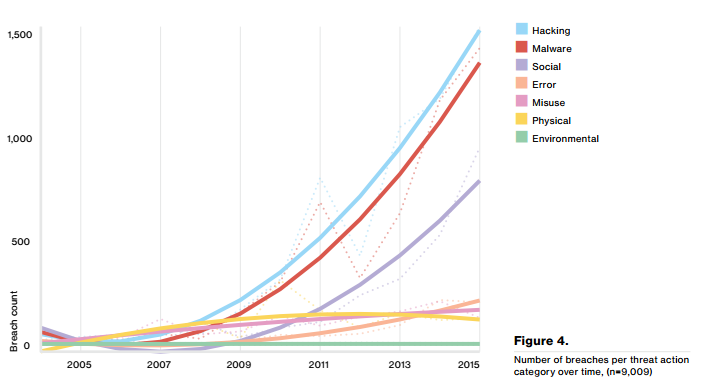
# DataSecurity

## Inleiding

### Algemene begrippen

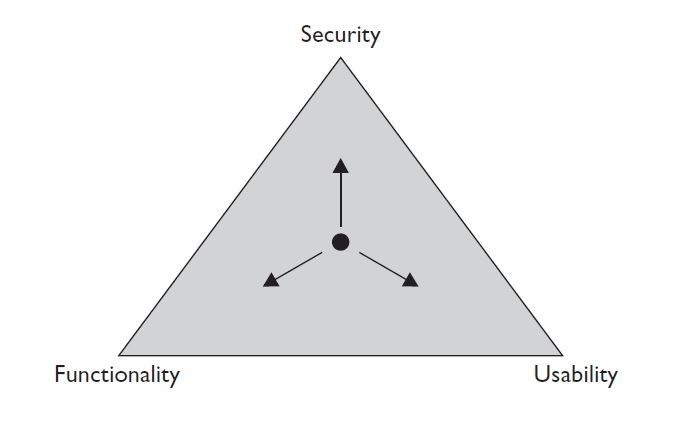
hacking 🡪 nog altijd een probleem

confidentialiteit

integriteit

Availability

Data services



De Engelse term die wordt gehanteerd is de CIA Triad: **c**onfidentiality, **i**ntegrity en **a**vailability.  
 ***Beschikbaarheid*** bevat de garanties voor het afgesproken niveau van dienstverlening gericht op de beschikbaarheid van de dienst op de afgesproken momenten (bedrijfsduur, waarbij rekening wordt gehouden met uitvalstijden, storingen en incidenten). Kenmerken van beschikbaarheid zijn tijdigheid, continuïteit en robuustheid. Zie ook [Business Continuity Management](https://nl.wikipedia.org/wiki/Business_Continuity_Management).  
 ***Integriteit*** geeft de mate aan waarin de informatie actueel en correct is. Kenmerken zijn juistheid, volledigheid en geautoriseerdheid van de transacties.  
***Vertrouwelijkheid* of *exclusiviteit*** is het kwaliteitsbegrip waaronder [privacybescherming](https://nl.wikipedia.org/wiki/Privacybescherming) maar ook de exclusiviteit van informatie gevangen kan worden. Het waarborgt dat alleen geautoriseerden toegang krijgen en dat informatie niet kan [uitlekken](https://nl.wikipedia.org/wiki/Uitlekken_(informatie)).  
 ***Toerekenbaarheid*** geeft de garanties dat bij een transactie waarin twee partijen betrokken zijn aantoonbaar is dat beide partijen deel hebben genomen aan de transactie; bijvoorbeeld een verzender kan aantonen dat de ontvanger ontvangen heeft en de ontvanger kan aantonen wie de verzender was.

**Confidentiality**

* Toegang tot informatie beperken
  + Geautoriseerde gebruikers
* Data in verschillende toestanden
  + In transit
  + At rest
  + In use
* Hoe beschermen?

Confidentiality houdt in dat de toegang tot data beperkt wordt tot geautoriseerde gebruikers. Data is waardevol en kan in verschillende vormen toegankelijk zijn:

* Als data verzonden wordt (in transit)
* Als data opgeslagen wordt (at rest)
* Als data verwerkt wordt in een applicatie (in use)

De bescherming zal neerkomen op een goede implementatie van encryptie. Dit omvat ook goed key management en een goed authenticatie systeem.

**Integrity**

Integriteit van data en services

* Enkel geautoriseerde aanpassingen

Welke gevaren / hoe beschermen?

* Data in verschillende toestanden
  + in motion
  + at rest
  + in use
* Services

Integriteit van data houdt in dat de data enkel aangepast mag worden door geautoriseerde gebruikers.  
In eerste instantie kan je denken aan het beperken van toegang tot bepaalde gebruikers, maar ook encryptie zal weer een rol kunnen spelen. Denk bijvoorbeeld aan lees en schrijfrechten op een bestandssysteem, die omzeild kunnen worden door het bestandssysteem aan te koppelen in een ander besturingssysteem.  
In het geval van verzonden data betekent het dat de data tussen zender en ontvanger niet aangepast mag worden. Hier is het probleem dat de data langs verschillende apparaten passeert die, zeker als je ze zelf niet beheert, niet te vertrouwen zijn. Er zal dan een soort van verificatie nodig zijn op basis van geheime informatie die toelaat om na te gaan of het bericht verstuurd werd door de afzender.  
Ook op het vlak van services zijn problemen met integriteit mogelijk. Denk bijvoorbeeld aan de mogelijkheid voor een aanvaller om bijvoorbeeld een website na te bouwen om login gegevens te pakken te krijgen.

**Availability**

Beschikbaarheid van data en diensten  
Gevaren en maatregelen

* Fysiek
  + Stroomvoorziening
  + Connectiviteit
  + ...
* over het netwerk
  + vb ddos

Uiteraard wil je dat de services en data die je aanbiedt, steeds beschikbaar zijn.   
Eén van de gevaren kan een fysieke beschadiging van de infrastructuur zijn. Hier is niet eens kwaadwilligheid voor nodig.

Via het netwerk bestaan er ook tal van mogelijkheden om de dienstverlening in het gedrang te brengen. De meest bekende is waarschijnlijk een DdoS aanval (die zich kan richten op verschillende lagen van de netwerkverbinding), maar ook hier is kwaadwilligheid geen probleem, denk maar aan het down gaan van websites voor ticketverkoop (of afschakelplannen).  
Tegenmaatregelen bestaan naast de specifieke maatregelen voor bepaalde aanvallen onder andere ook uit bijvoorbeeld redundantie en backups.

**Strategie voor beveiliging**

Preventive actions

* plaatsing firewalls, beperking fysieke toegang, backup, salting...
* ..

Corrective actions

* bugfix, updaten van vulnerable software,
* ...

Detective actions

* IPS, virusscan, audit, ...

**Terminologie**

* **vulnerability**: kwetsbaarheid. Deze krijgen telkens een uniek nummer mee voor identificatie. CVE-2016-xxxx (<http://www.cve.mitre.org/find/>)
* **exploit**: toont aan hoe een vulnerability kan misbruikt worden.

### Meet your enemy

**Externe aanvallers**

Hackers  
kunde tonen (opportunisme)  
hacktivism  
professionele organisaties

* crimineel
* veiligheidsdiensten (state

verschuiving:

* professioneler
* toegankelijker
  + voorbeeld: Google hacking
  + voorbeeld: Cain: <https://www.youtube.com/watch?v=HtiBReckA80> (video over ARP Poison Demo: )

( wiki )**ARP poison routing**, is a technique by which an attacker sends (spoofed) Address Resolution Protocol (ARP) messages onto a local area network. Generally, the aim is to associate the attacker's MAC address with the IP address of another host, such as the default gateway, causing any traffic meant for that IP address to be sent to the attacker instead. ARP spoofing may allow an attacker to intercept data frames on a network, modify the traffic, or stop all traffic. Often the attack is used as an opening for other attacks, such as denial of service, man in the middle, or session hijacking attacks. The attack can only be used on networks that use the Address Resolution Protocol, and is limited to local network segments.

**Interne aanvallers**

Legitieme gebruikers

* kwade bedoelingen
  + data theft
  + wraak
* goede bedoelingen
  + voorbeeld: slechte wifi dekking oplossen met eigen access point

Mobiele apparaten

* diefstal, verlies
* data buiten gecontroleerde omgeving (BYOD)

Leveranciers

* voorbeeld: Target PoS attack: HVAC company

### Management & wetgevende context

**Risicoanalyse**

Risico’s kunnen inschatten en beperken voor:

* Infrastructuur
* Data
* Gebruikers
* Applicaties

Inzicht in verantwoordelijkheid

* Mogelijkheden als beheerder van infrastructuur / applicaties
* Legale aspecten

**Belgische wetgeving**

‘hacking’ is volledig verboden. Enkele aandachtspunten:

* ook vb portscan zonder toelating is niet toegestaan.
* ook toegang verschaffen tot niet beveiligde systemen is strafbaar (vb wifi met eenvoudig wachtwoord, router met default password, …)
* regeling op komst om ethical hacking wettelijk te kaderen
* uiteraard ook steeds onderheving aan andere ‘algemene’ wetgeving, zoals vb wet op privacy

Hacking is een zeer vaag begrip, zelfs informatici verschillen van mening over de precieze betekenis van het woord. De meest gangbare definitie van hacking is het ongeoorloofd binnendringen in een computersysteem. Met de inbraak is meestal kwaad opzet gemoeid, maar ook onopzettelijk een verbinding tot stand brengen en die verbinding vrijwillig behouden, wordt als hacking beschouwd.   
  
Zelfs het hacken van een informaticasysteem dat niet of nauwelijks beveiligd is, is strafbaar.  
Bij de beoordeling van hacking maakt de wet een onderscheid tussen interne hackers (insiders) en externe hackers (outsiders). Insiders zijn personen die wel een bepaalde toegangsbevoegdheid hebben tot een bepaald systeem, maar die bevoegdheid overschrijden. Zij zijn alleen strafbaar als ze hacken om schade toe te brengen of met bedrieglijk opzet. Deze beperking geldt niet voor outsiders: zij zijn altijd strafbaar, ook al kraken ze een systeem 'met goede bedoelingen'. (bron: justitie)

**Policies**

Information security policy:

* zegt waarvoor en hoe IT-systemen mogen gebruikt worden. Ook de sacties bij overtredingen worden beschreven. (ook bekend als “acceptable use policy”

Information protection policy

* omschrijft de gevoeligheidsniveaus van informatie en wie toegang heeft tot die niveaus en op welke manier ze mag verspreid worden.

Password policy

* omschrijft hoe wachtwoorden moeten opgebouwd worden. Bevat oa minimum lengte, complexiteit, ...

Email policy  
Information audit policy

* definieert hoe, wanneer en wat moet geaudit worden. Soms wordt ook vermeld wie dat mag doen.

**Internationale security management frameworks**

ISO27001/ISO27002  
PCI-DSS  
Common Criteria (<https://www.commoncriteriaportal.org/>)

**ISO27001: zeer algemene richtlijnen**

Samengevat: standaard die de richtlijnen bevat om zelf een set van policies te ontwikkelen. Kan dus toegepast worden op elk bedrijf. ISO27002 bevat iets praktischere richtlijnen over hoe ISO27001 kan geïmplementeerd worden.  
This International Standard specifies the requirements for establishing, implementing, maintaining and continually improving an information security management system within the context of the organization. This International Standard also includes requirements for the assessment and treatment of information security risks tailored to the needs of the organization. The requirements set out in this International Standard are generic and are intended to be applicable to all organizations, regardless of type, size or nature. ….

Voorbeeld: A.13.1.1 Network controls   
Networks shall be managed and controlled to protect information in systems and applications.

**ISO 27001: inhoud**

1. Beleidsmatig (management)
2. Organisatorisch (verantwoordelijkheden)
3. Bedrijfsmiddelen (infrastructuur, netwerk, systemen en overige bedrijfsmiddelen)
4. Personeel (huisregels, fouten, diefstal, fraude, misbruik)
5. Fysiek (sloten, brandbeveiliging)
6. Communicatie en operatie (beheer van systemen, processen en procedures)
7. Toegangscontrole (password, biometrie)
8. Systeem- en softwareontwikkeling en onderhoud (documentatie, processen)
9. Continuïteit (calamiteitenvoorzieningen)
10. Regelgeving (Wet Computercriminaliteit, Wet Bescherming Persoonsgegevens)

**PCI-DSS**

Payment Card Industry – Data  
‘Checklist’ van praktische regels waaraan een bedrijf moet voldoen dat credit card informatie gebruikt.  
“Kredietkaarten zijn zeer veilig en praktisch. Jammer genoeg worden kaartgegevens soms toch nog misbruikt. Het goede nieuws is dat u zelf misbruik kunt helpen voorkomen via de PCI-DSS-norm, kort voor ‘Payment Card Industry Data Security Standard’. Deze norm werd door Visa International, MasterCard International, American Express, JCB en Discover gezamenlijk uitgewerkt. Indien u niet aan de norm voldoet, kunt u aansprakelijk worden gesteld voor eventuele schade. Dit kan tot erg hoge boetes leiden.” (bron: Worldline services)

### Aspecten van beveiliging

L8

Applicatielaag

Presentatielaag

Sessielaag

Transportlaag

Netwerklaag

Datalink laag

Fysische laag

**Zo sterk als de zwakste schakel**

onveilige protocollen

* http
* pop
* smtp
* …

foutieve implementaties

* vb heartbleed
* misbruik funtionaliteit

‘L8’: de gebruikers

* authenticatie
* social engineering

**‘L8’: menselijke zwakheden misbruiken**

* social engineering
* malafide mails sturen
* phishing
* ...

**L7: applicatieniveau**

* cross side scripting
* SQLi
* gevoelige info via foutmeldingen
* ...

**L2/L3/L4**

* sniffen van netwerkverkeer
* arp spoofing/poisoning/..
* ...

**L1: fysieke laag**

* fysieke diefstal van gegevens
* ongeautoriseerde toegang tot computer of datalokaal
* aangepaste firmware op routers, printers, switches, ...
* ...

