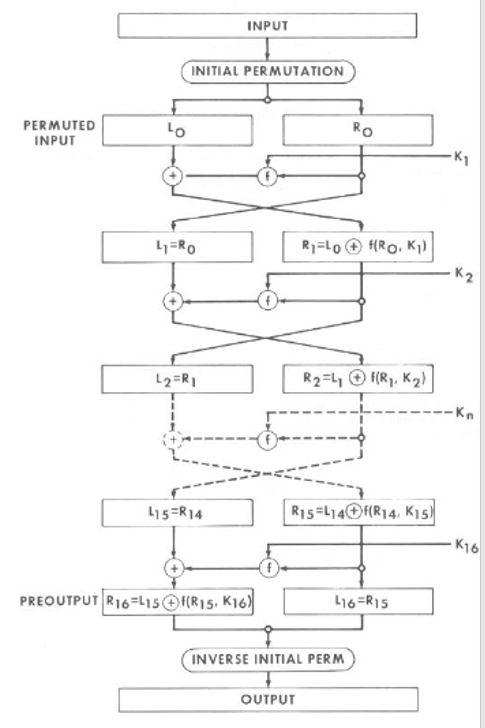
**2.2 (3)DES**

* 1977 standaard in USA
* Gegevens worden omgezet naar binaire waarden
* Encryptie en decryptie werken met dezelde sleutel
* Is een block cipher – blokken van 64 bits
* Avalanche-effect
  + Een kleine verandering in X of K genereert een cascade effect en zorgt voor een totaal andere Y.
  + M.a.w. door de 16 iteraties wordt een kleine fout steeds groter
* Weak keys
  + Als er alleen maar 1’en of 0’en zijn.
    - 2x encryptie levert dezelfde tekst als resultaat
* is doenbaar om te brute forcen
  + 3X DES -> sleutel
  + Onkraakbaar



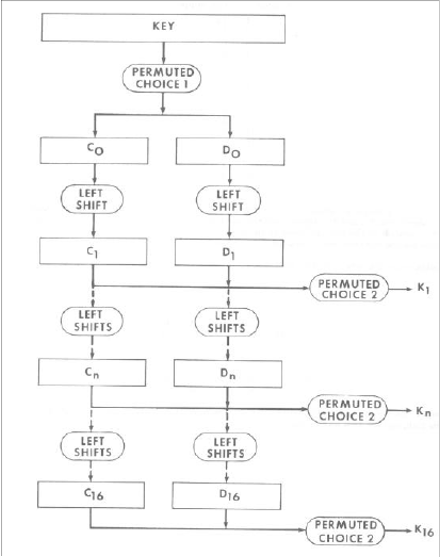
**2.2.1 Algemene werking**

1. De 64-bit invoer wordt gesplitst in 2x 32bits door een initiële permutatie ( een permutatie matrix die de bits door elkaar mixt)
2. Nu hebben we een linkerblok van 32 bits en een rechterblok van 32 bits
3. Nu volgen 16 iteraties met 16 subsleutels afgeleid van sleutel K
   1. Iteratie:
   2. Bij iteratie 16 wordt er niet meer van kant gewisseld
      1. Symmetrie bewaren
4. Op het einde worden de 2 blokken door een inversie permutatiematrix terug samengevoegd tot een blok van 64 bits.



**2.2.2 Aanmaken van subkeys**

* De 16 subsleutels van 48 bits worden gemaakt de 64-bit K
* K gaat door een permutatiematrix die 8 bits, controle bits, afneemt en K verdeelt over 2 blokken van 28 bits
* De 2 blokken van 28 bits worden dan elke iteratie naar links geshift
* De 2 blokken worden weer samengevoegd door een permutatiematrix, die ook weer 8 bits afneemt, en zo krijgen we een key van 48 bits

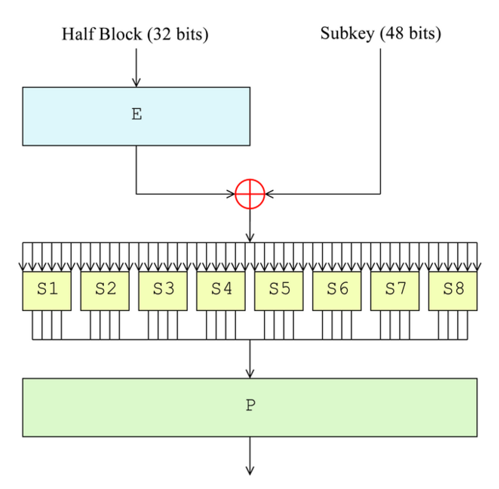


**2.2.3 De Feistel functie f(Rn, Ki)**

* Heeft als parameters het rechterblok van 32 bits en een subkey van 48 bits
* Voor elke iteratie wordt er een nieuwe subkey gemaakt
* Om R te kunnen XOR met Ki moet R groter gemaakt worden.

Werking:

1. R wordt groter gemaakt door een expansiematrix die bepaalde bits gaat dupliceren op bepaalde posities.
2. R XOR Ki
3. Het resultaat van 48 bits wordt verdeeld over 8 S-boxen, elke S-box krijgt 6 bits
4. Elke S-box krijgt 6 bits en geeft er 4 terug
   1. Door middel van S-box tabellen. Elke S-box heeft een verschillende tabel
      1. Rij: MSB en LSB van de input samennemen
      2. Kolom: middelste 4 bits
5. Dan krijgen we in totaal terug 32 bits
6. Het 32-bit resultaat wordt door elkaar geschud door een permutatie-matrix



**2.3 AES**

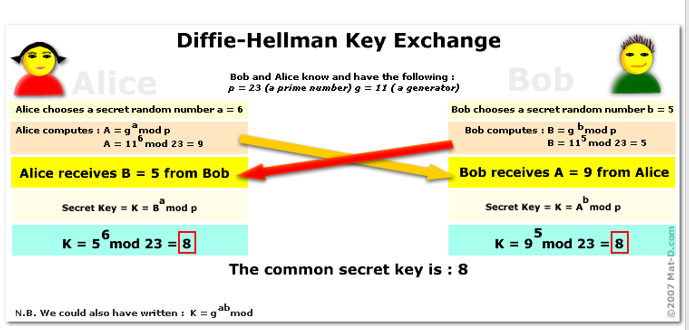
* Opvolger DES – 2001
* Symmetrisch
* Sleutels van 128/192/256
* Onkraakbaar

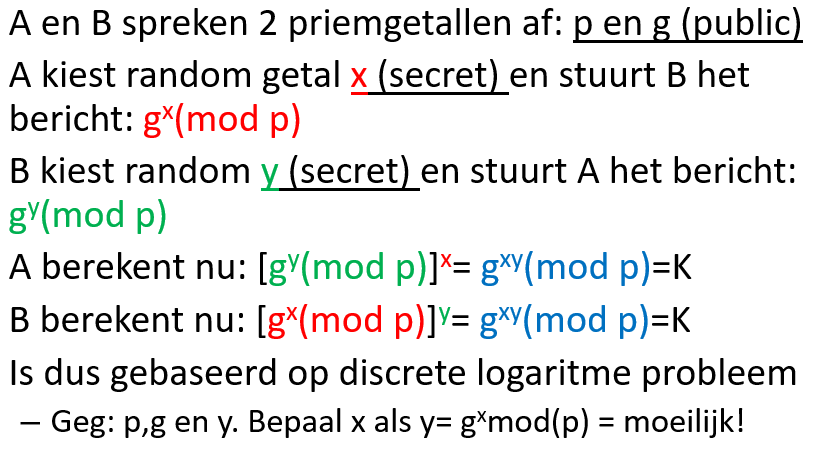
**H3 Asymmetrische Encryptie**

3.1 Asymmetrische cryptografie

* Encrypteren
  + Ervoor zorgen dat alleen de ontvanger de gegevens kan decrypteren
* Authenticatie
  + Bewijzen dat de doorgestuurde gegevens effectief van A zijn
* Werkt met public en private keys
  + Public keys zijn bekend
  + Private keys blijven geheim
  + Public en private key kunnen niet van elkaar worden afgeleid maar houden wel een wiskundig verband
* 4 grote principes
  + Ontbinden van grote priemgetallen (RSA) – factorisatie
  + Diffie-Hellman => discrete logaritmen berekenen
    - Zie volgend onderdeel
  + Knapzaksysteem
    - Rugzakken met objecten in met een bepaald gewicht
    - Niet meer veilig
    - Backdoor om de private key van de public key af te leiden
  + Elliptische curven
  + Gebaseerd op ellipsen
    - killer van RSA
    - zeer snel
    - kortere sleutels
    - hogere beveiliging
* Voordeel;
  + Keymanagement
* Nadeel:
  + Traag

**3.2 Diffie-Hellman**







**Man in the Middle-attack (MITM):**

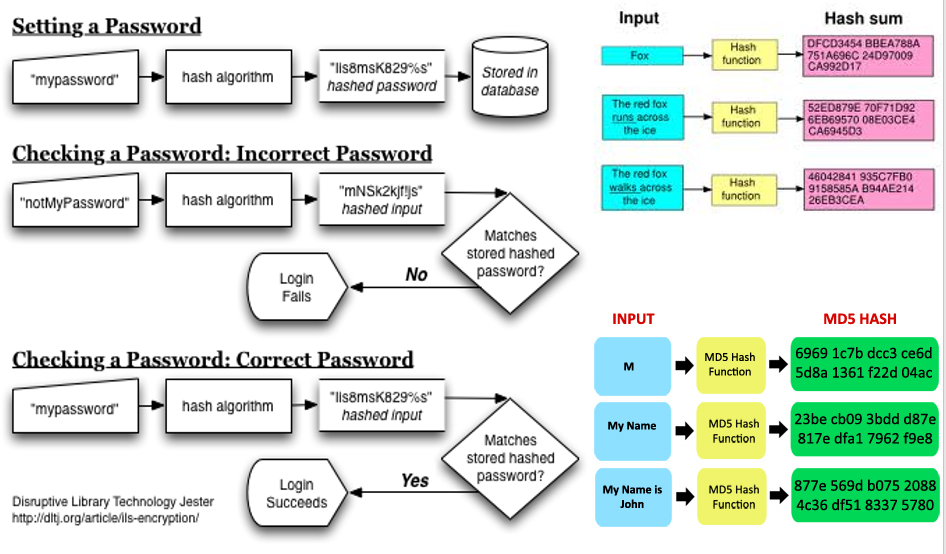
* Een aanval waarbij informatie tussen twee communicerende partijen onderschept wordt zonder dat beide partijen daar weet van hebben.
* Een aanvaller C stelt zich op tussen A en B
* Er gebeurt nu communicatie tussen A-C en C-B
* A-C spreekt sleutel K1 af, C-B sleutel 2
* A en B hebben op deze manier niet door dat iemand hun berichten onderschept en eventueel kan aanpassen
* Authenticatie noodzakelijk (digitale handtekeningen)

**H4 Hashing**

**4.1 Hash functions**

Hashes:

* Garanderen de integriteit van een boodschap
  + Geeft een vingerafdruk van een boodschap
  + Niet bedoelt om het bericht onleesbaar te maken
* Is een one-way function
  + F(x) is makkelijk te berekenen maar uit f(x) de x halen is moeilijk.
  + De message kan een variabele lengte hebben
  + De Hash heeft een vaste lengte (256)
    - MD5
    - SHA-1, SHA-2, …
  + Één bit wijzigt -> compleet andere hash
* Eigenschappen:
  + Gemakkelijk en snelle berekening
  + Heel moeilijk om x uit f(x) te bepalen. (one-way)
  + Collision free
    - Heel onwaarschijnlijk dat er een x1 en een x2 zijn die identiek dezelfde hash-waarden hebben
    - Theoretisch mogelijk maar in de praktijk niet.



**4.2 Birthday-attack**

Wachtwoorden worden gehashed bijgehouden. Wachtwoorden hebben een variabele lengte. Hashes een vaste lengte, bijvoorbeeld x. Dus het aantal verschillende wachtwoorden is ∞, en het aantal verschillende hashwaarden is . x is meestal 128 of 256, dus het is nog altijd enorm veel, maar minder dan ∞.

Hackers hebben dus een mapping van een gigantisch groot aantal variabelen naar een fixed-set van hashwaarden. (dictionary)

Stel een hacker raakt binnen in de DB van de wachtwoorden. Dan heeft een hacker veel meer kans om een willekeurige match te vinden dan een gerichte match van hashes met een birthday-attack.

Als er een match gevonden is (hoeft niet uw wachtwoord te zijn) raakt de hacker binnen met het wachtwoord in die tabel.

**4.2.1 Birthday paradox**

Het vinden van een exacte collision is zeer zeer moeilijk, maar het vinden van een willekeurige collision is iets minder moeilijk. Maar toch nog altijd moeilijk.

Aantal willekeurige mensen in een kamer zodat de kans 50% is dat iemand dezelfde verjaardag als u heeft? 256

Aantal willekeurige mensen in een kamer zodat de kans 50% is dat twee willekeurige personen dezelfde verjaardag hebben? 23 (Heel wat minder)

* Het aantal connecties (mappings) tussen de personen neemt exponentieel toe wanneer het aantal kandidaten toeneemt. (probleem van symmetrische encryptie).

**4.2.2 Rainbow tables**

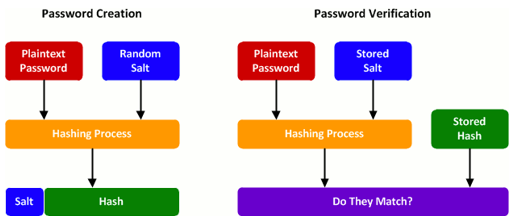
Wachtwoorden worden niet volledig random gekozen en hetzelfde wachtwoord wordt door sommigen meerdere malen gebruikt. Daarnaast is een wachtwoord meestal een combinatie van bestaande woorden

* Hacker gaat dictionaries gebruiken met veelgebruikte passwoorden die in een tabel worden opgeslagen samen met hun hash-waarden.
* Die tabellen worden vergeleken met de hash die men wilt kraken.
* => Rainbow tables

Om rainbow tables tegen te gaan wordt er gebruik gemaakt van een salt:

* Een random waarde die aan het wachtwoord toegevoegd wordt vooraleer men het gaat hashen

Als de hacker de DB kan kraken ziet hij de salt en de hash. Hierdoor zal hij zijn rainbow tables opnieuw moeten berekenen => duurt te lang.



**H5 Hybride cryptografie**

**5.1 Digitale handtekening**

We gebruiken digitale handtekeningen om zeker te zijn van de identiteit van de bestuurder.

Voorwaarden voor digitale handtekening:

* Niet vervalsbaar
  + Alleen afzender kan zijn eigen handtekening zetten
* Authentiek
  + Ontvanger is zeker dat het van die afzender komt
* Niet herbruikbaar
  + Is een deel van de boodschap, kan niet onder ander bericht gezet worden
* Niet veranderbaar
  + Na ondertekening van de boodschap is deze niet meer te veranderen
* Niet verloochenbaar
  + Afzender kan niet ontkennen dat de boodschap van hem kwam

Digitale handtekening => encryptie op het bericht met je private sleutel.

Iedereen kan het decrypteren met je publieke sleutel, maar ze zijn er wel zeker van dat het bericht van jouw kwam.

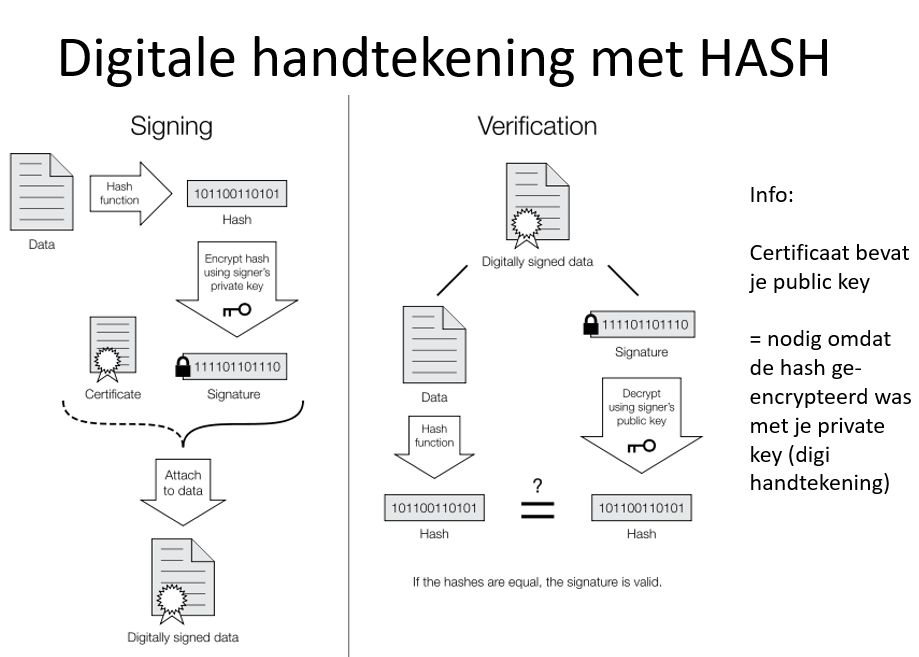
**Digitale handtekening + encryptie**

Geeft de combinatie van authenticiteit + vertrouwelijkheid

1. A zet handtekening op bericht met zijn private sleutel
2. A encrypteert bericht met publieke sleutel van B
3. A stuurt bericht naar B
4. B doet ontcijfering met zijn private sleutel
5. B doet handtekening weg met de publieke sleutel van A

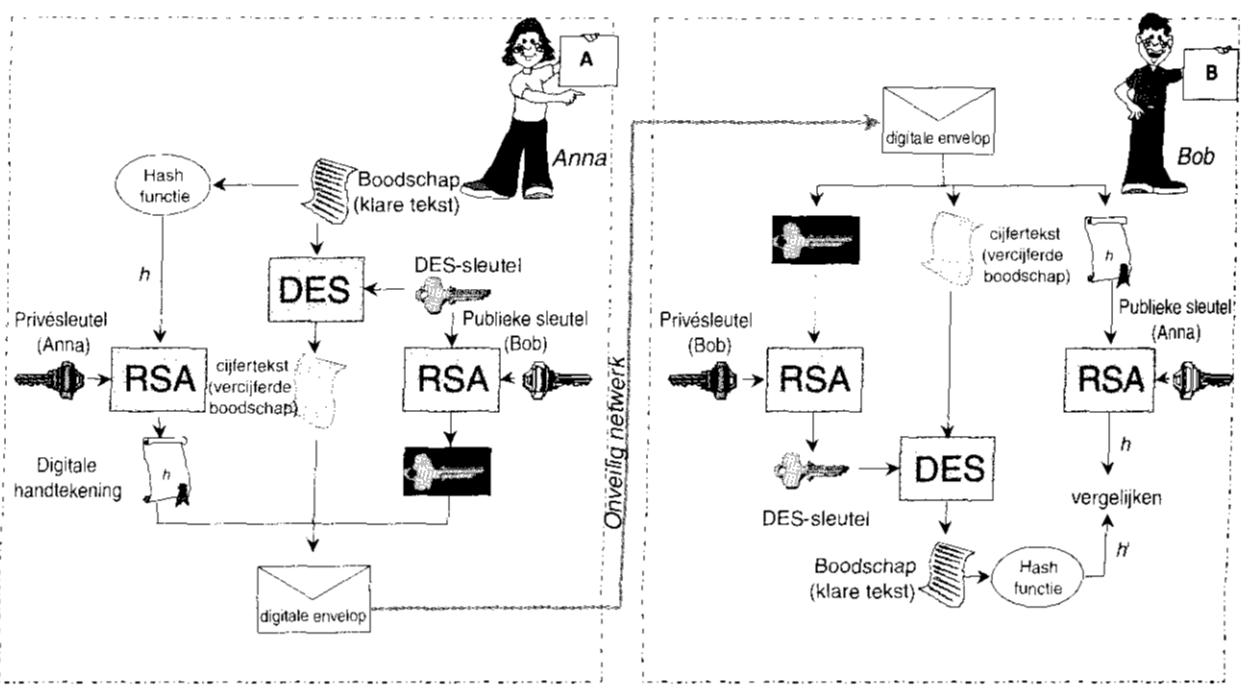
Voor grote berichten en bestanden is dit niet doenbaar want RSA is traag. Daarom tekenen we alleen de hash van het bericht.

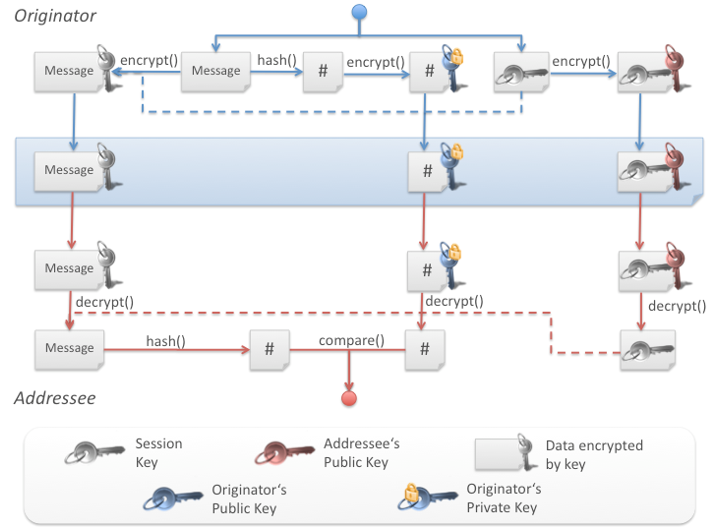
1. A maakt een hash H uit zijn bericht M
2. A vercijfert dit met zijn private sleutel
3. Verstuurt dit + de gewone boodschap
4. B ontcijfert dit door de publieke sleutel van A te gebruiken en krijgt H
   1. Bewijs dat het bericht van A komt
5. B berekent hash van gewone boodschap en verkrijgt H2
6. Als H en H2 gelijk zijn, is de boodschap ongewijzigt



**5.2 Hybride cryptografie**

Hybride cryptografie is een combinatie van symmetrische en asymmetrische encryptie.





**H6 Certificaten**

Het probleem van werken met publieke sleutels is dat er geen manier is om te garanderen dat een bepaalde publieke sleutel bij een bepaalde persoon hoort.