## Cryptografie

### Encryptie

Codering → data voorstellen  
Doel encryptie: data coderen in onherkenbaar formaat

* Omkeerbaar voor select clubje
  + makkelijk voor doelpubliek
  + lastig tot onmogelijk voor anderen

Select clubje: toegang tot geheime informatie

* nodig voor encryptie en/of decryptie

Encryptie kan je gebruiken om informatie vertrouwelijk te houden en de integriteit ervan te bewaken. Heel waarschijnlijk heb je er zelf al gebruik van gemaakt, bijvoorbeeld bij https verbindingen of is je schijf geëncrypteerd. Laten we voor het vervolg ons even beperken tot het verzenden van informatie (hetzelfde geldt ook bij bijvoorbeeld opslag, maar het is duidelijker als er over twee partijen gesproken wordt).  
De zender zal de clear tekst data vervormen en zal hierbij gebruik maken van iets dat geheim is. De ontvanger beschikt ook over geheime informatie (mogelijk delfde, maar het kan ook verschillend zijn), die hem toelaat om de data te lezen.  
Zolang de geheime informatie die de zender gebruikt, alleen bij de zender bekend is, kan niemand de communicatiestroom zinvol aanpassen (integriteit). Zolang de geheime informatie van de ontvanger alleen door de ontvanger gekend is, kan niemand de cipher tekst omzetten naar clear tekst (vertrouwelijkheid).  
Op die manier kan zelfs een vorm van authenticatie ontstaan. Als de ontvanger een bericht kan omzetten naar (zinvolle) clear tekst, dan weet hij dat het bericht afkomstig is van de enige die beschikt over de informatie om de ciphertext te produceren. Als de informatie waar zender en ontvanger over beschikken verschillend is, wordt zelfs non-repudiatie of onweerlegbaarheid mogelijk).

Dit is een bericht

**Voorbeeld**

Geëncrypteerde Nederlandse tekst:

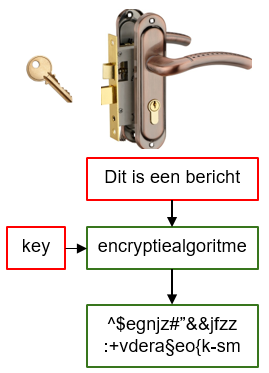
encryptiealgoritme

jhqlh

Wat is de bijhorende clear text?

^$egnjz#”&&jfzz :+vdera§eo{k-sm

De cipher text heeft niet meteen betekenis. Op dit moment is het achterhalen van de clear text erg moeilijk omdat niet eens bekend is welk algoritme gebruikt werd.  
Op basis van de enige kennis, met name dat het gaat om versleutelde Nederlandse tekst en de lengte van de cipher text, zouden we kunnen vermoeden dat het gaat over een woord van vijff letters (hoewel zelfs dit niet zeker hoeft te zijn).   
Dat maakt “vrouw” en “baard” bijvoorbeeld even waarschijnlijke opties, net zoals elk ander woord van vijf letters.  
Deze situatie, waarbij het gebruikte algoritme onbekend is, is echter weinig waarschijnlijk.

**Voorbeeld**

KerckHoff’s principe

* Crypto systeem bekend
* Waarom waarschijnlijk/noodzakelijk?

Oplossing: kracht van encryptie in sleutel

* geheime sleutel nodig om te decrypteren

Voorbeeld

* Versleutelde Nederlandse tekst
* Schuifsysteem
* ciphertext: jhqlh

Het geheim houden van het gebruikte algoritme is praktisch zelden haalbaar. Als we even het tekstuele voorbeeld verlaten en denken aan de praktische situatie waarbij je data beveiligd over internet wil verzenden. Als je dan het algoritme geheim wil houden, zijn er een aantal praktische problemen:  
het algoritme moet geïmplementeerd worden bij heel veel gebruikers, die verschillende systemen van verschillende fabrikanten gebruiken.  
als de kracht van de encryptie zit in het geheim houden van het algoritme, dan is er een groot probleem als het algoritme bekend wordt (bijvoorbeeld door reverse engineering). Het algoritme moet dan vervangen worden, wat een erg zware en dure operatie is. Bovendien kunnen alle met dit algoritme versleutelde berichten, plots weer omgezet worden in leesbare vorm.   
Als het algoritme bekend is, kan je de betekenis nog steeds verhullen door een geheime sleutel te gebruiken. Het voordeel van het gebruik van een geheime sleutel is dat die eenvoudig vervangen kan worden als die sleutel gecompromitteerd wordt.  
Als we terugkeren naar het voorbeeld en erbij zeggen dat het encryptiealgoritme een Caesar code is. Dit is een principe waarbij elke letter vervangen wordt door een andere letter die in het alfabet een vast aantal posities verder staat (waarbij je het alfabet circulair moet bekijken, zodat na de letter 'z' 'a' komt).  
De clear text wordt nu enkel beschermd door de geheime sleutel die bepaald over hoeveel posities er verschoven wordt. Je kan nu wel proberen de code te breken. Een evidente manier is alle mogelijke sleutels uitproberen, een zogenaamde brute force attack. Als je een paar sleutels uitprobeert, kom je al snel uit bij het woord “genie”.

**Brute force attack**

Algoritme bekend

* alle mogelijke sleutel proberen
* aantal mogelijke sleutels bepaalt moeilijkheid

Beperkingen:

* automatische herkenning clear tekst
* (perfecte) verhulling
  + voorbeeld: substitutiesysteem: cipher text: lz
  + clear text?

Een brute force is steeds mogelijk. Je kan niet verhinderen dat een aanvaller alle sleutels uitprobeert om je code te breken. Wat je wel kan doen, is proberen om het proces onhaalbaar te maken.   
In het voorbeeld kon de sleutel 26 mogelijke waarden aannemen, wat het zelfs voor een mens haalbaar maakt om alle mogelijkheden uit te proberen. Als je het proces automatiseert biedt deze codering dus weinig bescherming.   
Een codering die veilig is, moet het dus moeilijk maken om alle mogelijkheden uit te proberen. Eén manier om dit te bereiken is ervoor zorgen dat de berekening voor het decoderen complex en dus traag is. Het probleem met deze manier van werken, is dat het decoderen ook bij de bedoelde, vertrouwde ontvanger traag zal verlopen, wat uiteraard een ongewenst effect is.  
Een alternatief is zorgen dat er heel veel mogelijke sleutels zijn. Door dit aantal voldoende groot te maken, kan je de kostprijs van het aanvallen groter maken dan de waarde van de informatie (en dan is de aanval niet meer interessant).  
Een codering kan als veilig beschouwd worden als de waarde van de beschermde gegevens lager ligt dan die van de kost van de aanval. Hierboven werd er zelfs een letterlijke prijs op geplakt, maar het kan ook gaan om de moeilijkheid of duur van een aanval.  
Je kan bijvoorbeeld veiligheid bekomen als de beschermde gegevens hun waarde verliezen na een bepaalde tijd. Dan is elke codering die niet gebroken wordt op een kortere tijd, voldoende.  
Het voorbeeld had 26 mogelijke sleutels, dit is minder dan 5 bit en dus absoluut niet veilig. Het algoritme laat echter geen langere sleutels toe.  
Voor een geautomatiseerde brute force attack is het ook belangrijk dat je automatisch clear text kan herkennen. In het gegeven voorbeeld kan je heel makkelijk een programma schrijven dat voor de verschillende sleutels de clear text berekent, maar is het minder evident om het programma zinvolle Nederlandse tekst te laten herkennen. Als er gebruik gemaakt wordt van message integrity check’s, kunnen die wel door de aanvaller gebruikt worden.  
Perfecte verhulling betekent dat alle mogelijke clear tekst resultaten afgebeeld kunnen worden op de ciphertext. Dit betekent dat je met een brute force attack veel potentiële sleutels kan vinden, zonder zekerheid dat deze sleutels ook kloppen.  
In het voorbeeld zijn er vele sleutels die een mogelijk correct resultaat geven, eigenlijk elke combinatie van twee letters die samen een woord vormen voldoen. Als de tekst groter wordt zullen bepaalde combinaties afvallen.

**Veilige sleutel**

Von Neumann – Landauer: fysische limiet

* Niet reversibele computer
* Energie kT ln2 per omgeschakeld bit

128 bit sleutel:

* Enkel genereren alle sleutels: 2^128-1 keer omschakelen
* 10^18J of 30GW gedurende 1 jaar
* 262TWh of +/- 65.7Meuro

Computationele veiligheid

Doel: 2911MW 10 Doels gedurende gans jaar  
Let wel, dit is enkel de bescherming tegen een brute force attack. Veel aanvallen zullen proberen informatie te vinden over de sleutel, waardoor ofwel de sleutel gevonden kan worden ofwel een brute force met een beperktere selectie sleutels gedaan kan worden.  
Absolute veiligheid kan niet gegarandeerd worden, maar bovenstaande maakt duidelijk dat een brute force op deze sleutelgrootte computationeel onhaalbaar wordt.

**Crypto analyse**

Cryptoanalyse

* Zwakheden algoritme of kennis over plaintext
* Frequentietabellen, statistische analyse

Voorbeeld:

* Substitutiesysteem
* Clear tekst = Nederlandse tekst
* Ciphertext: “bzq zqza hc ahzq qz tnzqza”

Om meer sleutels mogelijk te maken moeten we een ander voorbeeld met meer bepaald een ander algoritme gebruiken. Als algoritme zouden we een permutatie van de letters kunnen gebruiken. Hierbij ga je elke letter voorstellen door een letter, met als enige verband tussen de gekozen letters dat ze verschillend moeten zijn.  
Zo kan je 'a' voorstellen door één van 26 mogelijke letters, voor 'b' heb je dan nog 25 keuzes over, voor 'c' 24, enzovoort. In totaal heb je dus 26! of ongeveer 4\*10^20 wat neerkomt op een sleutel van ongeveer 70 bit.  
Je kan nu opnieuw proberen om met een brute force attack de clear text te achterhalen. Met het aantal mogelijkheden voor de sleutel, ben je er waarschijnlijk niet toe geneigd dit handmatig te proberen.  
Een ander soort aanval maakt gebruik van eigenschappen van de clear text om via analyse te proberen de sleutel te vinden of het aantal te proberen sleutels te beperken.  
In het voorbeeld weten we dat Nederlandse tekst gecodeerd is. We zouden nu het aantal letters kunnen tellen en de meest voorkomende letters vervangen door letters die veel voorkomen in Nederlandse teksten. In dit voorbeeld zijn enkel de letters gecodeerd, strikt genomen zou je de leestekens en spaties ook kunnen coderen, maar dan moet je daar ook gewoon rekening houden met hun frequentie.

Als je het aantal keer dat een letter voorkomt gaat tellen, dan kom je al snel tot de conclusie dat de ‘z’ meest voorkomt. Nog rekening houdend met de positie in de woorden, is het waarschijnlijk een klinker en is ‘e’ een goede gok.Vervolgens kan je weer gaan tellen en op zoek gaan naar wat de letter ‘q’ voorstelt. Gebaseerd op veel voorkomende medeklinkers kom je uit bij ‘n’ of ‘t’. Op dit ogenblik ben je misschien al in staat om de woorden te raden. Anders ga je weer tellen en de nieuwe waarden uitproberen. Uiteindelijk komt je uit bij “het eten is niet te vreten”.  
Dergelijke analyse werkt beter naarmate er meer ciphertext beschikbaar is, waarbij die ciphertext dan wel op dezelfde manier gegenereerd moet worden.  
Om je gegevens dus te beschermen tegen dit soort aanval, moet je de hoeveelheid data die op dezelfde manier gecodeerd wordt, beperken. Je kan niet zomaar je gegevens inkorten, maar je kan wel de codering aanpassen door een andere sleutel te kiezen. Dit maakt het probleem van sleuteldistributie wel complexer: in plaats van één sleutel uit te wisselen moet de ontvanger nu verschillende sleutels te pakken krijgen en ze op het juiste stuk gecodeerde data toepassen.  
Merk op dat je door regelmatig de sleutel te veranderen, je ook beschermd tegen andere aanvallen. Bijvoorbeeld een brute force attack zal meerdere keren uitgevoerd moeten worden (één keer op alle blokken die een verschillende sleutel gebruiken). De waarde van het vinden van een sleutel is ook kleiner geworden, omdat die maar een deel van de data beschermd.

**Side channel attacks**

Side channel attacks

* Gebruik maken van nevenverschijnselen

Voorbeelden

* timing attack houdt rekening met rekentijd (<http://goo.gl/Fxzg1P>)
* iPhone keylogger aanval op desktop of laptop (<http://goo.gl/RcqU4>)

Side channel attack vallen de codering niet rechtstreeks aan, maar proberen nevenverschijnselen te gebruiken om informatie prijs te geven die kan leiden naar de sleutel.  
Timing attacks kunnen aan de hand van de duurtijd van de decryptie of encryptie te weten komen hoe de sleutel eruit ziet.  
Het voorbeeld van de keylogger valt niet de codering aan, maar de codering wordt beschermd met een paswoord, door dit paswoord te proberen detecteren, is de aanvaller in staat de sleutels voor de decryptie zelf te genereren.

**Algoritme voor encryptie**

Goed algoritme

* beschermt data op basis van (geheime) sleutel
* is publiek beschikbaar
  + ook voor onderzoek naar veiligheid

Maak gebruik van bestaande algoritmes

### Doelstellingen

**Confidentiality 🡪 Informatie geheim houden  
Integrity 🡪 Garantie op correctheid van de informatie  
Authenticatie 🡪 Correcte versleuteling data geeft zekerheid over identiteit  
Non-repudiatie 🡪 Onweerlegbaarheid**

**Codeertechniek**Symmetrisch – asymmetrisch

* Secret key (shared key) – public key

Block cypher

* Bericht in blokken (bepaalde grootte)
* Blokken versleuteld

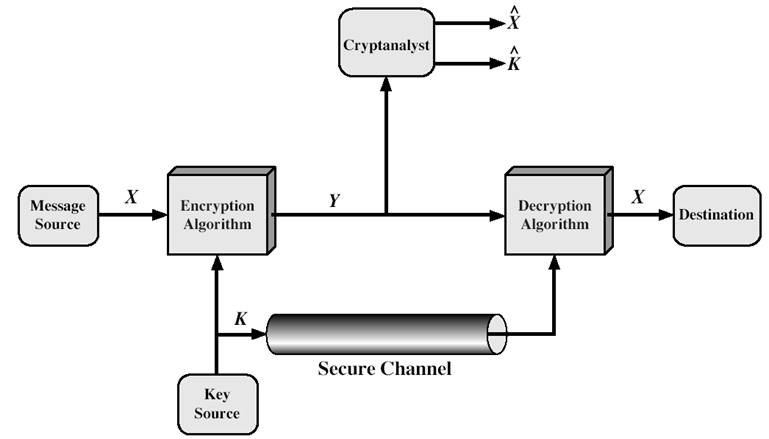
Stream cypher

* Symbool per symbool versleutelen
* Geen padding nodig

Qua codeertechniek kunnen een aantal onderscheiden gemaakt worden. Het eerste verschil is tussen secret key en public key encryptie of tussen symmetrische en asymmetrische coderingen.  
Bij een symmetrische key gebruiken zender en ontvanger dezelfde sleutel, deze sleutel moet dan uiteraard geheim zijn. Bij asymmetrische encryptie gebruiken beide partijen verschillende sleutels. Afhankelijk van het doel dat je met de codering wil bereiken kan één van beiden dan ook publiek gemaakt worden. Dit maak het probleem van sleuteldistributie gemakkelijker (maar lost het nog niet noodzakelijk op).  
Een tweede verschil is dat tussen een block cypher en een stream cypher. Bij een block cypher wordt een blok van bepaalde grootte versleuteld (bijvoorbeel 64, 128 of 512 bits zijn typische blokgroottes). Als je een datastroom wil versleutelen, betekent dit dat je die stroom moet onderverdelen in een aantal blokken en eventueel de datastroom moet uitbreiden met padding bits om een volledig blok te bekomen.  
Stream ciphers zullen de datastroom symbool per symbool versleutelen. Ze doen dit typisch sneller dan block cipher, maar kunnen op een beperktere hoeveelheid gegevens ook minder complexe operaties uitvoeren.