A wil een bericht naar B sturen. Als er een MITM-aanval door C gebeurt denkt A dat hij communiceert met B terwijl hij eigenlijk met C communiceert.

Een certificaat:

* Verbindt een publieke sleutel aan een individu of organisatie
* Is een elektronische identiteitskaart
* Uitgereikt door een **Certification Authority (CA)**
  + Trusted Third party
* Verschillende soorten
  + Persoonsgebonden certificaat
    - Gekoppeld aan 1 persoon
  + Rolgebonden certificaat
    - Rol die een persoon vervult binnen een organisatie
  + Toepassingsgebonden certificaat
    - Applicaties die zelf gegevens veilig versturen gaan gegevens ondertekenen
  + SSL-servercertificaat
    - Beveiligde verbinding tussen client en server
    - Identificatie IP-adres + domeinnaam van de server
  + Signing certificaat
    - Ondertekening van de software: code afkomstig van de ondertekenaar

PKI biedt faciliteiten en maatregelen voor het beheer van publieke sleutels

**6.1 PKI – Public Key Infrastructure**

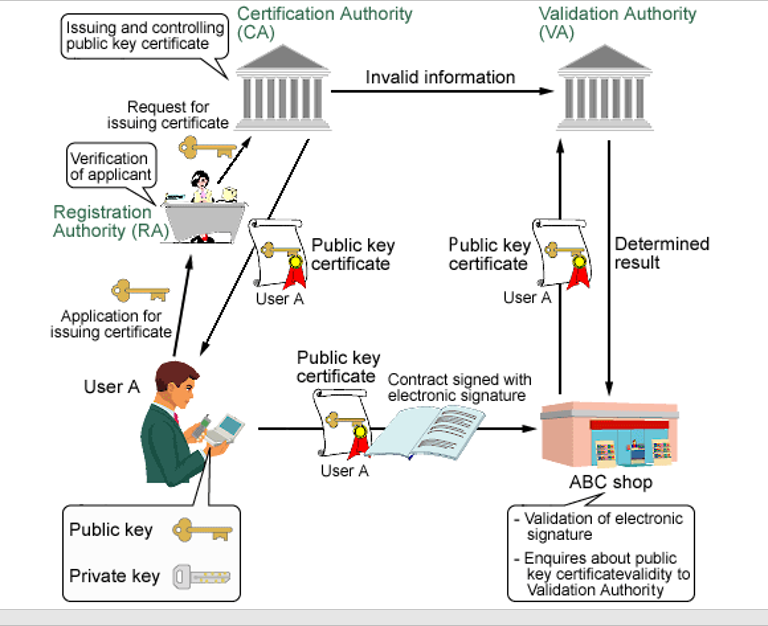
**Wat is PKI?**

* Technologie
  + Public key Technologie gebruikt voor encryptie samen met digitale certificaten
* Infrastructure
  + Certification Authorities
  + Security management
* Services
  + Vertrouwelijkheid
  + Authenticiteit
  + Integriteit
  + Onweerlegbaarheid

Een PKI-Certificaat heeft een public key en een aantal extra gegevens. Het certificaat is ondertekent door de CA die hiermee garandeert dat de public key in het certificaat bij een bepaalde persoon hoort. Als je de CA vertrouwt, vertrouw je het hele certificaat.

PKI Componenten:

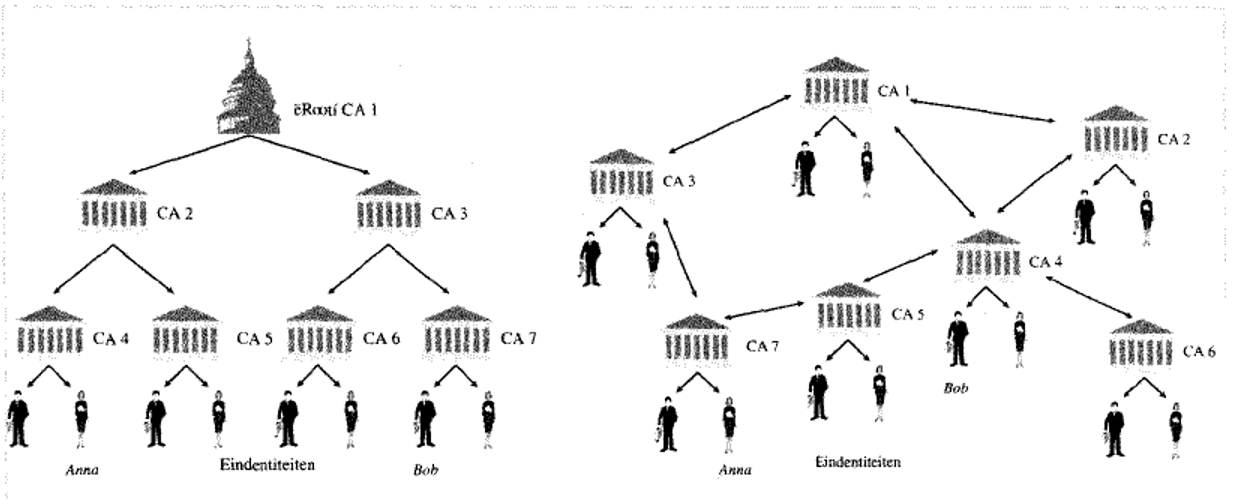
* Certification Authority (CA) - bouwers
  + Generatie certificaten
  + Uitgave en distributie
  + Intrekken van certificaten
  + Beleid is vastgesteld in 2 documenten: CP en CPS
    - Laat de gebruiker bepalen hoeveel vertrouwen hij heeft in het certificaat
    - Vertrouwensklassen zijn gestandaardiseerd (IETF)
* Registration Authority (RA) – secretariaat
  + Registratie van gebruikers
  + Vaststellen identiteit van gebruikers
    - Proof of possession (POP)
* Validation Authority (VA) - controleurs
  + Verifiëren of certificaat al dan niet is ingetrokken of opgeschort
* Database
  + Opslag van certificaten
  + Opslag black list (CRL)
    - Certificate Revocation List
* Certificate Policies (CP)
  + Regels en normen die aan CA wordt gesteld
  + Beleidsniveau
  + Gestandarizeerd document
    - Contractuele bepalingen
    - Certificaat formaat
    - CRL formaat
* Certificate Practice Statement (CPS)
  + De door de CA gevolgde procedures om aan die regels en normen te voldoen
  + Hoe ze de CP concreet invullen
* Eindentiteiten



**PKI-architectuur**

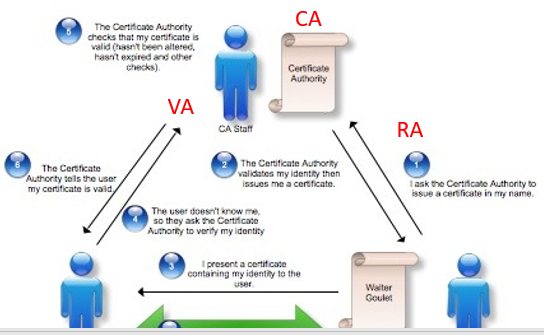
Twee soorten architecturen

* Hierarchisch
  + Veel toegepast
  + Iedereen kent public key van root CA
    - Root CA signeert CA’s van onderliggende levels, die dan signeren voor daaronder en uiteindelijk het certificaat uitreiken op het laatste leven
    - Wordt gebruikt om validiteit te controleren
  + Voordelen
    - Duidelijkheid in opbouw vertrouwens-relatie
  + Nadeel
    - Kwetsbaar, als het vertrouwen tussen twee CA’s wegvalt, valt de hele structuur in elkaar
* Mesh
  + Alle CA’s hebben gelijkwaardige relaties met elkaar
  + Geen root of hierarchie
  + Iedereen kent public key van lokale CA waar hij bij aangesloten is
  + Verificatie kan langs verschillende paden
    - Cross-certificering
      * Op basis van CPS (Certificate Practice Statement)
      * Erkenning van 2 CA’s in een verschillende omgeving
  + Voordelen
    - Flexibiliteit
      * Uitvallen van CA is geen probleem
  + Nadeel
    - Niet duidelijk hoe de vertrouwensrelatie juist is



**Certificaat aanvragen**

* A geeft zijn publieke sleutel + persoonlijke gegevens aan RA
* Certificaat voor A wordt aangevraagd bij CA
* CA keurt dit goed, maakt het certificaat aan en ondertekent deze
* Certificaat wordt naar A gestuurd en toegevoegd aan de database (directory service)
* B zoekt certificaat op in DB
* B controleert of certificaat met publieke sleutel van de CA om te kijken of het echt van de CA komt
* B is zeker dat het van A is en communiceert met A



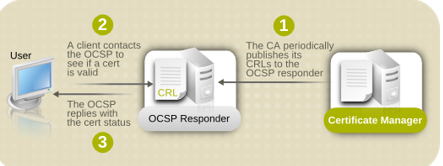
**Intrekken van certificaat**

* Wanneer?
  + Vermoeden dan certificaat niet meer integer is
  + Private sleutel is gecompromitteerd
  + Geldigheidsduur is verlopen
* Gevolg
  + Certificaat uit de database (public directory) halen/revoken
  + Wordt in de Certificate Revocation List (CLR:Black list) gestoken
  + Intrekking wordt gepubliceerd door CA
  + Structuur volgens standaard x.509

**Online Certificate Status Protocol - OCSP**

Het OCSP protocol wordt gebruikt om de geldigheid van certificaten na te gaan.

* Online realtime controle van certificaat
* Statusinfo aan de OCSP server vragen die in rechtstreeks contact staat met CA
  + Naam CA, serienummer, versienummer
* Verwerking door OCSP met mogelijke antwoorden
  + Goed – niet op CRL
  + Revoked – op CRL
  + Unknown – OCSP kent CA niet



**Key escrow**

Key escrow is een concept waarbij je een derde partij toelaat om je private sleutels te beheren. De private keys worden dan beheerd door de CA en TPP

In de VS wordt crypto gezien als een militair aspect

* De uitvoer van sterkre crypto-systemen is verboden
* Zwakke crypto mag wel, met korte sleutels
* Sterke crypto is soms toegestaan, maar dan moet de gebruiker zijn private key laten beheren door de CA (TTP) = key escrow
  + Big brother toestanden
  + Zo kan de VS de communicatie bekijken als ze een gerechtelijk bevel hebben
    - Nodig in de strijd tegen terrorisme

**PKI beveiligde systemen**

* Secure e-mail
* Code signing
* VPN’s
* Secure single sign-on

**H7 Web Secutiry**

Het internet is gevaarlijk

* Internet werkt in twee richtingen
  + Server -> client
  + Client -> server
* Makkelijk om tools te gebruiken en server op te zetten maar onderliggende technologie is complex
  + Mogelijk veel veiligheidsrisico’s
* Webserver kan toegangsportaal zijn tot bedrijfsnetwerk
* Veel gebruikers zijn zich niet bewust van de veiligheidsrisico’s

Er zijn verschillende soorten bedreigingen

* Integriteit
* Vertrouwelijkheid
* Denial of Service
* Authenticatie

Het webverkeer beveiligen

* IPSec
  + In de netwerklaag
  + Transparant voor eindgebruikers en toepassingen
  + Universele oplossing
  + Filtermogelijkheid: alleen verkeer dat beveiligd moet worden heeft de IPSec overhead
* SSL/TLS
  + In de transportlaag
  + Beveiliging bovenop TCP
  + De meeste browsers en webservers hebben SSL geimplementeerd
  + Applicatie moet zo ontworpen zijn om SSL/TLS te gebruiken
* Toepassingsgebonden beveiligingsservices
  + In de applicatielaag
  + Service kan op maat van de toepassing gemaakt worden
  + Bv. s/MIME – PGP

De methoden zijn bijna dezelfde qua services en mechanismen maar verschillen in toepassingsbereik en plaats in het OSI-model.

**7.1 Electronic mail**

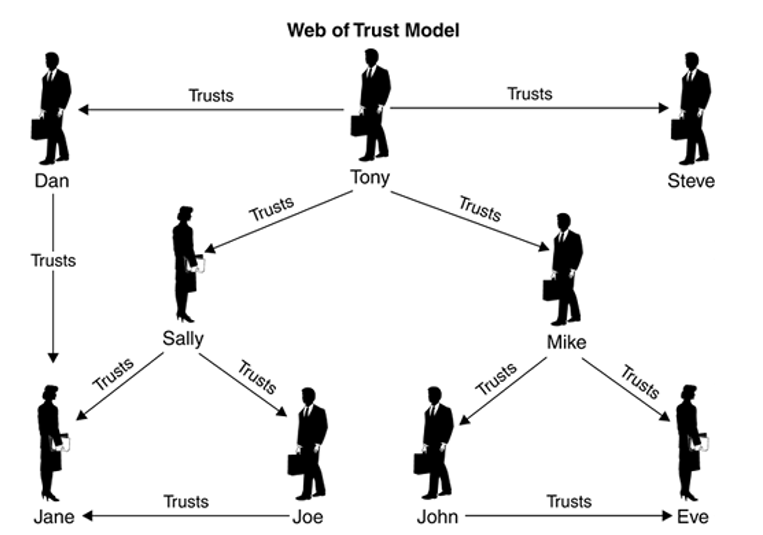
Er worden twee systemen gebruikt om email te beveiligen. Beide gebruiken het principe van hybrid crypto om e-mails te versturen. Grootste verschil is het gebruik van certificaten.

**Secure Multipurpose Internet Mail Extension (s/MIME)**

* Gebruikt PKI x.509v3 certificaten
  + Uitgereikt door een CA
* Meestal gebruikt in web-based mailingsystemen

**Pretty Good Privacy**

* Gebruikt zelf gemaakte certificaten die vertrouwd worden door een **Web of trust (WoT)**
  + Een ge-decentraliseerde manier om certificaten te vertrouwen => niet afhankelijk van CA
  + “Als x-aantal van uw vrienden een certificaat vertrouwen, dan vertrouw jij dit ook” (opbouw WoT)
    - Het certificaat vertrouwen = signing met private key
  + Wordt door vele IT’ers gebruikt



**7.2 SSL/TSL**

**SSL/TSL**

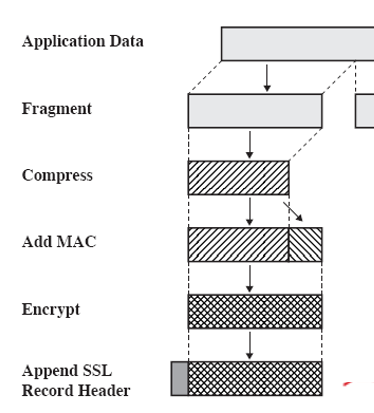
* Secure Socket Layer
* Ontwikkeld door Netscape
* Werkt op de transportlaag
* SSL beveiligt verbinding tussen webbrowsers en webservers
  + HTTPS = http + ssl
* Heel flexibel
* Gebruikt TCP om betrouwbare end-to-end service te leveren

**7.2.1 SSL-Architectuur**

Bestaat uit twee lagen van protocollen

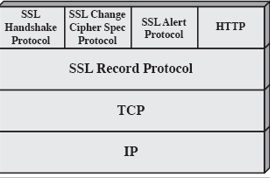
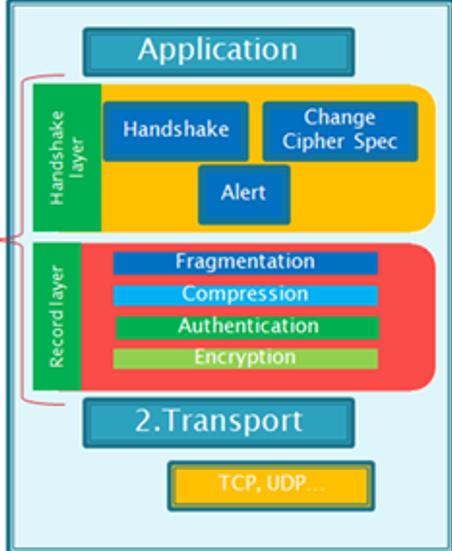
**Laag 1**

* SSL Record Protocol
  + Levert elementaire beveiligingsservices aan hogere lagen
    - Vertrouwelijkheid
    - berichtintegriteit
  + Beveiligde data transfer
  + Gebruikt gegevens en methoden die in de handshake afgesproken zijn
    - Soort compressie, encryptie, keys, hashing, …
  + Data afkomstig van de datalaag wordt:
    - Gefragmenteerd
    - Gecomprimeert
    - Een MAC toegevoegd
    - Versleutelt
    - Een header toegevoegd en naar de TCP laag gestuurd



**Laag 2**

* Wordt voor het beheer van SSL uitwisseling gebruikt
  + SSL Handshake protocol => Zie 7.2.3
  + SSL Change Cipher Spec protocol
    - Geeft aan dat de afspraken in orde zijn
  + SSL Alert Protocol
    - Iets is misgegaan in de communicatie => alert
    - Onderhandelingen zijn afgebroken => alert



**7.2.2 Handshake protocol**

Bedoeling van het handshakeprotocol:

* Authenticatie
* Cryptografische parameters afspreken zoals welke ecryptiealgoritmes, sleutels, …

Bestaat uit 4 fasen:

**Fase 1**

* Client stuurt “Client Hello” met de volgende parameters
  + SSL versie van de client
  + Sessie-ID
  + Lijst met vercijferingsalgoritmes die de client kent
  + Lijst met compressiealgoritmes die de client kent (zip + hash-functies)
  + Random getal van de client
* Server stuurt “Server Hello” et de volgende parameters
  + SSL versie van de server
  + Sessie-ID
  + Gekozen encryptie algoritme
  + Gekozen compressie algoritme (en gekozen hash functie)
  + Random getal van de server

Nu weet de server dat ze een sessie kunnen starten en met welke algoritmes ze aan werken.

**Fase 2**

De server’s authenticatie stuurt:

* Zijn sleutelcertificaat (waar zijn publieke sleutel inzit)
  + Een “Server Key Exchange” indien de server geen certificaat heeft
    - Hiermee wordt dan afgesproken hoe een master key kan gemaakt worden op een andere manier voor encryptie
* Een “Certificate request”
  + Vraagt de client zijn certificaat
* Server Hello Done
  + Hello bericht gedeelte is afgelopen voor de server

Tussen fase 2 en 3 gaat de client nagaan of de server hello acceptabel is door het nagaan van het certificaat bij CA/VA via OCSP

**Fase 3**

De clients authenticatie stuurt:

* Zijn sleutelcertificaat (waar zijn publieke sleutel inzit)
  + Een “no certificate” -alert indien de client geen certificaat heeft
  + Server neemt beslissing of communicatie verder kan
* “Certificate verification” (proof of possession)
  + Maakt een hash van alle berichten die tot nu toe werden uitgewisseld in de handshake versleuteld met de private key van de client (= signature)
    - De server heeft deze berichten ook en berekent dus zijn eigen hash en decrypt de hash van de client. Als die overeen komen heeft de client bewezen dat hij de private key heeft die met de publieke key overeenkomt
  + Server moet niet de CA checken om een certificaat na te kijken ( => bespaart veel tijd!)
* “Client key exchange”
  + Client maakt een 48 byte “pre-master secret” (interim sleutel)
  + Versleutelt dit met de publieke sleutel van de server

Gevolg:

* Client en server hebben beiden de premaster secret
  + Beiden berekenen uit de premaster secret de master secret
    - Premaster secret + random getallen die in het begin uitgewisseld waren
  + Berekenen uit de master secret o.a. de encryptie key en MAC key
    - Encryptie key is bv. AES secret
    - MAC = Message Authentication Code (beveiligde hash functie)

De random getallen voorkomen een **replay attack**

De aanvaller houdt alle berichten bij die gecommuniceerd worden met de server, en stuurt die op een ander tijdstip terug naar de server.

Omdat er random waarden in het spel zitten, zal de server een andere random waarde spelen als de aanvaller op “play” duwt en dus zal de replay attack falen.

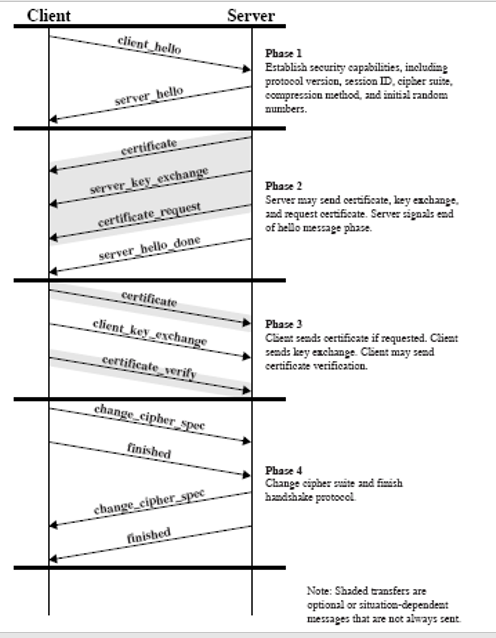
**Fase 4**

De client stuurt:

* Een “change cipher spec”
  + Vanaf nu beveiligde berichten gebruiken
* Een “finish”-bericht

De server stuurt:

* Een “cange cipher spec”
* Een “finish”-berich



**7.3 VPN**

Een virtual private network is een afgesloten netwerk dat gebruik maakt van het internet.

* Private => Beveiligd door allerlei mechanismen
* Virtual => Is geen directe lijn, maar over het internet door gebruik te maken van een tunneling techniek
* Voorbeelden
  + PPTP
  + IPSec
  + SL/TLS

**7.3.1 IPSec**

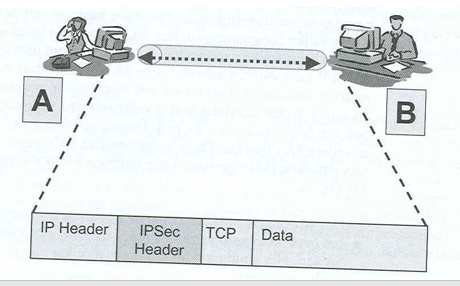
IPSec

* Internet Protocol Secure
* Standaard voor het beveiligen van het IP-protocol d.m.v. encryptie en/of authenticatie op alle IP-paketten.
* Werkt op de netwerklaag
* Is een framework van open standards -> open standaarden kiezen
  + Welke IPSec protocol?
    - AH – Authentication Header
    - ESP – Encapsulated Security Payload
    - AH+ESP
  + Welke symmetrische encruptie
  + Welke authenticatie
  + Hoe afspreken shared key?