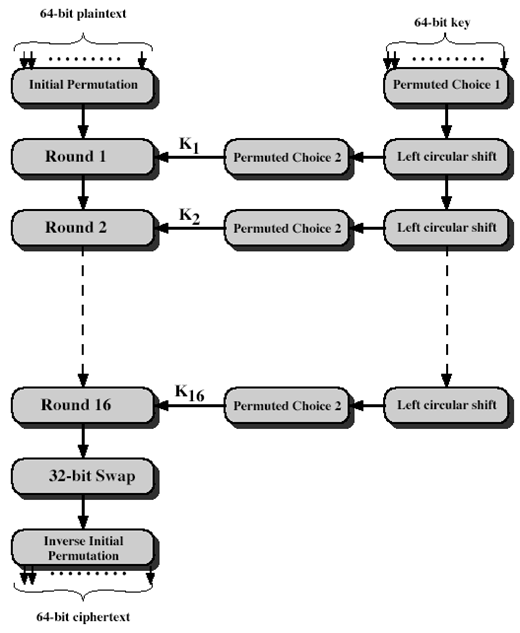
### Symmetrische encryptie

**Zelfde geheime sleutel voor encryptie en decryptie**

****

Schema van secret key, waarbij beide partijen dezelfde sleutel gebruiken die op een veilige manier gedistribueerd moet worden. Dit kan via een kanaal dat absoluut niets met de communicatiestroom te maken heeft (bijvoorbeeld fysieke uitwisseling), maar dit is niet altijd een haalbare manier van werken.

**Doelen**confidentiality?  
integrity?  
authenticatie?  
non repudiatie?

****Voorbeelden:  
- RC4: 40 tot 2048 bit (stream cipher)  
- DES: 56 bit  
- 3DES: 56 - 168 bit  
**- AES: 128, 256, 512 bit**  
- Blowfish  
**- Twofish**  
**- Threefish**  
**- Serpent**

**DES  
1974: IBM codering geselecteerd  
Samenspraak NSA  
DES:**

* **Block 64 bits**
* **18 functies**
  + **Transpositie en permutaties**
  + **Sleutels afgeleid van 56 bit sleutel**
* **Geoptimaliseerd voor hardware**

**AES**1998: echte opvolger DES

* Rijndael algoritme (Daemen – Rijmen)
* Blok 128 bit, sleutel 128, 192 of 256 bit

Aantal ronden

* Rondesleutels afgeleid van hoofdsleutel
* Bytesubstitutie
* Rijrotatie (permutatie)
* Kolomvermenigvuldiging
* Sleuteltoevoeging

AES is de echte opvolger van DES met een Belgisch randje. Het algoritme werkt op blokken van 128 bit en kan werken met verschillende sleutelgroottes die allemaal wapenen tegen brute force. Ook hier werkt het algoritme in een aantal ronden, waarbij een aantal operaties gebruikt worden die een aanval bemoeilijken (de kolomvermenigvuldiging is bijvoorbeeld een transformatie die niet in tabelvorm kan opgeslagen worden (te veel ruimte: 4GB), waardoor ze telkens berekend moet worden)

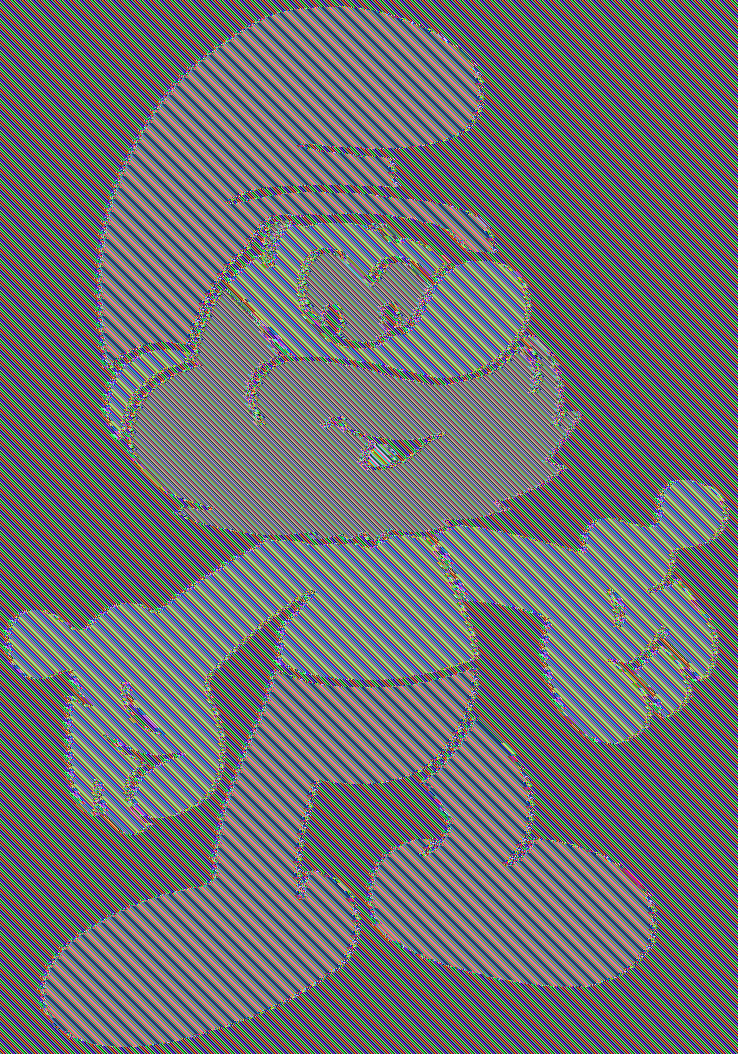
**Block encryption modes**

Grote hoeveelheden data encrypteren

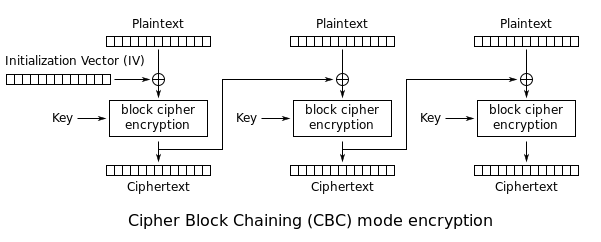
* Data opdelen in hanteerbare blokken
* blokken encrypteren

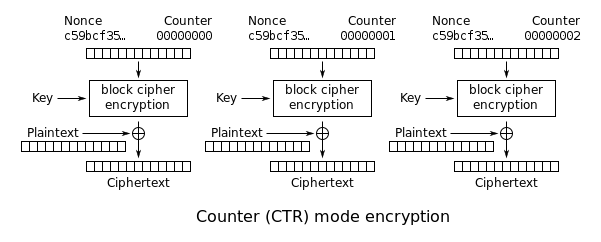
Verschillende modes om blokken te encrypteren

* ECB
  + alle blokken op zelfde manier → nadeel?
* CBC
* CFB
* CTR

Typisch is de te encrypteren data groter dan één blok. Dan moet het encryptiealgoritme een aantal keer worden losgelaten op opeenvolgende blokken. Dit kan op verschillende manieren.  
De meest voor de hand liggende manier is ECB. Hierbij laat je gewoon hetzelfde encryptie-algoritme met dezelfde sleutel los op de verschillende blokken. Nadeel van deze methode is dat ECB toch wat informatie prijs geeft. Weet je bijvoorbeeld hoe het bericht begint, dan zal ECB een plain tekst-cipher tekst paar bekend maken voor de aanvaller. Plain text die gelijk is, zal ook gelijke cipher text opleveren, dit is dan weer interessante informatie voor een cryptanalist.  
(ECB ) 

**CBC: IV + cipher text voorgaand blok**

****In cipher block chaining mode gebruik je de cipher tekst van het vorige blok om je volgende plain tekst blok te vervormen alvorens je het encrypteert. Op die manier worden gelijke stukken clear text op een andere manier vervormd en geven ze een ander verzonden resultaat. De ontvanger beschikt ook over het nodige cipher blok om na decryptie het resultaat terug om te kunnen vormen naar de juiste plain tekst.  
Enkel bij het begin moet er afgesproken worden hoe de eerste plain tekst wordt vervormd. Dit kan met een initialisatievector (IV) die bij beide partijen bekend moet zijn. Als deze ook geheim kan blijven, zal die de plain text ook extra beschermen.

**CTR: teller encrypteren xor** ****  
Teller waarde moet verschillend zijn voor elk blok  
Voordelen:  
Parallelle encryptie en decryptie mogelijk. Snelheid in hardware groter, in software ook indien software parallelle processing mogelijk maakt.  
Preprocessing mogelijk (niet afhankelijk van plaintext).   
Elk blok apart aanspreekbaar  
Zenders die ook ontvangers zijn moeten alleen encryptie algoritme implementeren  
 **Symmetrisch encryptie (samengevat)**1 gedeelde, geheime sleutel

* Voordelen
  + Snel
  + kleine sleutels
* Nadelen
  + uitwisseling sleutel

### Asymmetrische encryptie

Asymmetrische codering

* Sleutel voor encryptie, andere voor decryptie
* Eén van beiden bekend (public key)

Eisen?

Complexere wiskundige operaties

* Vorige: eenvoudige operaties op bitpatronen

Bij een asymmetrische codering gebruiken zender en ontvanger verschillende sleutels. Dit laat toe om één van beide sleutels bekend te maken (afhankelijk van welk doel je wil bereiken kan je kiezen voor die van de zender of de ontvanger). De sleutel die bekend gemaakt wordt, wordt de public key genoemd. De andere is de private key.  
Eisen aan de sleutels:  
Computationeel eenvoudig om een paar te genereren  
Private key is niet (computationeel eenvoudig) af te leiden uit de kennis van de public key  
Computationeel eenvoudig om decryptie en encryptie te doen (met de juiste sleutel)  
Public key encryptie gebruikt typisch complexere wiskundige formules die geen inverse hebben of waarvan het computationeel onhaalbaar is om het inverse te berekenen. Dit maakt dat je met een sowieso bekende public key niet instaat bent om het effect van de private key te creëren (daarvoor zou je het inverse algoritme nodig hebben).

**Diffie-Hellman key exchange**

Gemeenschappelijke secret key genereren

* “moeilijke” berekeningen
* Met publieke waarde en private key

Bijvoorbeeld:

* Publiek getal g, primitieve root van priemgetal q
* Private sleutels x1, x2
* Beide partijen publiceren y=g^x mod q
* Berekenen key = y^x mod q

Complexiteit berekeningen = basis public key

Veilig?

Diffie-Hellman key exchange pakt het probleem aan om een gemeenschappelijke key te genereren. Het maakt hierbij gebruik van moeilijk reversibele berekeningen gebaseerd op publieke waarden en private waarden  
Een voorbeeld:   
Vertrek van een priemgetal q en neem een tweede getal g dat een primitive root is van q (dit betekent dat elk getal kleiner dan q gevonden kan worden door een macht van g module q te berekenen).  
Beide partijen beschikken over een private geheime sleutel x1 en x2 en willen een geheime sleutel genereren. Hiervoor maken ze allebei y=g^x mod q bekend (dit betekent dat ook g en q bekend moeten zijn.  
Beiden kunnen ze nu een key berekenen: bijvoorbeeld de eigenaar van x1 krijgt een waarde y2=g^x2 mod q. Hij kan nu K=y2\*g^x1 mod q uitrekenen. Volgens de regels van het modulorekenen kan je dit ook anders schrijven: K=(g^x2 mod q)^x1 mod q= (g^x1mod q )^x2 mod q  
De laatste stap is de berekening die de eigenaar van x2 maakt. Dit betekent dus dat beide partijen in staat zijn om op basis van de bekende informatie en hun eigen private key een gemeenschappelijke waarde uit te rekenen. De vraag is nu natuurlijk of die gemeenschappelijke waarde enkel door hen kan worden uitgerekend, in dat geval hebben we een geschikte kandidaat voor een geheime sleutel gevonden.  
Een aanvaller beschikt over de waarde Y1, Y2, g en q en mag op basis hiervan niet in staat zijn de private sleutels terug te vinden. Nu is x1=dlog g,q (Y1) . Je kan dus een eenvoudige formule vinden om de private waarde te vinden. Helaas (voor de aanvaller) is het uitrekenen van de discrete algoritme een computationeel erg zware taak, waardoor de sleutel toch beschermd is, mits je geschikte waarden kiest voor g en q (groot). Uiteraard moeten ook de waarden van x1 en x2 voldoende groot zijn om te beschermen tegen een brute force attack.  
Op basis van het voorgaande kan je beslissen dat deze methode veilig kan zijn. Er is echter nog een belangrijk probleem dat aangepakt moet worden: in dit systeem is (nog) geen garantie ingebouwd dat de bekende waarden ook afkomstig zijn van de tegenpartij waar je mee wil communiceren. Dit maakt een MITM aanval mogelijk.

**RSA**

Twee priemgetallen p en q  
n=pq, z=(p-1)(q-1)  
Kies d<z, ggd(z,d)=1(relatieve priem)  
d heeft inverse mod z (e)  
Publieke sleutel: (n,e)  
Private sleutel: (n,d)  
Encryptie: Y = Xe mod n  
Decryptie: X = Yd mod n  
X,Y blokgrootte < log2 (n)

**Voorbeeld**

Publieke sleutel: (55,9)  
Elk karakter omgezet naar getal (a=1, b=2, …)  
Cipher: 4818254825  
Wat is het verzonden woord?  
Wat is de private key?  
Wat is de private key voor de combinatie (55,13)?  
Wat is de private key voor de combinatie (391,9)?