**7.3.2 VPN Modes**

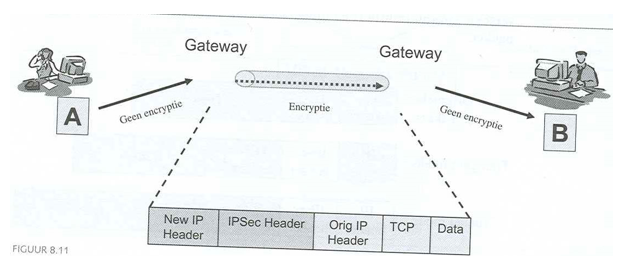
**Transport mode**

* Behouden van de originele IP header
  + Originele pakket wordt niet geïncapsuleerd in een ander IP-pakket, maar behoudt zijn eigen header
* End-to-end security
* End computers doen de security processing
  + Niet belangrijk om IP-adressen te verbergen voor iemand



**Tunnel mode**

* Alles verbergen, nieuwe IP-header geven
  + Originele pakket inkapselen in een ander IP-pakket met een nieuwe IP-header
  + Originele IP-adressen worden dus verborgen
* Als één van de eindgebruikers een gateway (router, firewall) is, dan moet tunneling mode gebruikt worden
* Security voor
  + Network-to-network
  + Host-to-network
  + Host-to-host



**H8 Systeembeveiliging**

**8.1 Firewalls**

Een firewall is elk apparaat dat als toegangscontrole-mechanisme voor een bepaald netwerk wordt gebruikt.

* Gebruikt regels om toegang te verlenen
  + Poortnummbers
  + Doeladres
  + Bronadres
  + …

Een firewall is een fundament waarop informatiebeveiligsbeleid van de meeste organisaties op rust, maar is niet volledig waterdicht.

**8.1.2 Soorten firewalls**

**L4 Firewall**

* Firewalls die pakketten filteren (routers, cisco, …)
* Variabelen:
  + Bronadres
  + Doeladres
  + Protocol
  + Poortnummer
* Voordeel
  + Gemakkelijk te implementeren, meestal heb je een router
* Nadeel
  + Niet voorbereid op DoS-aanval
  + Kunnen geen sessietoestanden aan
  + Grote ACL’s kunnen netwerkprestatie verminderen bij grote belasting

**L5 Firewall**

* Stateful packet-filter firewalls (Checkpoint, PIX, ..)
* Minimumvereiste voor een firewall binnen een netwerk
  + Doet pakketfiltering + houdt sessies en verbindingen bij in tabellen
    - Kijkt naar sockets
  + Kijkt niet alleen naar pakketjes maar ook naar de context
  + Speciaal ontworpen tegen DoS aanvallen
  + Kunnen poorten boven 1024 (TCP-connecties) standaard gesloten laten en openen wanneer het nodig is.

**L7 Firewall**

* Firewalls die als proxy fungeren
* Inspecteert het verkeer:
  + Op netwerk- en sessieniveau
  + Toepassingniveau
  + Deep packet inspection
    - HTTP pakket => doorgeven aan HTTP-proxyprocedure
    - FTP pakker => doorgeven aan FTP-proxyprocedure
* Voordeel
  + Veiliger dan de L3 of L4, want het begrijpt het toepassingprotocol
* Nadeel
  + Trager
  + Nieuw protocol => geen procedure

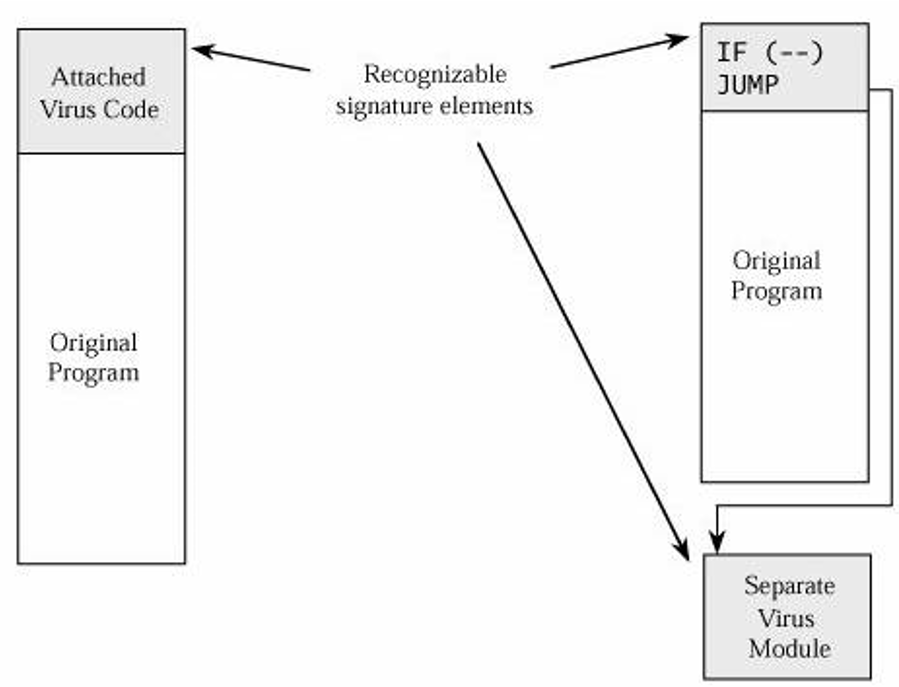
**8.1.3 Tekortkomingen**

* Hoe hoger de beveiliging, hoe minder functionaliteit
* Geven een vals gevoel van veiligheid
  + Zorg voor een layered defence
* Kijk de firewall logs na
  + Iemand aan de deur aan het rammelen?

**8.2 Virussen en wormen**

8.2.1 Virussen

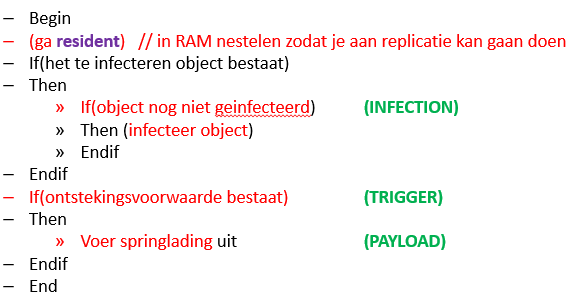
* is een programma dat zich kopieert door andere programma’s te infecteren zodat zij ook een versie van het virus bevatten.
* Infectie
  + Virussen hechten zich aan 1 of meerdere programma’s op het doelsysteem. Als het programma opstart activeert het virus zich
* Grootste kenmerk is replicatie
* Meeste virussen volgen volgende route
  + User start legitiem programma (user interaction)
  + Virus, genesteld in de instructiereeks van dat programma, wordt uitgevoerd ipv het originele programma
  + Viruscode wordt beëindigt en controle wordt teruggegeven aan het legitieme programma



**8.2.2 Wormen**

* Worm doet ook aan replicatie, maar manier van hechting is anders van een virus
  + Virus hecht zich aan legitiem programma
  + Worm verspreidt zich via netwerken/systemen zonder zich ergens aan te hechten
    - Besmet zijn omgeving ipv specifieke objecten
    - Meestal geen user interaction nodig
      * Misbruik van lekken in software
      * Snellere verspreiding dan virus

**8.2.3 Hoe werken virussen**



**8.2.4 Verschillende soorten virussen**

**Boot Sector Infectors (BSI)**

De boot sector bevat info over de bestandsstructuur van een schijf. Wat zich waar bevind en extra info om te booten.

Er zijn twee soorten Boot sectors:

* Master Boot Record
  + Eerste sector van een storage device dat gepartitioneerd is. Bevat essentiële info over de harde schijf.
* Volume Boot Record (Partition Boot Sector)
  + De eerste sector van een storage device

Een boot sector infector is een virus op een diskette of usb die de normale bootrecord vervangt met zijn eigen code. Als een HD geïnfecteerd is zullen alle USB’s die via de PC gebruikt worden geïnfecteerd worden.

De meeste bootsector virussen kopiëren de originele bootsectorcode naar een andere plaats.

**Memetische virussen**

* Virussen zonder code, netpvirussen, virus van geest, metavirus, hoax
* In de vorm van een kettingbrief

Andere soorten:

* Virussen die bestanden infecteren
* Multivirussen
* Macrovirussen
* Scriptvirussen
* (wormen)

**8.2.5 Virus kenmerken**

* Stealth
  + Proberen aanwezigheid te verbergen om de kans op ongemerkte verspreiding te vergroten
  + Verdachte springladingen (payload) worden vermeden
    - Vertraging activiatie van de trigger
    - Onregelmatige tijdsippen
  + Bewaren originele code voor als het OS bepaalde informatie opvraagt
* Polymorfie
  + Om moeilijker detecteerbaar te zijn
  + Hecht iedere keer een andere virusvorm aan het object
    - Ge-evolueerde versie
    - Veranderen van instructievolgorde, toevoegen lege bytes, …
  + Meeste virusscanners moeten een soort patroon herkennen en niet gewoon identiek dezelfde code

**8.3 Trojans and Rootkits**

**8.3.1 Trojans**

Het is heel moeilijk om trojans te onderscheiden van virussen en wormen, maar het belangrijkste verschil is:

* Virussen en wormen zijn replicerende progs
* Trojans zijn statische code
  + Altijd geassocieerd met misleiding

Het is moeilijk om op een echte trojan te scannen, want definitie van trojan houdt in dat er een verschil is wat het programma doet, en wat de gebruiker verwacht.

**8.3.1.1 RAT – Root/Remote Acces Trojan**

* Trojan om toegang te krijgen zonder medeweten van de victim
  + Een hacker in is controle van het systeem
* Bypass van de authenticatie
* Legitiem gebruik:
  + Teamviewer

**Application-level backdoor trojan**

* Op victim wordt een server app geïnstalleerd
* De hacker geeft een client om op die server te connecteren

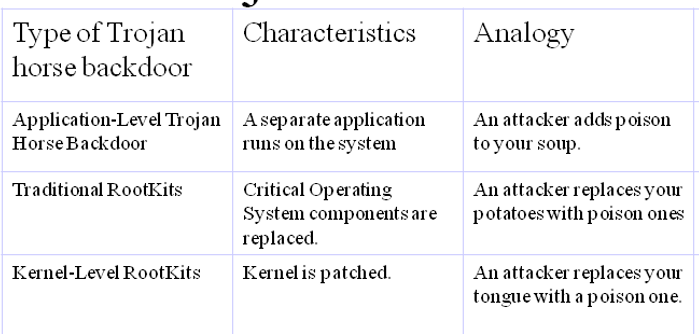
**8.3.2 Rootkits**

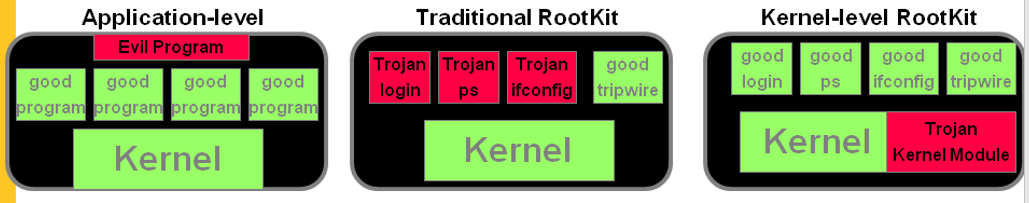
**Traditionele rootkits/User mode rootkit**

* Zoals een RAT, maar sneakier
* Gaat veel gebruikte en belangrijke executables aanpassen/vervangen om zo een backdoor te maken en full access te geven.
* Gaat zich diep in het systeem verbergen

**Kernel based rootkit**

* “The sneakiest of the sneaky”
* Vervangen/aanpassen van de kernel
  + Opvangen en veranderen van system calls
  + Zeer moeilijk te detecteren
  + Zo een backdoor maken voor full system acces





Watch: <https://www.youtube.com/watch?v=H23qyUbKuHM,http://www.youtube.com/watch?v=mObYxn26RmM>

**8.4 Sniffers – Protocol analyzers**

* Sniffers voor de hackers => data afluisteren voor slechte doeleinden
* Protocol analyzers voor de crackers => netwerkverkeer bestuderen
* apparaten die netwerkpakketten onderscheppen
* Analyse van het netwerkverkeer en signaleren van potentiele gevaren
* Protocolanalyzers
  + Standard Ethernet, TCP/IP, NetBios/…
* Network Intrusion Detection System (NIDS) gebruikt ook sniffer
  + NIC in promiscuous mode (overspel-modus)
    - Normaal pakt netwerkkaart alleen pakketten op die voor dat systeem bedoeld zijn, de rest negeert hij.
* Zijn passief: alleen afluisteren
  + Moeilijk op te sporen!

**8.4.1 Risico’s sniffers**

Sniffers kunnen

* Accountnamen en wachtwoorden afluisteren
  + Goed geplaatste sniffer kan duizenden wachtwoorden onderscheppen
* Vertrouwelijk of prive-info afluisteren
* Gebruikt worden om de beveiliging van naburige netwerken te ondermijnen of op een hoger niveau toegang te krijgen
* Kunnen alle pakketten opvangen, maar aanvallen moet kieskeurig zijn
  + Duizenden pakketten gaan over een netwerk
  + Aanmeldingsinfo is waardevolste voor aanvallers
  + Aanmeldingsinfo voor versleutelde verbindingen is niet kwetsbaar
    - SSL en SSH

**Waarschijnlijkste plek om een sniffer te vinden**

* Overal in principe
* Strategische punten zijn favorieten
  + Vlakbij machine met veel wachtwoorden (servers)
  + Vlakbij default gateway

**8.4.2 Aanvallen van sniffers afslaan**

Aanvallen van sniffers afslaan is niet gemakkelijk. Ze zijn moeilijk te detecteren omdat het passieve programma’s zijn die geen auditspoor achterlaten en geen netwerkresources gebruiken.

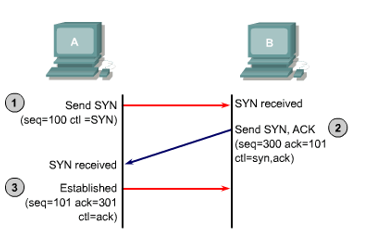
* Sniffers detecteren en elimineren
  + Uitkijken naar netwerkkaarten die in overspel-modus staan
  + Berekenen van uw systeem/database met een controlegatal
  + Gespecialiseerde programma’s
  + Programma’s die veranderingen in topologie waarnemen: Bv. extra PC wordt toegevoegd
* Gegevens tegen sniffers beschermen - data protection
  + Segmentatie van het netwerk
    - Sniffers kunnen alleen pakketten detecteren die hun passeren => netwerk slim opsplitsen in compartimenten.
    - Switches/routers en bridges maken het een sniffer moeilijk
  + Versleutelde sessies
    - Encrypteer de gegevens in het pakket, zodat de sniffers het niet kunnen begrijpen.

**8.5 (TCP) Spoofing**

8.5.1 TCP/IP

Een TCP/IP Connectie

* Is volledig gedefinieerd door 4 parameters
  + Source socket (source ip en poort)
  + Destination socket (destination IP-adres en poort)
* Data wordt in IP-Pakketjes gestuurd
* IP is een wrapper rond TCP (UDP)-data
* Alle pakketjes hebben een header
  + IP header
    - IP source-destination
    - Wrapper type
  + TCP header
    - Poort source-destination
    - SEQUENCE/ACKNOWLEDGEMENT nummer
* Heeft een 3-way handshake om een connectie tot stand te brengen
  + Berust op sequence getallen die gestuurd worden tussen client en de server



**8.5.2 Spoofing**

(TCP Spoofing)

* Geavenceerde techniek om de ene machine bij de andere to authenticeren door pakketten, die afkomstig zijn van een vertrouwd bronadres, te vervalsen

**Authenticatie**

Je moet je bijna overal authenticeren

* Bv. Id en wachtwoord
* Email/ftp/websites/…

Een oud systeem is **RHOSTS (remote hosts)**

* UNIX-systeem die een vetrouwensrelatie tussen computers maakt
* \*.hosts bestanden
  + Identificering op basis van IP-adres
  + Bepaalt wie bepaalde services mogen gebruiken
  + Geen authenticering meer nodig als ze op de lijst staan
  + Wordt bijna niet meer gebruikt => vervangen door SSH

Vertrouwen en authenticatie zijn omgekeerd evenredig

Flawed TCP/IP model

* Authenticatie op basis van IP-bronadres is zeer makkelijk te misbruiken
* IP header kan gemakkelijk nagemaakt worden met een bogus of een trusted source IP-adres.
* Source routing (strict-loose)
  + Optie in IP-protocol
  + Antwoord wordt volgens een bepaalde route teruggestuurd
    - Hacker dan het antwoord naar zijn eigen late routen
  + TCP/IP kijkt alleen maar naar source in de header

**8.5.3 Spoofing mechanisme**

Alleen het bronadres vervalsen is niet genoeg. De hacker moet ook de volgordenummerdialoog van TCP in stand houden.

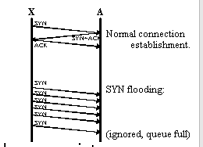
Er zijn twee manieren van spoofing

**Non-blind spoofing**

* De hacker ziet het server antwoord en dus zijn start sequence number
* Pakketje komt langs een sniffer of dergelijke van de hacker (hacker binnen het netwerk)
* IP-Source routing
  + Pakketje komt voorbij de hacker

**Blind spoofing**

* Hacker ziet dit niet, dus moet er een beredeneerde gok naar dat sequence nummer gedaan worden
* Antwoord wordt teruggestuurd naar de client (die je wilde spoofen)
  + Als die echt bestaat, gaat hij een antwoord naar de server sturen dat hij geen communicatie gestart had (RESET)
  + Server stopt met alle communicatie
* Hacker moet de client doen inslapen, zorgen dat de client niet actief is
  + Wachten tot hij uitgelogd is
  + SYN-flooding (DDOS)
    - Heel veel aanvragen tot connectie sturen naar client
    - De client stuurt een antwoord terug en wacht op een antwoord
    - Er komt geen achtwoord
    - Reset duurt even => maakt de aanval zo zwaar
    - Veel aanvragen zorgen ervoor dat de client de inkomende berichten van de server niet kan verwerken want hij is continu bezig met die andere verzoeken en smijt het bericht zelfs web bij overflow.
* Hacker moet het sequenceNr raden
  + Paar keer proefdraaien
    - Contact maken met server onder echte adres
    - Sequence Nr van de server krijgen
    - Connectie verbreken en analyse doen van die sequenceNr
      * Patronen ontdekken
      * Gok doen naar gebruikte random generator
* Als dat gelukt is maakt de hacker een backdoor
  + Hacker wil niet iedere keer opnieuw de server spoofen
  + Past de instellingen van de server aan zodat hij een trusted user is
* Hacker verbreekt de verbinding en logt gewoon in



**8.5.4 Ingredienten geslaagde spoofing**

1. Hacker moet doelwitten identificeren
2. Moet de source waarvoor hij zich wil uitgeven verdoven (SYN-flood)
3. Vervalsen van het adres van de source
4. Connectie maken met de server als die source
   1. Non-blind: via source routing/ARP Poisoning antwoord berichtjes onderscheppen voor de Seq/Ack nummers
   2. Blind: vooraf proefdraaien, en daarna raden van het volgordenummer waar de server om vraagt
5. Eenmaal binnen een backdoor maken.

**8.5.5 IP-spoofing voorkomen**

* Geen IP sourcing toelaten
* Geen proxy-authenticatie van een firewall doen d.m.v. IP-bronadres
* Logprocedures maken en netwerkverkeer in de gaten houden
* Ingress filtering
  + Blokkeren van pakketten die van buiten het netwerk komen, maar een intern IP-adres hebben als source
* Egress filtering
  + Blokkeren van pakketten die van interne LAN komen met als source-IP een extern adres hebben
    - Hacker heeft waarschijnlijk de computer overgenomen

**8.5.6 ARP Spoofing/Poisoning**

**ARP**

* Address Resolution Protocol
  + Mappen van IP->MAC
* ARP request
  + Host met IP x, wat is uw MAC?
* ARP response
  + Ik met IP x heb MAC y

Vervolgens wordt de MAC in de ARP tabel gezet.

**ARP Spoofing/Poisoning**

* Hacker stuurt ARP response naar de target
  + Reponse bevat: Ik met IP x bevat MAC y
  + Binnenkomde ARPS worden in de ARP cache gezet (zonder controle)
  + Als de target nu een bericht wil sturen naar het vertrouwde IP zal het bericht bij de hacker aankomen want het target zoekt het MAC adres voor een bepaalde IP.
* Meest gebruikte target voor APR spoofing: Default Gateway van de LAN (router)
  + Hacker stuurt naar alle devices in LAN ARP Response
  + Ieder device stuurt zijn data bestemd voor de gateway naar device van de hacker
    - Hacker kan nu kiezen
      * Sniffing
        + Alleen de pakketjes bekijken en ze dan doorsturen naar de echte gateway
      * Man in the middle attack
        + Pakketjes veranderen en dan verdersturen naar de gateway
      * DoS-attack (Denial of Service)
        + Pakketjes allemaal tegenhouden
        + Niemand kan buiten de LAN communiceren
* Problemen
  + Cache wordt regelmatig vernieuwd op devices => hacker moet regelmatig ARP responses sturen om dit in stand te houden
  + Werkt meestal enkel binnen dezelfde LAN
* Voorkomen/beveiligen tegen ARP spoofing
  + Statisch maken van de linking
    - Niet automatisch cache updaten bij binnenkomende ARP response
  + ARPWATCH
    - Houdt wijzigingen in IP/Ethernet mapping in de gaten
    - Stuurt email als er een wijziging is gebeurd
  + DHCP Snooping
    - DHCP geeft IP-addressen aan PC’s (MAC’s)