(55,9)

48^9mod55=3 48=‘c’

Herhaal voor andere letters en je vindt “choco”

Zoek een p en q, zodat n=55=p\*q (5\*11)

Z=4\*10=40,zoek d zodat d.e mod 40=40 of d\*9 mod 40=1 d=9

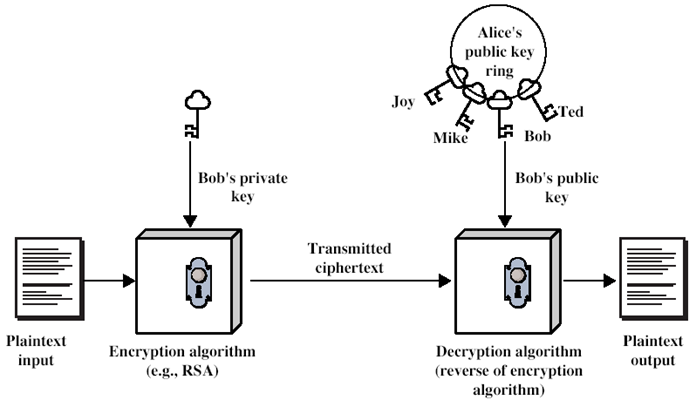
Voor 55,13: (55,37)

Voor (391,9): (391,313)

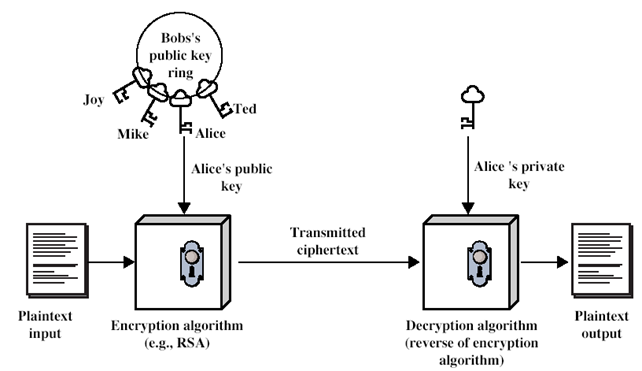
17\*23=391 (n)

Z= 16\*22=352

Je merkt dat naarmate het getal groter wordt, het moeilijker wordt om de priemdelers te vinden. Eerst moet je de priemgetallen zoeken, dan moet je de deelbaarheid testen.

Toepassing1:  
Private key voor zenden, public voor ontvangen  
Vertrouwelijkheid, authenticatie, integriteit, non-repudiatie?  
  
Het voorbeeld op de vorige slide gebruikte de private key voor de encryptie en de publieke voor decryptie.   
Als de private key gebruikt wordt voor het encrypteren, kan iedereen de publieke sleutel gebruiken om de boodschap te lezen. Vertrouwelijkheid wordt dus helemaal niet gegarandeerd.   
Het bericht kan echter maar door één persoon gegenereerd zijn. Daardoor ben je als ontvanger zeker over de identiteit van de zender (authenticatie) en kan die in dit geval ook niet ontkennen dat hij het verstuurd heeft (non repudiatie). Bovendien is het niet mogelijk voor iemand die het bericht onderschept, om het te vervangen door een ander versleuteld bericht. De aanvaller kan enkel meelezen.

Toepassing2:  
Public key voor zenden, private key voor ontvangen  
Vertrouwelijkheid, authenticatie, integriteit, non-repudiatie?

  
Als de public key gebruikt wordt voor het encrypteren, kan enkel de persoon met de private key de data lezen. Dit garandeert vertrouwelijkheid. Er is echter geen authenticatie (en dus zeker geen non-repudiatie) of integriteit, aangezien iedereen de publieke sleutel gebruikt kan hebben om de boodschap te genereren of een geëncrypteerd bericht te onderscheppen en te vervangen door een andere.

Voorgaande: keuze integriteit of vertrouwelijkheid  
Beide realiseerbaar met public key  
inclucief authenticatie en non-repudiatie  
Hoe?  
Beperking / nadeel?

Je kan alle eisen voldoen als je tweemaal asymmetrische encryptie toepast, eenmaal met de publieke sleutel van de tegenpartij en eenmaal met de eigen private key. De eerste laag encryptie garandeert de vertrouwelijkheid, de tweede garandeert inegriteit, authenticatie en non-repudiatie.  
Het nadeel is uiteraard de gevraagde rekenkracht (met alle erbij horende nadelen) ook verdubbelt.

**Voorbeelden asymmetrische encryptie**RSA: 512, 1024, **2048, 3072, 4096** bit key  
DSA: 512, 1024, **2048, 3072, 4096** bit key  
DH: 512, 1024, **2048, 3072, 4096** bit key  
ECC: **192, 224, 256, 384, 512** bit key  
 🡪 Veelbelovend   
 🡪 vrij recent → beperkt vertrouwen

**Asymmetrische encryptie**Twee sleutels  
1 publiek, 1 private  
Voordeel  
distributie publieke sleutel  
Nadelen  
grote sleutels  
berekeningen voor encryptie en decryptie duren langer  
Uitzondering: ECC  
Probleem sleuteldistributie opgelost?

### Hashing

Hash algoritme reduceert data tot waarde (digest)

* niet omkeerbaar
* collision resistant
  + moeilijk om H(x)=H(Y) te vinden
    - weak: voor gegeven x
    - strong: x, y te kiezen
* Voordelen
  + zeer snel
  + controle op waarde (checksum)
* Nadelen
  + Niet reversibel
  + niet bruikbaar als encryptie

**Voorbeelden hashing algoritmes**CRC32: 32 bit (geen cryptografische hash)  
MD5: 128 bit  
SHA-1: 160 bit  
SHA-2: **224, 256, 384, 512** bit  
SHA-3: **224, 256, 384, 512** bit

**Message authentication**

Doel:

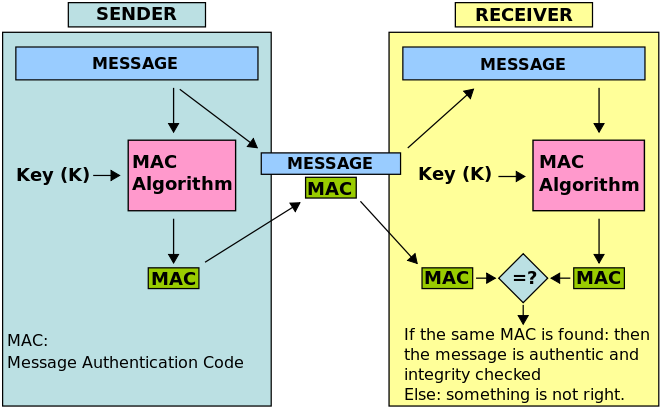
* Integriteit
* Identiteit afzender bevestigen, ev. non-repudiatie

Encryptie (symmetrisch en asymmetrisch)

* Soms niet-gewenste effecten
* Indien encryptie niet gewenst
* Integriteit bij encryptie

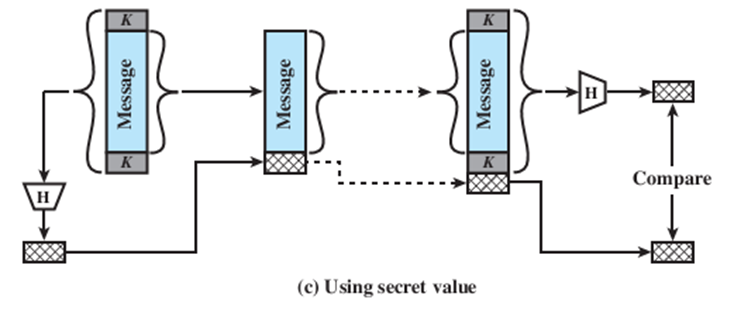
Klein blok met vaste lengte toevoegen

* Message authentication code (MAC)
* Geheime sleutel

Authenticatie kan zowel met secret als public key, maar in het laatste geval moet een private key gebruikt worden voor de encryptie.   
Aantal situaties te bedenken waarin encryptie niet gewenst (meestal omwille van rekenkracht), bijvoorbeeld seismografische waarneming USA USSR (doorgestuurde info controleerbaar en integriteit gegarandeerd) ter controle kernverdrag.  
Zelfs als encryptie gebruikt wordt blijft evenwel de vraag rond integriteit bestaan. Een Man In The Middle kan bijvoorbeeld wel de data manipuleren zonder sleutel. Hij kan de boodschap niet vervormen om de inhoud weer te geven die hij zou willen, maar hij kan de boodschap wel aanpassen (meest waarschijnlijk in complete nonsens). Integriteitscontrole moet de ontvanger toelaten vast te stellen dat de boodschap aangepast werd.  
Een Message Authentication Code is een stukje data dat aan het bericht wordt toegevoegd en dat toelaat aan de ontvanger om te verifiëren of de (clear text) data niet werd aangepast nadat die bij de zender verzonden werd.  
Hierbij wordt dikwijls gebruik gemaakt van hash functies.  


Gecombineerd met encryptie kan het resultaat van een hash functie gebruikt worden als MAC.  
In bovenstaande voorbeelden wordt enkel de MAC versleuteld. Indien gewenst kan ook het bericht zelf mee versleuteld worden.

Bij gebruik van public key encryption, creëert de combinatie van hash met private key encryptie een digitale handtekening. De handtekening is via de hash gekoppeld aan de inhoud van het bericht en via de private key aan de afzender.

Ook zonder MAC kan de hash functie het bericht verwijderen. Zender en ontvanger maken gebruik van dezelfde sleutel om het bericht aan te vullen en aan te bieden aan de hash functie. De sleutel zelf wordt niet mee verstuurd.

### Veiligheid

**Computeraritmetisch**

Sterk cryptosysteem:

* time complexity for decryption
  + lineair met sleutel
  + 2^n zonder sleutel

AES: voldaan

RSA:

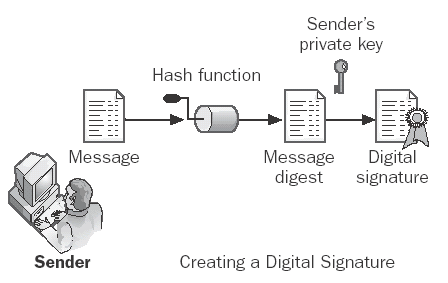
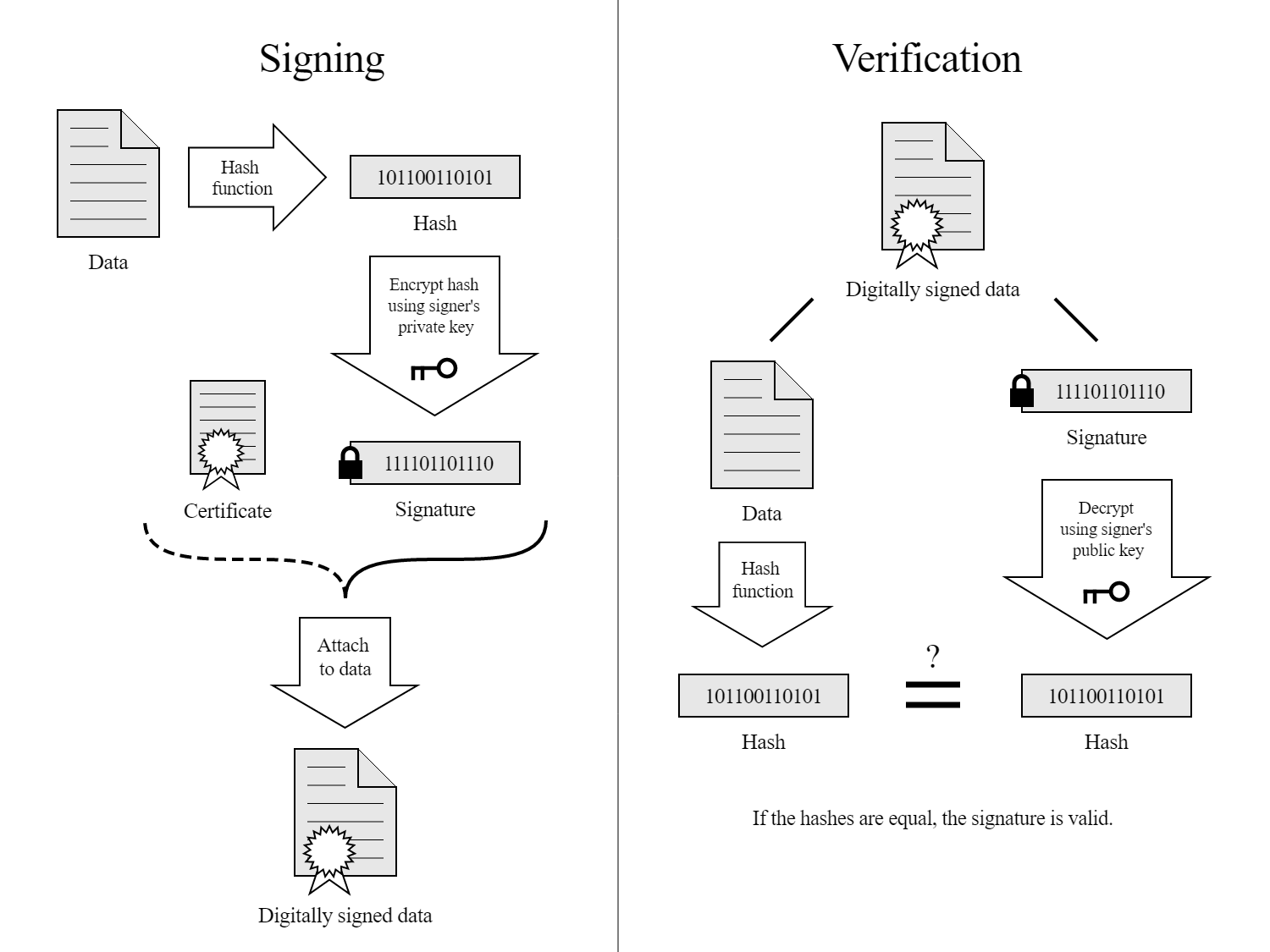
* factorisatie
* langere sleutels nodig
* quantum computing bedreiging
  + ook voor andere asymmetrische algoritmes (ook ECC)

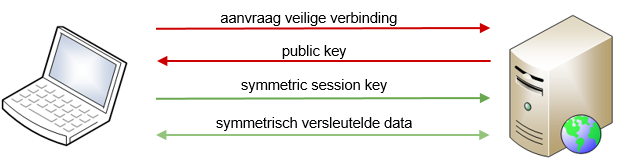
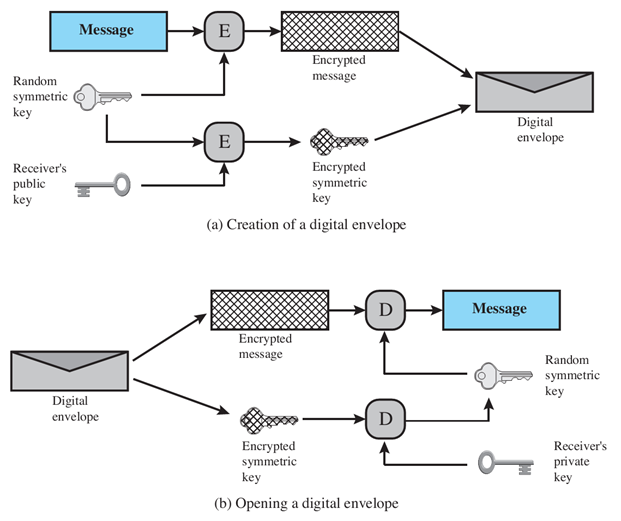
Een goed cryptosysteem maakt decryptie zonder sleutel een stuk lastiger dan met sleutel.   
Concreet moet gekeken worden naar de rekenlast naarmate de sleutel groter wordt. Als de rekenlast lineair toeneemt met het aantal bits in de sleutel als je die sleutel kent en exponentieel als je de sleutel niet kent, kan je makkelijk wapenen tegen de constant toenemende rekenkracht van nieuwe computers door de sleutellengte te vergroten.  
AES is een algoritme waarvan op dit moment geen noemenswaardige beveiligingsproblemen bekend zijn (en het gaat al meerdere jaren mee). Dat betekent dat de enige echte optie een brute force aanval is en die moeilijkheid neemt effectief exponentieel toe met de lengte van de sleutel.   
Voor RSA (en andere asymmetrische algoritmen) ligt de zaak anders, omdat er een verband is tussen de publieke sleutel en de private sleutel. Hier kan dan geprobeerd worden om dit verband aan te vallen. In het geval van RSA komt dit neer op het ontbinden in priemgetallen (factorisatieprobleem). Op dit moment is het snelste klassieke algoritme de “getallenlichamenzeef”, dat een complexiteit heeft die net onder de exponentiële ligt, maar die er dus wel voor zorgt dat voor dezelfde mate van veiligheid de sleutel grosso modo tien keer groter moet zijn dan bij AES.  
Nog dramatischer wordt het als quantum computing om de hoek komt kijken. Quantum computers werken op een heel andere manier dan klassieke en kunnen ook andere algoritmes uitvoeren. Voor het probleem van de priemgetallen is het algoritme van Shor dan een veel betere oplossing (log n^3), waardoor de veiligheid van het algoritme volledig op de helling komt te staan.

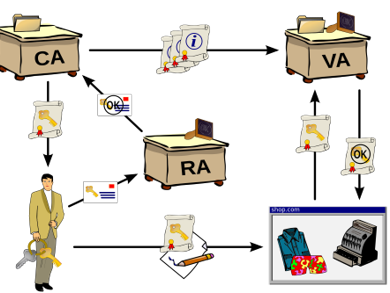
Op dit moment vormen quantum computers nog geen probleem, maar afhankelijk van de tijd waarover informatie geheim gehouden moet worden, kan het toch relevant zijn om hierbij stil te staan.  
Wat AES betreft zijn er geen quantumalgoritmen die kunnen helpen om de berekening zonder sleutel te versnellen. Als natuurlijk informatie versleuteld werd met AES en de sleutel zelf beveiligd werd met RSA, zal de quantumcomputer ook deze data bedreigen.

Andere aspecten bepalen mee de veiligheid van de encryptie. Neem bijvoorbeeld AES waarbij de sleutel afgeleid wordt van een gemeenschappelijk paswoord. In dat geval wordt een veilig cryptosysteem onveilig als het paswoord aangevallen kan worden.

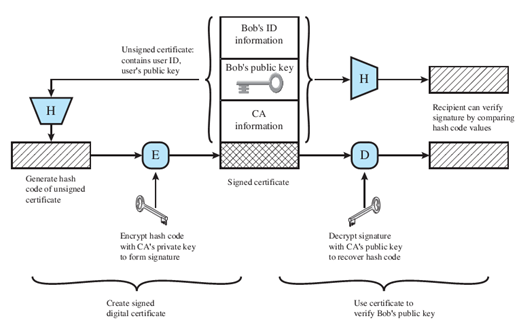
### Toepassingen

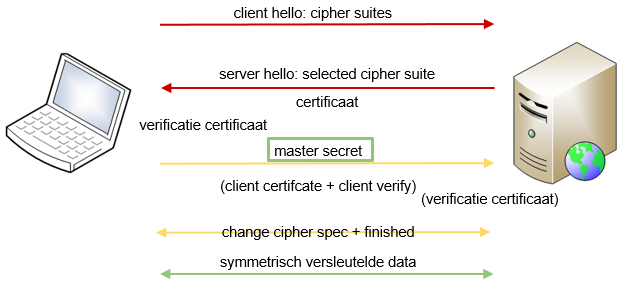
**Digitale handtekening  
**

  
Bovenstaand voorbeeld illustreert hoe een beveiligde connectie met een server zou kunnen ontstaan. De client vraagt een beveiligde verbinding, waarop de server reageert met een public key. Deze berichten kunnen onderschept en meegelezen worden.  
In een volgende bericht kan de client een session key versturen naar de server door gebruik te maken van die zijn public key. Het bericht zelf kan al versleuteld zijn met de session key, zolang deze sleutel zelf alleen versleuteld werd met de public key van de server. De server beschikt als enige over de private key en kan dus als enige de session key decrypteren.  
Op deze manier wisselen de twee machines vanop afstand op een veilige manier een geheime sleutel uit. De bulk van de data kan vanaf nu symmetrisch (en dus snel) versleuteld worden.

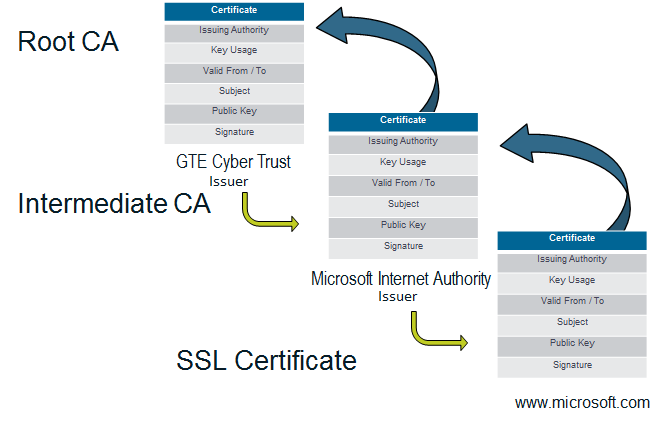
**Controle identiteit**Vorige voorbeelden missen verificatie identiteit

**PKI**Public Key Infrastructure  
hardware, software, people, policies, procedures  
create, manage, distribute, store, revoke certificates  
manage public key encryption

**Certificaten**Certificaten bevatten behoorlijk wat informatie, maar een paar essentiële zaken zijn een identiteit, ce public key die bij de identiteit hoort, de geldigheidsduur, informatie over de CA die het certificaat ondertekent en de ondertekening door de CA.  
Het principe van de digitale ondertekening komt hier terug. De eigenaar genereert een certificaat, dat hij laat ondertekenen door een CA. Die CA berekent een hash van het certificaat en versleutelt die met zijn private key.  
Als de eigenaar het certificaat doorgeeft aan een tegenpartij, dan kan die aan de hand van de public key van de CA n verifiëren of de onddertekening klopt en of het certificaat ook echt hoort bij de identiteit die erin vermeld wordt.   
Het probleem van de verificatie is nu verschoven naar de identiteit van de CA.

**SSL/TLS**De figuur geeft een vereenvoudigde weergave van de SSL/TLS handshake.   
In de eerste fase zullen client en server eigenlijk eerst berichten uitwisselen om een bepaalde cipher suite te selecteren. Als er een gemeenschappelijk cipher suite gebruikt kan worden, biedt de server zijn certificaat aan.   
De client kan nu de inhoud van het certificaat controleren en als die vertrouwd wordt, kan de informatie in het certificaat gebruikt worden om uiteindelijk een gemeenschappelijke key voor symmetrische encryptie vast te leggen.  
Als RSA gebruikt wordt in de cipher suite, kan de client bijvoorbeel een versleutelde random waarde aan de server bezorgen, die enkel met de private sleutel van de server ontcijferd kan worden. Dit is dan de master secret waar alle key materiaal uit afgeleid kan worden.  
Met een change cipher spec geeft de client aan dat alles vanaf geëncrypteerd zal verlopen en vervolgens stuurt hij een versleuteld finished bericht. De server kan dat laatste verifiëren en verstuurt zelf ook een change cipher spec en finished bericht. Als de verificatie goed loopt, kan vanaf dan data symmetrisch versleuteld en met integriteitscontrole verstuurd worden.

**Chain of trust: Chain of trust herleid vertrouwen naar voorgeconfigureerd vetrouwen in “beperkt” aantal CA’s.**

****  
Probleem private key CA

**Perfect forward secrecy**

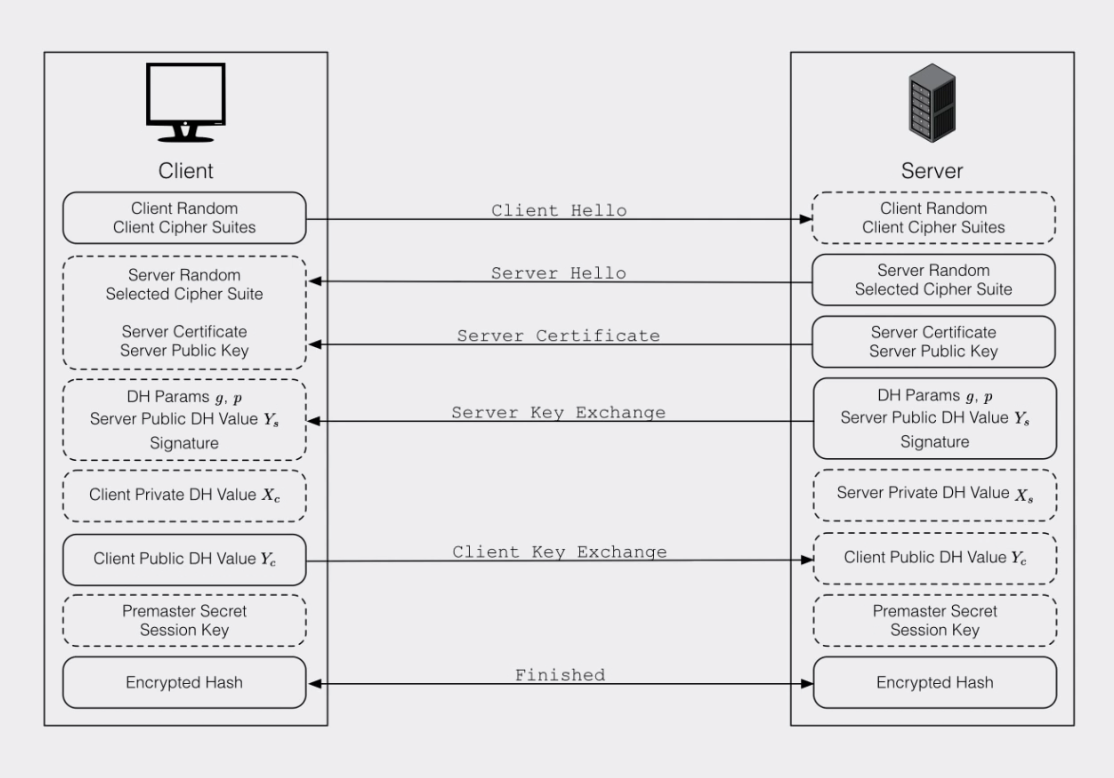
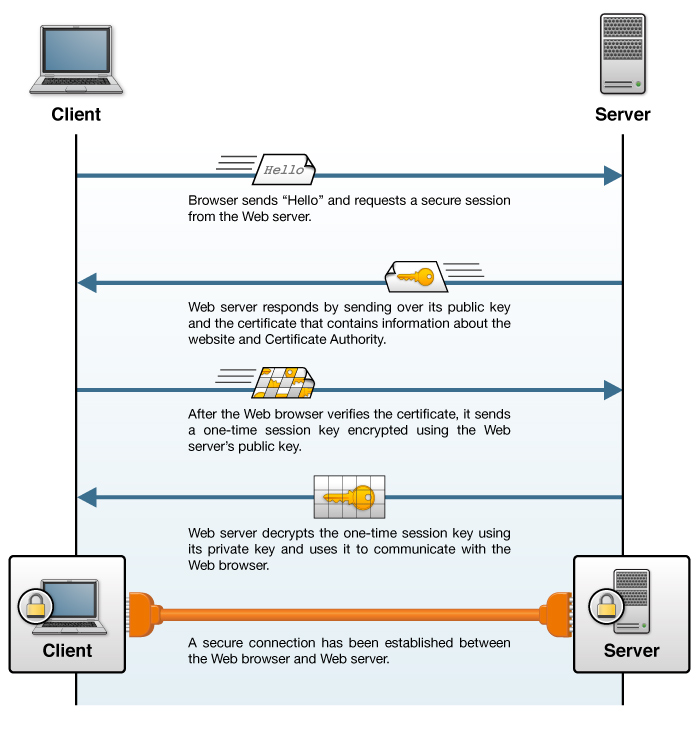
Bij perfect forward secrecy kan je stellen dat zelfs als de master key verloren gaat, dat je dan nog niet in staat bent om een gecapteerde conversatie te vertalen. Is dit zo bij RSA of Diffie Hellman?

## Beveiligde toepassingsprotocollen

### ssl, tls, https

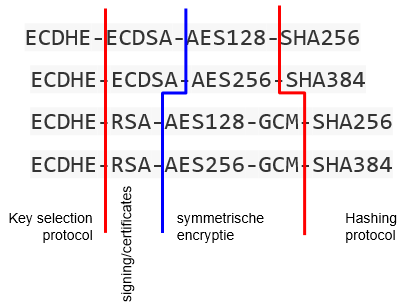
**https**: hiermee wordt aangegeven dat http over een secure kanaal (ssl of tls) gestuurd wordt  
**ssl**: (secure sockets layer): Naam van protocol door Netscape ontwikkelt voor beveiliging van http verkeer. Hiervan werden drie versies ontwikkelt, maar deze zijn deprecated.  
**tls**: (transport layer security): opvolger van het Netscape SSL-protocol dat in versie 1.0 quasi geen extra beveiliging bood tov ssl v3. De meest recente versie is TLS 1.3  
Voornaamste reden voor de naamsverandering van ssl naar tls is dat men rechtzaken wou vermijden over de ‘beschermde’ naam SSL.  
De termen https, ssl en tls worden vaak door elkaar gebruikt. SSL was de initiële naam van het protocol dat http veilig maakte.

**opzetten TLS met RSA**



**Keuze van protocolsuite (voorbeeld)**

bij het opzetten van de communicatie worden beschikbare protocolsuites uitgewisseld tussen client en server. Deze worden op bovenstaande manier genoteerd.