**8.5.7 DNS-Spoofing**

* Wijziging van DNS-tabellen die namen (URLS) met IP-adressen linkt
* Redelijk makkelijk te ontdekken
  + Bij verdachte DNS-sever pollen naar andere DNS servers en kijken of die hetzelfde resultaat geven
    - Geeft een probleem als de server allang gecompromitteerd is want DNS servers wisselen onderling hun tabellen uit.
* Hacker moet zowel voorwaartse als achterwaartse tabellen wijzigen

**8.6 Denial-of-Service aanvallen**

* Aanval die tot gevolg heeft dat een bepaalde service verloren gaat of niet meer kan functioneren
  + Gevolgen voor netwerkprestaties, integriteit van gegevens

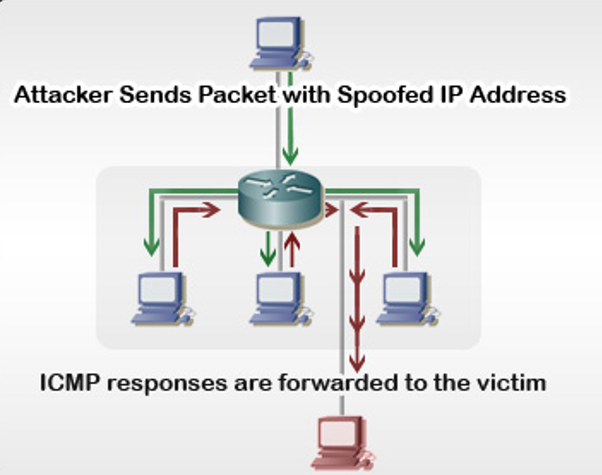
DoS aanvallen komen meer en meer voor. Het is een manier om aan te tonen aan ontwikkelaars dat er lekken zijn in hun beveiliging. Soms is een DoS niet altijd het gevolg van een aanval. (9/11 – veel mensen die de site van CNN wilde bekijken).

**8.6.1 Verschillende soorten DoS**

Een DoS is alleen mogelijk omdat software programmeerfouten bevat.

**Zoveel mogelijk bandbreedte in beslag nemen / aanval op netwerkbronnen**

* Netwerk kan maar een bepaalde hoeveel netwerkverkeer aan
* Als de bandbreedte wordt opgesoupeerd kunnen er geen nieuwe gegevens meer verzonden worden.
* Is een aanval van een active soort. Na de DoS aanval komt de bandbreedte weer ter beschikking
* Maakt gebruik van fouten in het protocol
  + Bandbreedte wordt opgebruikt door speciaal bewerkte netwerk-gegevens te verzenden – router loopt vast.
  + SMURF-attack
    - Met een gespoofed IP-adres broadcasten in een netwerk waardoor er allemaal ICMP (ping) berichtjes naar het slachtoffer gestuurd worden en zijn netwerk overbelast.



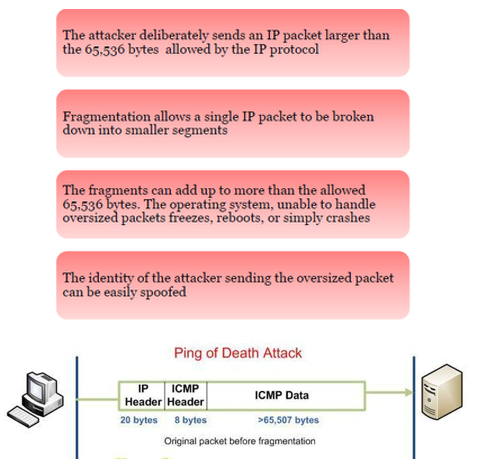
**Zoveel mogelijk bronnen verbruiken/ aanval op systeembronnen**

* Computersysteem heeft maar een beperkt aantal bronnen beschikbaar
  + Geheugen/opslag/processortime/…
* SYN-aanval / SYN-flood
  + Veel halfopen verbindingen met doelserver tot stand brengen
  + Misbruik van 3-way handshake in TCP/IP
* HTTP-aanval op webserver
  + Tegelijkertijd HTTP-verbindingen tot stand brengen met de webserver
  + Server kan het aantal niet aan, wordt trager en valt uiteindelijk uit
    - Door tijd dat het duurt om één aanvraag af te werken (complexe inforatie opvragen, query)
    - Processortijd van een aanval laten toenemen (complexe taak, sorteren)
* Php scripts
  + Oplossing hiertegen zijn human checks (captcha’s)

**Programma’s en systemen laten chrashen**

* Ping of Death
  + Grote ICMP verzoeken verzenden
  + Het is illegaal om zo’n grote ping door te sturen
  + Gefragmenteerd
  + Komt aan bij bestemming, reassembling van de pakketten tot één grote ping
  + Bufferoverflow bij de bestemming -> chrash
* Meestal gericht op de gateway => heeft meer effect dan een enkele machine lam leggen

Valt te voorkomen door er voor te zorgen dat goed beveiligde computers gateways mogen beheren.



**8.6.2 E-mailbommen**

* Geniepige, simpele aanval met soms grote gevolgen
* Heel veel berichten naar een postvak sturen
  + Vult de harde schijf van de gebruiker op, of die die van de mailserver
* Postvak vol => er kunnen geen nieuwe berichten meer gestuurd worden
  + Kan leiden tot gegevensverlies, hoger bandbreedte gebruik en hogere netwerkkosten

Valt te voorkomen door: mailfilters, uitsluitschema’s en het contacteren van de postmaster als je aangevallen wordt.

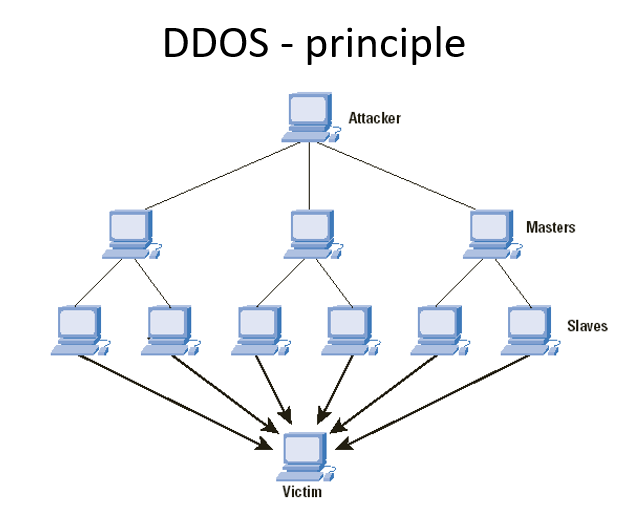
**8.6.3 Distributed DoS-aanvallen**

**DDoS-aanval**

* Functioneert op basis van **Master-Slave (zombie) principe**
* Netwerkverkeer afkomstig van een heleboel verschillende systemen tegelijkertijd
* Bot-net, Zombie-net, Master-Slaves

**Master-Slave (zombie) principe**

* Master is controlerende station
  + Aanvaller definieert daarop zijn doel en methode
  + Slaves zijn externe systemen waarop het aanvalsprogramma is geïnstalleerd
  + Master zegt tegen slaves om de aanval uit te voeren en/of de aanval te stoppen
    - Moeilijk om tegen te houden omdat het van zoveel verschillende system komt



Wat kan men tegen DDoS doen?

* Onnodige services en netwerkgegevens filteren uit je systeem
* Gebruik kracht verificatiemechanisme
* Controleer toegang tot externe systemen grondig
* Actualiseer systemen en software regelmatig (updaten)
* Load balancing

**8.7 Poortscanners**

* Scant een netwerk device voor openstaande poorten
  + Crackers: hoe en waar in te breken
  + Admins: hoe is de security? Zijn er lekker?
* Portscan
  + Kijkt welke poorten er openstaan op 1 host
* Portsweep
  + Kijkt of een specifieke poort openstaat bij verschillende hosts
* Scans geven meestal 1 van de drie mogelijke antwoorden
  + Positive – Open/accepted
  + Negative – Closed/denied/not listening
  + No reply – Filtered, Dropped, blocked
* Gebaseert op de 3 way handshake: Krijg je een ACK terug van de server op die poort?
  + TCP data is zeer gewaardeerde data voor hackers
* NMAP/NETSTATS commandos

**8.8 Vulnerabilityscanners**

**Vulnarabilityscanners**

* Tools om kwetsbare plekken op te sporen in uw beveiliging
* = poortscanner + kwetsbaarheidDatabase + rapport
* Ongeveer hetzelfde principe als een poort scanner
* Scanmechanisme
  + kan detecteren of een poort openstaat en bepalen welke service op welke poort draait en welke versie het is => is niet gemakkelijk en verschilt veel van kwaliteit
  + Bekende services op andere poorten aansluiten
  + Banner grabbing method
* Kwetsbaarheidsgegevens
  + Hebben een interne database met info over zwakke plekken die bepaalde versies van services kunnen hebben
  + Database is meestal gevuld met alle CVE’s known to date
    - Common Vulnerabilities and Exposures (CVE)
* Rapportmechanisme
  + Helder meedelen wat ze gevonden hebben, zodat er actie kan ondernomen worden.
    - Classificatie van problemen => kan leiden dat er false negatives en false positives tussen zitten (confusion matrix)
      * False positive => beveiligingslek gerapporteerd maar er is geen
      * False negative => Geen beveiligingslek gerapporteerd, maar er is er wel een.

**Confusion Matrix**

De kwaliteit van de classificatie hangt af van de false positives en false negatives.

* True positive
  + Goed => lek was degelijk een lek
* False positive
  + Slecht => Lek was geen lek
* True negative
  + Goed, geen lek is degelijk geen lek
* False negative
  + Heel slecht => Geen lek was eigenlijk lek

Nessus is één van de beste kwaliteitsscanners!

**8.9 IDS en IPS**

**Intrusion Prevention System (IPS)**

* Een soort van alarmsysteem
  + Ontdekt indringers, alarmeert en onderneemt zelf actie
  + Bv. het uitsluiten van een IP-adres of het blokkeren van een gebruiker

**Intrusion Detection System (IDS)**

* Is een soort van alarmsysteem
  + Ontdekt indringers en alarmeert, maar onderneemt zelf geen actie
* Mogelijkheden tot detectie
  + Logboeken van systeem, web, … bijhouden en analyseren/controleren.
  + Binaire integriteitscontroles (hashing)
  + Eventlogs bijhouden en speuren naar mislukte aanmeldingspogingen
  + Netwerkpackets checken op fingerprints van known attack vectors

Er zijn Netwerk-IDS en Host-IDS

**Netwerk IDS (SNORT)**

* Sniffers die alledaagse pakketten inspecteren en kijken naar bekende aanvalspatronen
  + Analyse van elk pakket over het bewaakte segment
* Passieve component
* Beperkingen
  + Gebaseerd op aanvalssignaturen van bekende aanvalstypen
    - Niew soort aanval => Geen detectie
    - Updaten van de signature-db
  + In een netwerk staan veel switches
    - Filtering van de pakketjes alleen over het juiste segement => signature valt misschien niet correct te identificeren
  + Heel veel pakketten maakt het moeilijk om alles te verwerken
  + Valt te omzeilen met pakketversleuteling

**Host IDS (Tripwire)**

* Agents die geïnstalleerd worden op de PC
* Analyseren systemlogs op events
  + Misluke aanmeldingen/schending toegangsrechten/…
  + Registreren van kernelberichten over mogelijke vijandige activiteit
  + Doorzoeken van systeem naar trojans en backdoors
  + Integriteitscontrole van files
* Nadelen
  + Organisatorisch => deployen naar verschillende machines, beheren, …
  + Alleen bekende problemen
  + Extra processortijd nodig
* Voordeel
  + Houdt de logs centraal bij en analyseert ze ook.
* Tripwire, Ossec, …

Daarin kan er nog een onderscheid gemaakt worden:

**Signature based**

Werken met een database.