Recente problemen met https:  
BEAST attack (juni 2011)  
HEARTBLEED attack (april 2014)  
POODLE attack (oktober 2014)  
FREAK (maart 2015)  
DROWN attack (maart 2016)  
MITM: twee sessies opzetten, één tussen client en aanvaller, één tussen aanvaller en server. Mogelijk te merken aan certificaat dat client ontvangt.  
Heartbleed: buffer overflow in openssl implementatie  
Poodle: CBC weakness in ssl3,0, MITM nodig om ssl versie te bepalen en pakketten te injecteren.  
Beast: java applet om Same Origine Policy te omzeilen, CBC weakness uitbuiten

**HTTPS-problemen: concept blijft overeind**aanvallen gericht op:

* Onderhandeling:
  + vb degradation naar SSLv3
  + vb export keys
* Implementatie:
  + Bugs/zwaktes in openssl en andere ssl/tls-bibliotheken
    - vb HEARTBLEED
  + Slecht gekozen ciphers
    - vb standaard diffie-Hellman keys op veel servers

in metasploit:  
use auxiliary/scanner/ssl/openssl\_heartbleed  
set rhosts 10.129.28.241  
set rport 10002  
set verbose true  
exploit  
  
demo: **ssllabs SSL scanner**

### SSH

nog een voorbeeld waar public key encryption toegepast wordt   
in combinatie met cipher voor encryptie  
kan gebruikt worden als tunnel met kanalen voor andere protocollen (vb sftp)

### HSTS (RFC6797)

Manier om SSLstrip-achtige aanvallen tegen te gaan  
forceert https-connecties voor (sub)domeinen  
wordt ondersteund door alle recente browsers  
gedefinieerd in RFC6797

**HSTS: werking**

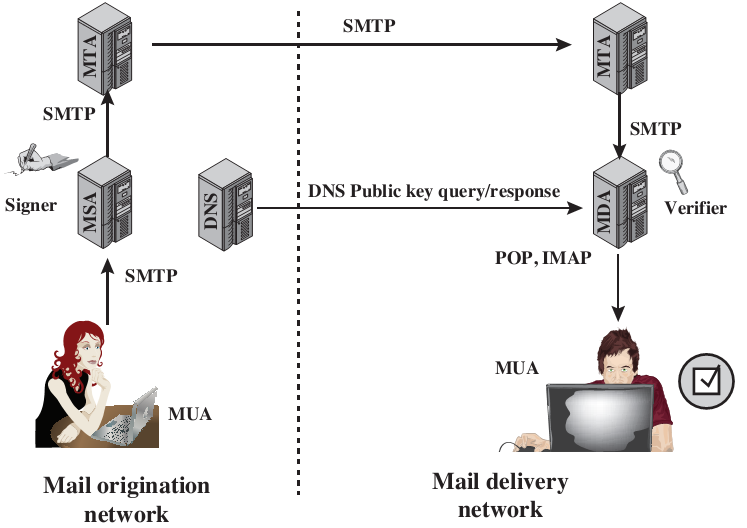
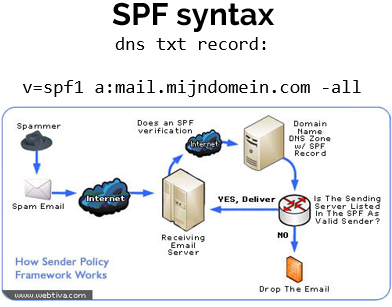
bij eerste request naar site wordt headerveld toegevoegd  
Strict-Transport-Security: max-age=15768000 ; includeSubDomains

### DKIM

**DomainKeys Identified Mail**

DomainKeys Identified Mail (DKIM) lets an organization take responsibility for a message that is in transit. The organization is a handler of the message, either as its originator or as an intermediary. Their reputation is the basis for evaluating whether to trust the message for further handling, such as delivery. Technically DKIM provides a method for validating a domain name identity that is associated with a message through cryptographic authentication.

DKIM-Signature: v=1; a=rsa-sha256; d=example.net; s=brisbane;  
 c=relaxed/simple; q=dns/txt; l=1234; t=1117574938; x=1118006938;  
 h=from:to:subject:date:keywords:keywords;  
 bh=MTIzNDU2Nzg5MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTI=;  
 b=dzdVyOfAKCdLXdJOc9G2q8LoXSlEniSbav+yuU4zGeeruD00lszZ  
 VoG4ZHRNiYzR



### SPF: **Sender Policy Framework**

**Sender Policy Framework** (**SPF**) is a simple [email](https://en.wikipedia.org/wiki/Email)-validation system designed to detect [email spoofing](https://en.wikipedia.org/wiki/Email_spoofing) by providing a mechanism to allow receiving [mail exchangers](https://en.wikipedia.org/wiki/Mail_exchanger) to check that incoming mail from a domain comes from a host authorized by that domain's administrators.[[1]](https://en.wikipedia.org/wiki/Sender_Policy_Framework) The list of authorized sending hosts for a domain is published in the [Domain Name System](https://en.wikipedia.org/wiki/Domain_Name_System) (DNS) records for that domain in the form of a specially formatted [TXT record](https://en.wikipedia.org/wiki/TXT_record). [Email spam](https://en.wikipedia.org/wiki/Email_spam) and [phishing](https://en.wikipedia.org/wiki/Phishing) often use forged "from" addresses, so publishing and checking SPF records can be considered [anti-spam techniques](https://en.wikipedia.org/wiki/Anti-spam_techniques). [IETF](https://en.wikipedia.org/wiki/IETF) publication [RFC 7208](https://tools.ietf.org/html/rfc7208) defines Sender Policy Framework.   
**Andere beveiligde toepassingsprotocollen:** ftps: ftp over tls (niet te verwarren met sftp: ftp over ssh) smime: imap over ssl, pop over ssl

## Authenticatie

### User authentication

Doel: identiteit gebruiker vaststellen  
Middel

* Alleen gekend door gebruiker
  + Paswoord, PIN, antwoorden op vragen
* Alleen in bezit van gebruiker
  + Token (smart card, key card, fysieke sleutels)
* Alleen gebruiker is/doet
  + Statische biometrie (vingerafdruk, iris, handpalm)
  + Dynamische biometrie (stem, handschrift, snelheid typen)
* Combinatie

### Passwoorden

Identiteit koppelen aan paswoord

* Identiteit bepaalt rol
* Wie kent paswoord?

Gevaren?

* Hoe paswoord aanvallen?
* Ihb over netwerk

Het meest gebruikte systeem voor authenticatie zijn paswoorden.  
Het paswoord moet minstens gekend zijn door de gebruiker en het authenticerend systeem (al dan niet clear text).  
De gevaren manifesteren zich dus ook bij beide partijen:  
Gebruiker kiest te eenvoudig paswoord  
Gebruiker krijgt complex paswoord en schrijft het op  
Databank authenticerend systeem wordt gestolen (offline aanvallen)  
Speciefieke gebruiker en kennis hierover aanvallen  
onderscheppen authenticatieverkeer bij remote authentication  
Hergebruik van paswoord  
Het laatste is op zich geen probleem. Alleen is het hergebruik een probleem, zodra ergens het paswoord bekend wordt. Dit kan onder vele vormen gebeuren, bijvoorbeeld ook door de gebruiker gewoon te laten registreren op een website, die beheerd wordt door de aanvaller. Die kan het zichzelf dan erg makkelijk maken om het paswoord te achterhalen en kan dit paswoord uitproberen op andere websites.

**Aanvallen op wachtwoorden**  
Sniffing  
 - Paswoord verzonden over netwerk  
ID bekend, proberen paswoord te raden  
 - Tegenmaatregel?  
ID’s bekend: populaire paswoorden proberen  
Specifieke user aanvallen (kennis over slachtoffer)  
Workstation hijacking / user mistakes / social engineering  
Offline attack  
 - Tegenmaatregelen?

Bij sniffing probeert een aanvaller om het paswoord van een gebruiker te onderscheppen bij het aanmelden over het netwerk. Maatregelen die hiertegen genomen kunnen worden, kunnen zich concentreren op het niet clear text verzenden van paswoorden, maar ook in het netwerk zijn er mogelijkheden om het moeilijk te maken voor een aanvaller om zomaar iemand anders zijn communicatie te onderscheppen.   
Als een gebruikersID bekend is, kan de aanvaller proberen om het paswoord te raden. Hierbij kan gebruik gemaakt worden van kennis over de gebruiker (die al dan niet online terug te vinden is)  
Als meerdere gebruikerID’s bekend zijn, kan de aanvaller proberen aan te melden met die verschillende ID’s en kan hij die combineren met paswoorden uit een woordenboek, dat hij zelf kan opbouwen met de meest waarschijnlijke paswoorden.  
Dit kan in de vorm van een online attack. Dit kan handmatig, maar er bestaan ook tools om de aanval te automatiseren. De eenvoudigste tegenmaatregel is het gebruik van een lockout mechanisme, dat minstens een ernstige vertraging oplevert voor de aanval.  
Voor een offline attack moet de aanvaller eerst de opgeslagen paswoorden te pakken krijgen (databank of password file). Als het goed is, zijn de paswoorden niet clear text opgeslagen en moet de aanvaller nog proberen de clear text paswoorden te pakken zien te krijgen. Hiervoor heeft hij het aangevallen systeem niet meer nodig, de aanval kan doorgaan op zijn eigen infrastructuur. Als er paswoorden gevonden worden, kunnen die dan gebruikt worden om op het systeem aan te melden.  
De meest elementaire vorm van bescherming tegen dit soort aanvallen is het beschermen van de paswoord informatie. In de eerste plaats door de paswoorden niet zomaar clear text op te slaan en in de tweede plaats door te vermijden dat iemand de password file te pakken kan krijgen. Eens de paswoord informatie in handen is van de aanvaller, hangt het van de sterkte van het gekozen paswoord af hoe lang de aanvaller erover zal doen om het te achterhalen. Door regelmatig van paswoord te veranderen wordt dit soort aanval moeilijker, omdat de tijd waarin hij afgewerkt moet worden, verkleint. Dit aanpassen kan ook afgedwongen worden door het authenticerend systeem of kan, zodra de diefstal wordt vastgesteld, aan de gebruikers aangeraden worden. Dit laatste gaat natuurlijk wel gepaard met de nodige imagoschade. Voor de gebruiker komt het er dan ook op aan om zo snel mogelijk alle accounts aan te passen waarvoor hij het paswoord gebruikte (in het bijzonder als het gekoppeld is aan dezelfde username, die dikwijls een e-mailadres is).

**Veiligheid paswoord**

Welke eisen aan veilig(e omgang met) paswoord?

* User
  + Lengte
  + welke karakters
  + hoe onthouden
* authenticator
  + opslag paswoord
  + transmissie authenticatiegegevens
  + controle authenticatiegegevens

**Brute Force attack**Massive cracking scenario  
10 karakters: 1 week  
11: 1,84 jaar  
12: 1,74 eeuwen  
Afhankelijk van rekenkracht  
bescherming paswoord

Bovenstaande toont op basis van de vermelde website een aantal relevante gegevens ivm brute force password attacks.  
Belangrijk om hierbij op te merken is dat de berekeningen als een soort van vuistregel moeten geïnterpreteerd worden. Het aantal paswoorden dat per seconde geprobeerd kan worden, hangt af van de omstandigheden (online/offline), de beschikbare rekenkracht en de bescherming van het paswoord. Verderop zullen we zien dat paswoorden best gehashed worden opgeslagen. Als dit gebeurt, moet bij een poging om een paswoord de hash functie telkens opnieuw berekend worden. Hoe langer die berekening duurt, hoe minder paswoorden er per seconde geprobeerd kunnen worden.  
In de vermelde cijfers is duidelijk dat de tijd voor een brute force attack exponentieel toeneemt met het aantal karakters. Een goed paswoord heeft dus niet noodzakelijk veel karakters (uiteraard wel voldoende), maar maakt vooral de alternatieven voor de brute force aanval onbruikbaar.

**Opslag paswoorden**Clear text paswoord niet nodig om kennis ervan te verifiëren

* niet clear text opslaan
* niet encrypteren

Hashing

* voor authenticatie: trage hashfunctie interessant

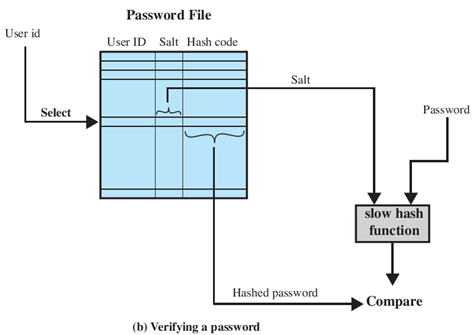
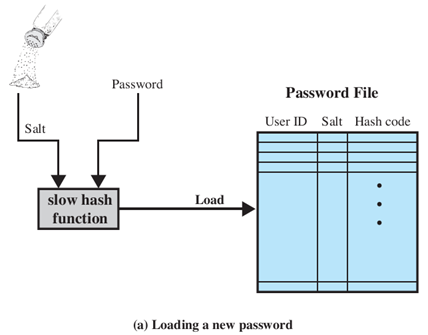
Salt:

* Vaste lengte
* Toevoegen aan paswoord voor hash
* Voordelen?