**Anomalie based IDS**

* Gebaseerd op regelmatige gebruikerspatronen
  + Moeilijk om vast te stellen als er veel verschillende soorten van legaal gebruik zijn
  + IDS eerst in trainingsmode om het netwerk te leren kennen en het patroon vast te stellen
  + Dan in detectiemode
    - Nieuw gedrag => legaal of niet? Admin moet dit bekijken en laten weten of het goed of slecht is
    - Tijdsintensief gedrag want moet regelmatig bijgestuurd worden.
* Werkt regelmatig met “fuzzy rules”
  + Statistische berekeningen om met verschillende waardeverhoudingen rekening te houden.
  + Wat is normaal warm? Voor de één is dat 20 graden, voor de ander 25
    - Niet iedereen gebruikt dezeldfe waardeverhoudingen
  + Vage concepten die een bepaalde distributie hebben
  + Rekenen met die distributies om exacte regels te kunnen opstellen

**SNORT**

* NIDS
* Open source
  + Ondersteuning van developpers
* Zeer veel gebruikt
* Grote verzameling signaturen
* Volgens netwerk-model (NIDS)

**8.10 Log Management en SIEM**

**8.10.1 Logboek management**

**Waarom logs bijhouden?**

Logboeken kunnen problemen helpen opsporen, trends weergeven of anomalieën op het netwerk ontdekken. Ze moetsten bijgehouden worden om in een rechtzaak bijvoorbeeld te bewijzen welke dingen een indringer had gedaan.

* Hackers leren hoe ze logoboeken moet wijzigen om sporen uit te wissen
* Bestaan zelfs geautomatiseerde programma’s voor
* Omzeilen door log tools van derde partijen te gebruiken en de logs naar een beveiligde server te sturen

**Syslog**

* IETF Standaard
* Wordt door de meeste devices gebruikt
* Alle netwerk devices op het netwerk sturen hun systeemlogboeken naar een syslog server (syslogd)
  + Centrale opslag
  + Enige verkeer dat toegelaten is, is via UDP poort 514

**Logboek strategieën**

* Logboek en audittools van een derde partij te gebruiken in combinatie met de standaard logging-tools op uw systeem.
* Onafhankelijk logboekprogramma stelt zijn logs onafhankelijk samen
  + De twee logboeken kunnen vergeleken worden

**Logprogramma’s die analyses uitvoeren**

* Uit de analyse wordt een rapport gegeneerd
* syslog
* Door analyse worden trends duidelijker
* Er kan AI op toegepast worden om aanvalspatronen te herkennen
  + Het systeem wordt in principe een soort van intrusion detectie systeem
  + Start van de evolutie naar SIEM

**Samenvatting Log Management**

* Verzamelen van grote hoeveelheiden logs (log collection)
* Logs naar een centrale server sturen (centralized aggregation)
* Logs analyseren / searchen om rapport te genereren (analysis en reporting)
* Logs lang genoeg bijhouden (long term retention)

**8.10.2 SIEM**

SIEM

* Security Information and Event Management
* SIM
  + Security Information Management
  + Log management specifiek gebruikt voor security doeleinden
  + Long term storage/analyse/rapportering van logs (analyse na een aanval bv.)
* SEM
  + Security Event Manager
  + Real time monitoring (dashboards) van events, correlatie van events
  + Hoger niveau => logs worden gebruikt om aanvallen in real-time te kunnen detecteren (tijdens een aanval )
  + Correlatie van alle data (context) is nodig om real time analyse te kunnen doen
    - Context data is nodig => niet alleen syslogs maar iser accounts, firewall events, …
    - Alle data moet worden genormaliseerd zodat de analyse mogelijk is



**8.10.3 Big Data SIEM**

**IoT**

* Internet of things
* Meer en meer devices op het netwerk
* Ieder device genereert context informatie en logs
* Geeft een enorme berg aan data
* Big data tools – data mining tools nodig om al die data op een nuttige manier te kunnen verwerken.

**Big Data Tools**

* Security intelligence
  + Security events
* Operational intelligence
  + Nieuwe inzichten in het gebruik van de infrastructuur, optimalisatie van de servers, …
* Business intelligence
* Splunk
  + Beste big data engine op het moment

**8.11 Privacy**

**8.11.1 Soorten aanvallers**

Welke soorten aanvallers bestaan er?

* Script kiddies
  + Grootste groep, meestal onervaren en jong
  + Surfen op het web naar utilities, scripts .. die ervaren aanvallen gepost hebben
  + Meestal mislukt hun aanval
  + Zullen eerder gegevens op een kwadaardige manier beschadigen, dan welk ander type cracker ook
  + Weinig sympathie voor hen, zelfs binnen de cracker community
* Black Hats
  + The dark side
  + Zeer veel ervaring met computers
  + Zelden gepakt, doen grondig onderzoek en zeer gerichten aanvallen
  + Je hoort weinig over hun vaardigheden of capaciteiten (ninja’s)
* White Hats
  + The light side
  + Beveiligingsprofessionals die gaten opsporen en dichten
    - Pentesting
  + Bedrijven huren hen in om beveiliging te testen
  + Zijn meestal begonnen als black hat

**8.11.2 Bespionering**

Voor beveiliging van gegevens bestaan veel hulpmiddelen

* Digitale certificaten
* Pakketfilters
* Cryptologie
* Firewalls
* Antivirusprogrammas
* …

Maar toch moeten we onze persoonlijke privacy beschermen op het internet

**Twee soorten bespionering**

**Collectieve informatievergaring**

* Alleen info verzamelen over uw persoon, bezigheden, … zonder direct contact te maken

**Penetrerende informatievergaring**

* Contact maken, vertrouwen winnen, meer info verkrijgen

Alsook:

* Inlichtingendiensten kunnen zonder huiszoekingsbevel internetactiviteiten nagaan
  + Tenzij de gegevens zich op particuliere systemen bevinden (ISP)
* 8/10 websites volgen uw bewegingen
* Iedereen kan elektronische gluurder worden

**8.11.3 Browserbeveiliging**

3 belangrijke methoden via de browser

**Sniffen naar IP-adres en in de cache**

* Meeste webservers registreren uw gegevens
* Al die gegevens zorgen ervoor dat een cracker enorm veel info over je systeem te weten komt
  + Proxy gebruiken om dit tegen te gaan

**Cookies/tokens**

* Worden gebruikt om
  + info over u op te slaan als u door een webpagina bladert
  + activiteiten op te slaan
* Wordt gebruikt om login gegevens bij te houden zodat je niet iedere keer moet authenticeren
  + Authentication cookie
* Is zoals een stempel op uw hand voor een fuif
  + Zolang de stempel erop staat mag je op een fuif rondlopen, iets drinken, terug naar buiten en binnen gaan zonder opnieuw te moeten betalen.

**Banneradvertenties en webbugs**

* Banneradvertentie
  + Doet zich voor als reclame voor een bedrijf maar is dit meestal niet
  + Banner wordt geladen vanuit de webserver die beheert wordt door reclame-bureau
  + Die gaat dan info opslaan over de gebruiker, eventueel in combinatie met cookie …
    - Om een profiel van de gebruiker te maken
    - Social engineering
* Webbug
  + Klein transparant gif bestand 1x1 pixel groot dat op dezelfde manier als een banner werkt
    - Niet echt zichtbaar dat dit gebeurt, behalve als je de source html code bekijkt
    - Wordt meestal van een andere server geladen dan diegene die je momenteel aan het bekijken bent.

Alle info die over u verzamelt wordt, wordt gebruikt voor marketing en social engineering.

**Bescherming tegen webbugs**

* Gebruik een goede proxy server
  + Filteren van webverkeer => kijkt naar in de html code naar webbugs of banners en verwijdert ze
  + Kan soms java scripts verwijderen
  + Kunnen ook liegen over het type browser dat je gebruikt zodat specifieke lekken niet kunnen worden gebruikt
  + Kan je anoniem doen surfen door je IP verborgen te houden
* Gebruik een goede pop-up blokker (bijna altijd een banner in een pop-up)

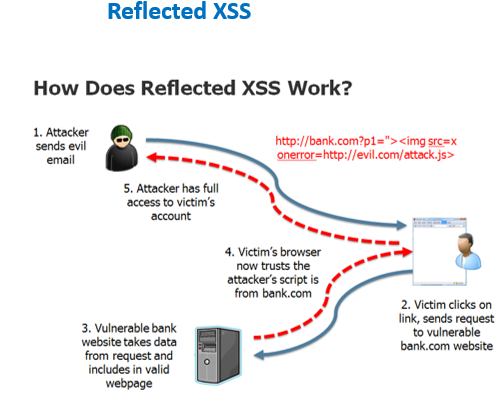
**8.11.4 Cookie stealing met XSS**

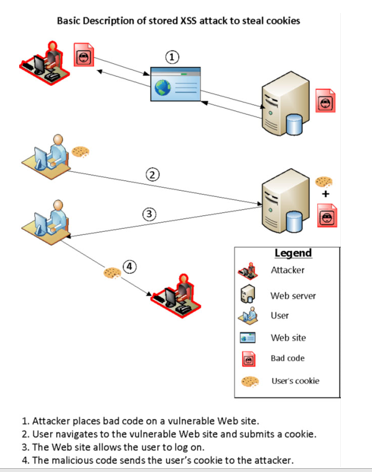
Sleutelconcepten van cookies:

* Authenticatie
  + Een cookie bewijst aan de server wie je bent en welke rechten je hebt
  + Als een hacker je cookie te pakken krijgt, kan hij zich voordoen als u
* Same origin principle
  + Een script van site X kan enkel die cookies gebruiken die afkomstig zijn van dat domein X (sandboxed) – hardcoded in iedere browser

**XSS: Cross-site scripting Cookie stealing**

* XSS
  + Proberen om het same origin principle te omzeilen
  + Hacker wil uw X-cookie stelen => script om dit te doen moet door het X domein aan u gegeven worden
  + Stored (persistent) XSS
    - De website gaan exploiten om daar het cookie-stealing script op te plaatsen
    - Het blijft op de server staan
    - Hacker kan aan de cookie van iedereen die naar die website komt
  + Reflected (non-persistent) XSS
    - Soort van witwassen van het script door het te laten returnen (reflecten) door een trusted site aan het target
    - Principe van **Code Injection** door de client laten doen zonder zijn medeweten
      * Client misleiden om script code naar een site te sturen (via social engineering truc zoals url) naar bv. een searchbar
      * Site valideert de code niet goed en returned zijn antwoord met die scriptcode in terug aan de client
      * Op deze manier is het script witgewassen: het komt van de trusted site dus het script mag aan de cookies van de site
      * Het is immers het same-origin principe
      * Script stuurt nu cookie van site X naar de hacker





**Concepten:**

|  |  |
| --- | --- |
| Concept | Beschrijving |
|  |  |