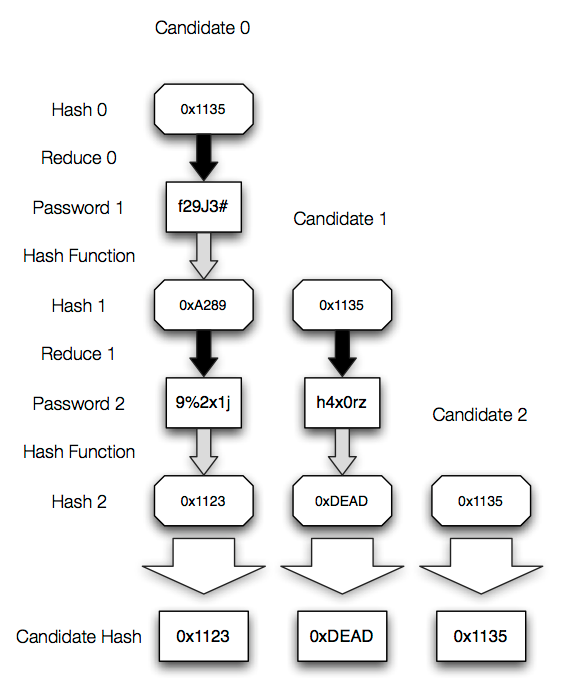
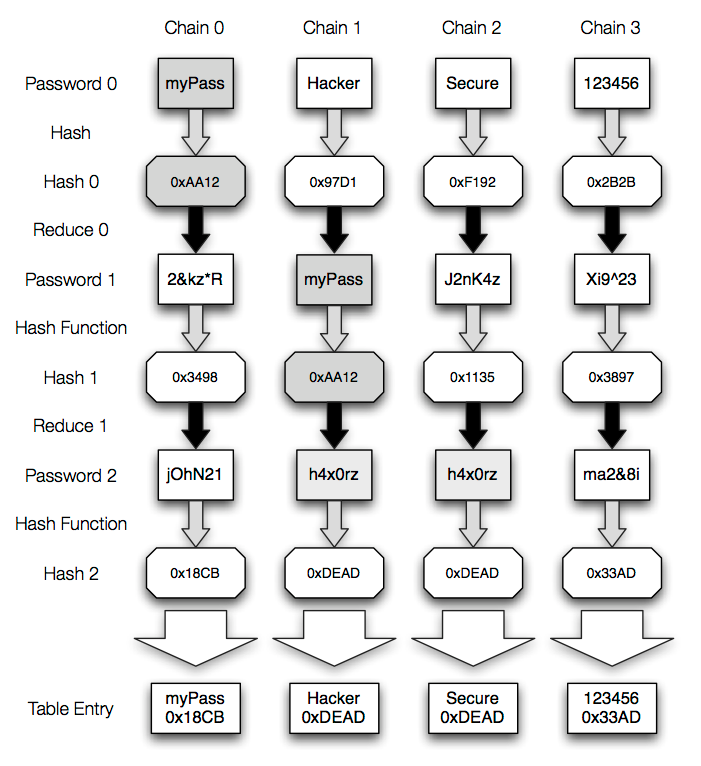
Bij het opslaan van paswoorden is het evident dat je dit omwille van de veiligheid best niet clear text doet.  
Je zou ervoor kunnen kiezen om de paswoordendatabank te encrypteren, maar dit is geen goed idee, omdat de volledige databank dan beschermd wordt door eenzelfde sleutel. Een gecompromitteerde sleutel geeft dan meteen de volledige database prijs.  
Het alternatief is om van elk paswoord een hash te berekenen en die op te slaan. Bij de controle van een paswoord moet het ingegeven paswoord dan ook gehashed worden en wordt het resultaat hiervan vergeleken met de opgeslagen hash.   
Aangezien een hash functie niet reversibel is, kan een aanvaller uit de gelekte databank niet meteen het paswoord afleiden. Hij kan enkel proberen om paswoorden te hashen en te controleren of hij dezelfde hash uitkomt. Als hij op die manier een paswoord te weten komt, weet hij nog niets over een ander paswoord (tenzij dat dezelfde hash-waarde heeft) en moet hij voor de andere paswoorden een gelijkaardige procedure doorlopen.  
Om gelijke paswoorden (eventueel op verschillende systemen) beter te beschermen kan een salt worden toegepast. Dit is een random patroon dat wordt toegevoegd aan het paswoord alvorens de hash functie te berekenen. De salt wordt bij het resultaat in de paswoorden databank opgeslagen, zodat bij controle van het ingegeven paswoord ook eerst de salt kan worden toegevoegd en de berekening herhaald kan worden.

Een aanvaller kan dus dezelfde aanval gebruiken op een gelekte databank, maar moet voor elk paswoord de aanval herhalen (in het vorige voorbeeld kon hij een paswoord hashen en vergelijken met alle hashes om eventueel een match te vinden), ook als gebruikers hetzelfde paswoord hadden gekozen. Om te controleren of een veel gebruikt paswoord voorkomt, moet de aanvaller het nu hashen met alle in de databank voorkomende salts.  
Hierbij valt ook het voordeel van een trage hash functie op. In tegenstelling tot andere toepassingen (bijvoorbeeld integriteitscontrole), is snelheid hier minder cruciaal. Als gebruiker wil je dit uiteraard zo snel mogelijk, maar een klein beetje vertraging speelt eigenlijk niet veel rol (eenmalig bij aanmelden), terwijl het een aanval op de password file een pak trager kan maken (heel veel hash berekeningen).

**SALT**



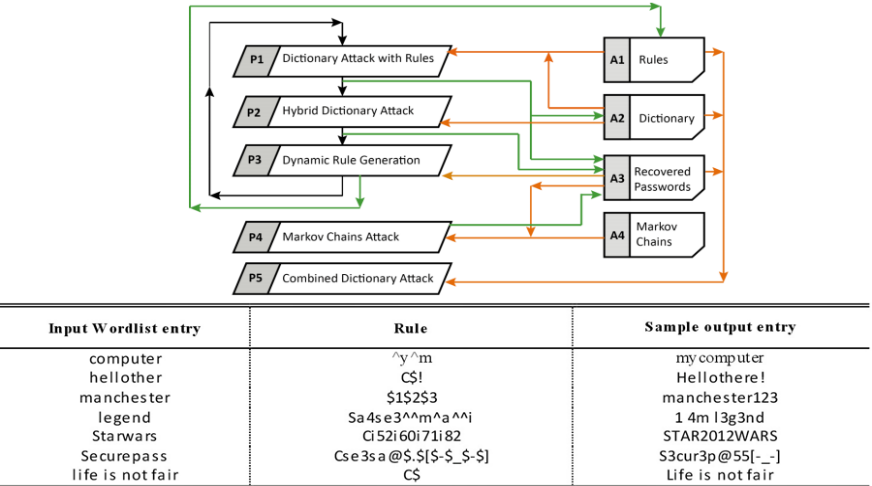
**Password cracking**Dictionary attacks  
Woorden hashen en vergelijken  
Veel rekenkracht  
Tegenmaatregel?  
Rainbow tables  
Op voorhand berekende hash vergelijken  
Veel geheugen nodig  
Tegenmaatregel?  
Voor paswoorden 7-10 karakters  
Korter: brute force met moderne rekenkracht  
Langer: opslagruimte overdreven groot

**Rainbow tables**Table generation:  
Afwisseling van hash en reductie functie om ketting van paswoorden en hashes te vormen. Enkel start en einde van chain worden opgeslagen.  
Hash functie is aangevallen functie (en dus steeds hetzelfde, zie conversie myPass), reductie is zelfgekozen en verschilt in de verschillende fasen (zie omzetting AA12 in fase 0 en 1).  
Cracking:  
Candidate hash generation (afb 2): aangevallen hash wordt in elke fase van de chain ingevoerd en levert telkens candidate hash  
Cadidate hashes worden opgezocht in tabel  
Chains die matchen worden opnieuw uitgerekend  
In elke fase wordt hash vergeleken met aangevallen hash, bij match hebben we paswoord  
Rainbow table attacks zijn beperkt in lengte van het aan te vallen paswoord. Bij lange paswoorden wordt de tabel te groot.  
Een eenvoudige tegenmaatregel is het gebruik van salts. Op windows systemen maakte NTLM het een tijdlang nog bruikbaar door lange paswoorden op te slaan als twee hashes van 7 bits. Het gebruik van deze methode kan in de moderne systemen uitgeschakeld worden in het register

**Dictionary attacks**

* Begonnen met woordenboeken
* Input van gevonden paswoorden → analyse
* Veel gebruikte strategieën:
  + Nummers, interpunctie aan einde
  + Hoofdletters aan begin
  + Bepaalde letters vervangen door nummer / speciaal teken:S $, E 3
* Zinnen
  + Bijbel
  + Youtube comments

Het vinden van paswoord is een interessante bron om woordenboeken aan te vullen of regels te verzinnen die automatisch kunnen worden toegepast op de woorden in de woordenboeken. Uit de gevonden paswoorden blijkt immers welke strategieën gebruikt worden om (in het beste geva) het paswoord complex te maken. Deze strategieën zal de aanvaller dan ook toepassen op de andere woorden in zijn woordenboek.  
Ook kan inspiratie gevonden worden in belangrijke werken (zoals de bijbel) of bijvoorbeeld youtube comments. Daar vindt een aanvaller misschien alternatieve schrijfwijzen (spelfouten, slang) die hij kan toevoegen aan zijn woordenboek.

**Passphrases (duh)**

**Pattern substitution**Patronen vervangen (A4, e3, hoofdletters, ‘ ‘ \_, …). Automatisatie kan ook op passphrases. Voorbeelden kunnen gevonden worden in enkele uren bij offline attack.

**Woorden lijsten:**

RockYou, LinkedIn

**Keuze veilig paswoord**Veel truckjes om “veilig” paswoord te kiezen

* bekend → toegepast voor samenstellen dictionary

Echt veilig: random patroon

* moeilijk te onthouden
* post-it?

Alternatieven

* Beginletters passphrase → random + te onthouden
  + eigennamen, leestekens
* Zinloze zin “otters vliegen blauw”
* niet hergebruiken → offline attacks irrelevant

**Veilig paswoord**

* user education
* reactive checking
  + periodiek cracker loslaten
  + beperkte resources en tijd tussen controles
* proactive checking
  + rule enforcement
  + regels bekend, patronen in paswoord
* password cracking
  + beperkte tijd, woordenboek
  + bloom filter

Bloom filter:  
K hash functies worden berekend voor elk woord in dictionary. De hash functies geven een resultaat tussen 0 en N-1.  
Er wordt een rij van N bits gemaakt (hash tabel), met een 1 op elke positie die een resultaat van de hash functie voorstelt.  
  
Bij de controle van een paswoord wordt elk paswoord gecontroleerd door alle hash functies te berekenen. Voor elke uitkomst wordt in de hash tabel gekeken of daar een 1 staat. Als voor elke uitkomst een 1 wordt gevonden, wordt het paswoord als kraakbaar bestempeld.

### Token based

Memory cards

* Slaan data op, verwerken ze niet
* Bijvoorbeeld bankkaart
* Speciale lezer nodig:
  + Verhoogt kost
  + Beperkt inzetbaarheid
* Verlies van token
  + Geen toegang voor gebruiker
  + Token vervangen
  + Beperkte beveiliging van PIN
* Gebruikers moeten bereid zijn systeem te gebruiken

Bij token based authentication moet de gebruiker een token hebben om geauthenticeerd te kunnen worden. Er bestaan verschillende soorten van tokens, die als gemeenschappelijke eigenschap hebben dat ze een zogenaamd One Time Password (OTP) genereren. Dit is een paswoord dat de gebruiker in principe slechts eenmaal zal gebruiken, het is in elk geval slechts gedurende een korte periode geldig.   
Uiteraard moet er een soort van synchronisatie bestaan tussen het token en het authenticerend systeem, zodat dit systeem kan verifiëren dat het ingegeven OTP afkomstig is van het juiste token en dus gebruikt kan worden.  
Soms kan het bezit van het token ook gecombineerd worden met de kennis over bijvoorbeeld een PIN code. Hoewel de PIN code een beperkte beveiliging is (4 cijfers), is het door de onbruikbaarheid zonder token toch een veilig systeem. Gaat het token verloren, dan is het belangrijk dat een ander mechanisme de beperkte aanvulling van de pin beschermd (beperkt aantal verkeerde pogingen, annulatie token).

* Smart cards
  + Embedded microprocessor
  + Authenticatie protocol:
    - Statisch
    - Dynamic password generation:
      * periodiek nieuw paswoord
      * Initialisatie en synchronisatie nodig
  + Challenge response
  + Response op basis van random challenge
    - Berekening op basis van unieke kennis bij token

Statische tokens geven een statische secret nadat het device unlocked wordt. Aangezien dit paswoord niet verandert, beschermt het alleen beter dan een paswoord in de mate waarin het paswoord random en complex kan zijn.  
Dynamische paswoorden veranderen voortdurend en moeten omwille van hun korte levensduur minder complex zijn. De afbeelding toont een voorbeeld van RSA. Hierbij is een initialisatie en synchronisatie nodig tussen authentication server en token zodat die op dezelfde momenten dezelfde waarden gebruiken. Er zijn ook maatregelen nodig om rekening te kunnen houden met het verschuiven de synchronisatie (tijd die op één van de devices net wat sneller of trager verandert).

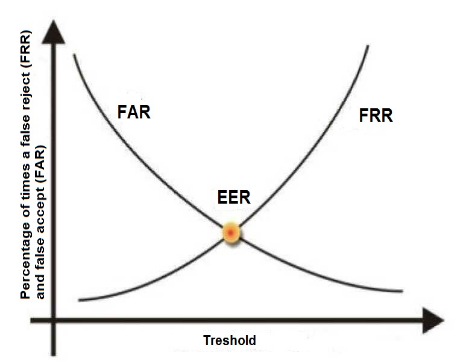
Met een challenge response mechanisme, wordt een antwoord berekend op basis van geheime informatie en een willekeurige waarde. Het antwoord hierop is bruikbaar binnen een beperkte tijdspanne na het genereren van de challenge en aangezien enkel de authneticator en het token beschikken over de noodzakelijke geheime informatie, wordt op deze manier een OTP gegenereerd.

### Biometrie

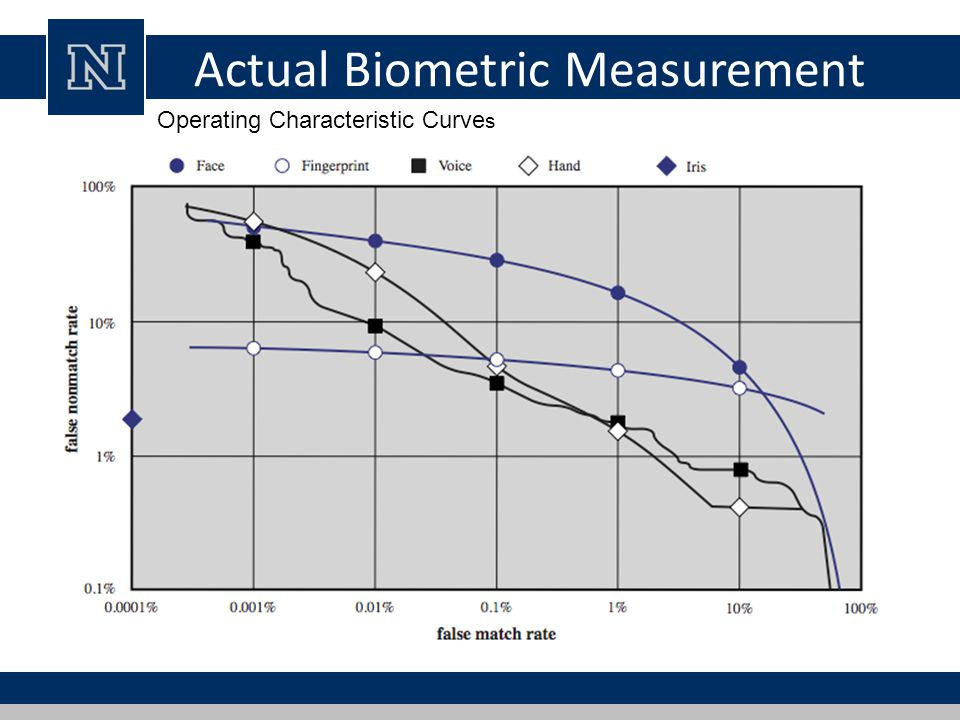
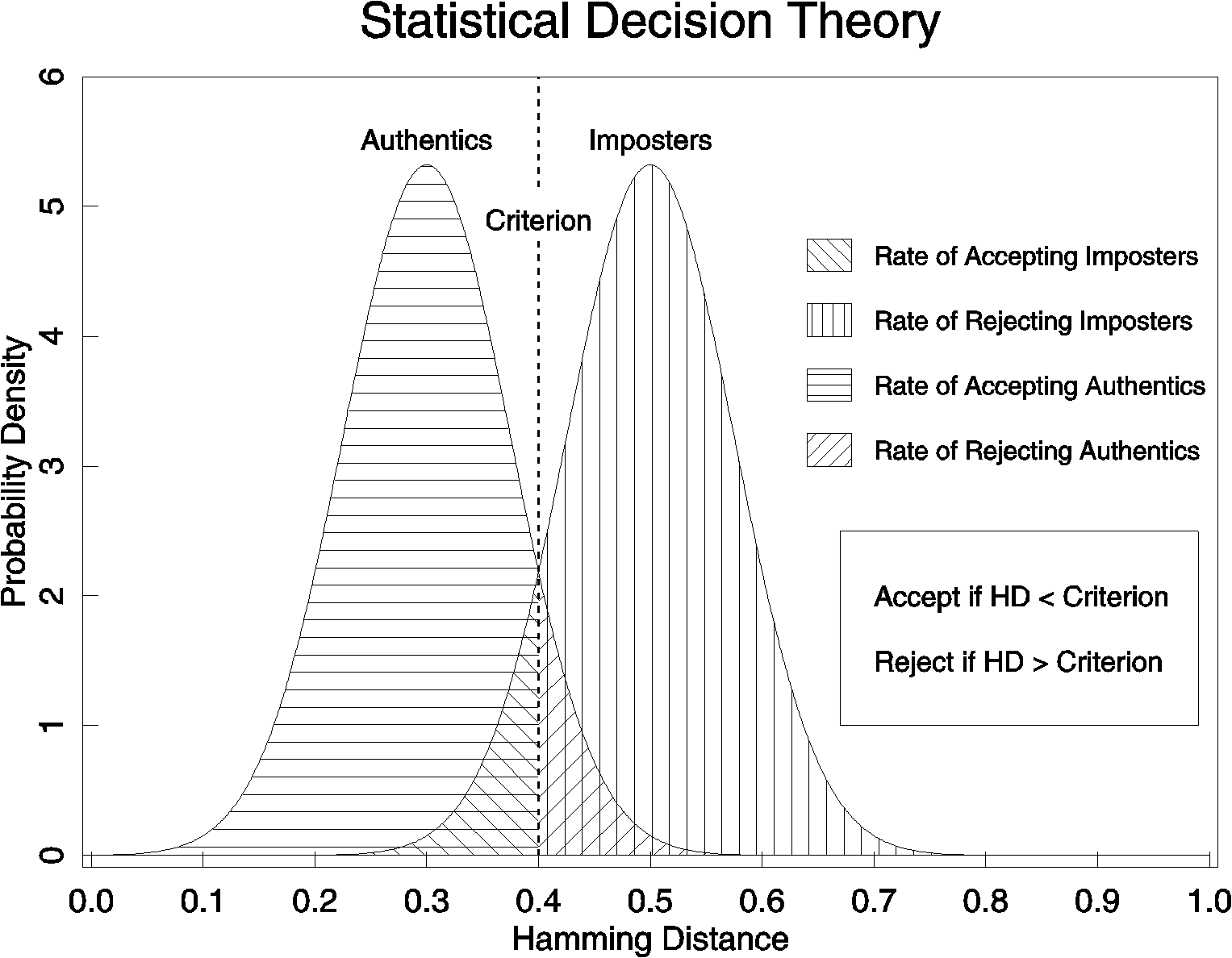
* Identificatie en/of authenticatie op basis van fysieke kenmerken
  + Gezicht
  + Vingerafdruk
  + Hand
  + Retina
  + Iris
  + Handtekening
  + Stem

Werking

* Enrollment
  + eigenschap scannen en digitaliseren
  + user template wordt opgeslagen
* Authenticatie
  + Verificatie:
    - gebruiker geeft extra info over identiteit
    - Identiteit wordt geverifieerd via biometrie
  + Identificatie:
    - Geen extra info
    - Zoeken in alle templates naar match

Uiteraard moeten eerst de biometrische eigenschappen van de gebruiker bekend worden bij het systeem. Hierbij zal een user template worden opgeslagen die later met de meting vergeleken kan worden.   
Bij de authenticatie kan er op twee manieren gewerkt worden. Ofwel geeft de gebruiker informatie over zijn identiteit op en wordt die geverifieerd aan de hand van de biometrische gegevens (verificatie), ofwel wordt er in alle templates gezocht naar een match en krijgt de gebruiker de identiteit van de user template waarmee een match gevonden wordt (identificatie).

**Nauwkeurigheid**Template = digitale voorstelling fysieke eigenschap  
Niet exact: geen volledige match bij correct aanmelden  
Dilemma: vals positief <-> vals negatief



De template is een digitale voorstelling van een fysieke eigenschap, deze is niet exact, waardoor een volledige match bij authenticatie niet mogelijk is. Daarom moet een gebied gedefinieerd worden waarbinnen een resultaat als een match gezien wordt.   
Hoe smaller dit gebied, hoe meer het zal voorkomen dat een scan van een geldige gebruiker geen match oplevert (false negative). Omgekeerd zal een breder gebied zorgen dat het waarschijnlijker wordt dat de scan van een niet-geregistreerde gebruiker een match oplevert (false positive). Het is duidelijk dat hier een compromis gezocht zal moeten worden, waarbij het compromis ook beïnvloed zal worden door de toepassing (naarmate er meer eisen op het vlak van veiligheid gesteld worden, zullen false positives een groter probleem worden).

**Feilbaarheid**False positives  
Kopiëren fysieke eigenschappen  
Software die authenticatie regelt  
Biometrie zorgt niet voor 100% veiligheid. Hiervoor hadden we al gezien dat er false positives mogelijk zijn. False negatives zijn eerder vervelend op het vlak van gebruiksgemak, maar false positives kunnen ongewild toegang verlenen.  
Daarnaast bestaat ook de mogelijkheid om de scanners om de tuin te leiden door de fysieke eigenschappen na te bouwen. De link leidt naar een artikel over een dergelijke hack van iphone authenticatie via vingerafdruk.  
Tenslotte is er ook de software die veilig moet zijn. De link geeft een voorbeeld van software die toeliet om aan te melden op windows, maar hiervoor het windows paswoord met zeer beperkte bescherming opslaat op de machine.

### Certificate based authentication

* Certificaat
  + Identiteit
  + Geldigheid
  + ondertekening door (erkende CA)
  + checken tegen CRL
  + public key
* Authenticatie
  + gebruiker toont bezit private key aan
* Erkend certificaat nodig
  + op apparaat, in applicatie

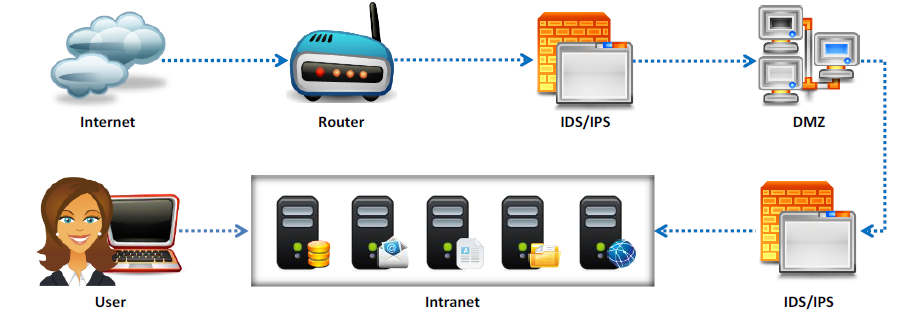
### Multi-factor authentication

* Combinatie
  + something you know / are / have
* Betere bescherming
  + beide mechanismes gelijktijdig omzeilen
* Veel voorkomende combinatie
  + paswoord + sms code
    - 2 factor
  + Voorbeelden: google, twitter, dropbox, facebook, …
  + Keuze om wel of niet te gebruiken

**Authenticatie apps:**Genereren passcode   
bijvoorbeeld TOTP (tijd) en HOTP (counter)  
synchronisatie secret en tijd of counter nodig  
Voorbeelden: google authenticator  
gelijkaardig: authy, HDE OTP, Duo

* Andere interactie met gebruiker
  + authy one touch
  + sound login authenticator
  + vasco digipass for mobile
* Extra beveiliging in afscherming app
  + pincode / paswoord
  + biometrie
    - vingerafdruk
    - gezichtsherkenning

**Aanvullingen paswoorden**Controle accounts:  
<https://haveibeenpwned.com>   
account informatie gestolen  
niet noodzakelijk zekerheid over paswoord  
afhankelijk van website / lek  
offline attack mogelijk   
paswoord hergebruikt?

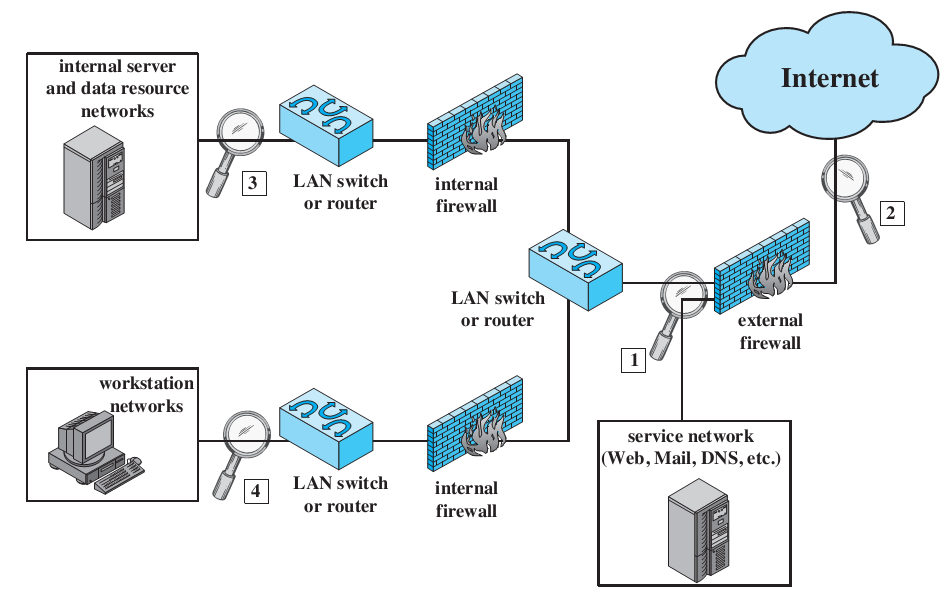
Adobe aanval:  
Netwerk (langdurig) gehackt  
Passwordfile: 9.3GB ~130 miljoen accounts ~56miljoen unieke paswoorden Geëncrypteerd met 3DES  
ECB  
Sleutellengte 3DES?  
Encryptie ongedaan maken, alle paswoorden bekend (onafhankelijk kwaliteit)  
Grote kans minstens een paswoord bekend → plaintext-ciphertext combinatie bekend  
ECB: entropie verlagen

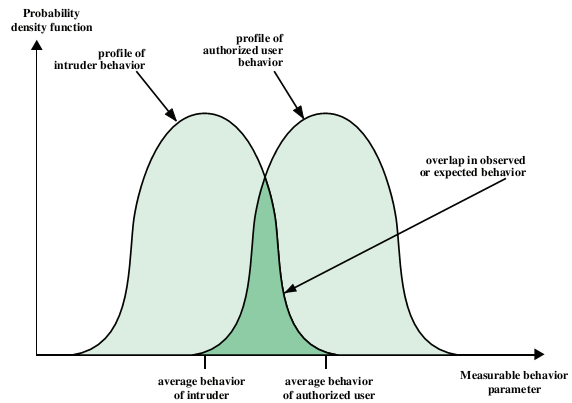
## IT security: detectie

### (N)IDS

Inspectie van alle inkomend en uitgaand verkeer  
op zoek naar verdachte patronen

**IDS vs IPS**  
(n)IDS: Intrusion detection system  
**Detecteert** problemen en **rapporteert** ze.   
De resultaten kunnen bijvoorbeeld doorgegeven worden aan firewalls etc om verbindingen af te sluiten. Vaak ‘offline’: de sensor wordt dan aan een ‘SPAN’-poort gekoppeld om zo een ‘kopie’ van het netwerkverkeer te inspecteren Voordeel: volledig onzichtbaar voor aanvallers...  
(n)IPS: Intrusion prevention system. kan zelf ingrijpen op netwerkverkeer (connecties sluiten, redirect, ..) **verkeer moet door dit toestel**, en kan ook geblokkeerd worden.

**Detectie binnen NIPS**

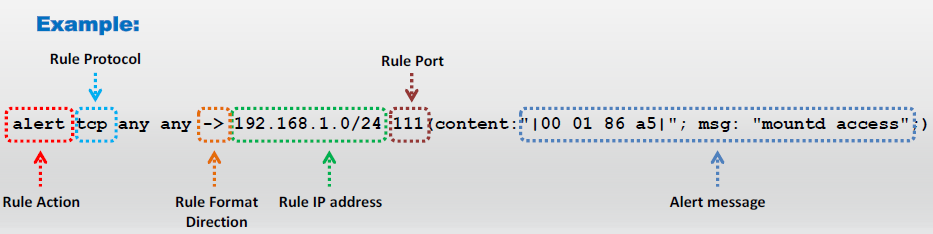
* Signature detectie
  + bepaalde byte-patronen worden gelinkt aan bepaalde threats
  + opletten voor false positives & false negatives
* Anomalie detectie
  + afwijkend verkeer detecteren
    - vb portscans, plotse verhoging bandbreedte, uitgaand verkeer, afwijkende pingpakketten, logs
  + vaak voorafgegaan door een meting/inventarisatie van ‘normaal’ verkeer (‘baseline’)
* Protocol anomalie detectie
  + detecteren wanneer standaarden niet gevolgd worden binnen protocols
    - vb: te grote payload, incorrecte flags, ..

opmerking: het herkennen van malafide patronen is niet evident. Een cryptolocker zal bijvoorbeeld alle bestanden encrypteren, maar dat doet een goeie encryptietool uiteraard ook… Iemand die 100 mails op tien minuten kan misschien verdacht lijken, maar het zou natuurlijk ook kunnen dat de jaarlijkse digitale nieuwjaarskaart net verstuurd wordt...

**Honeypot**  
Een ‘honeypot’ is een IT systeem dat is opgezet om aanvallers te lokken die het netwerk aanvallen.

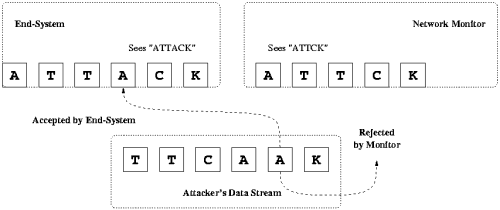
* Een honeypot zal:
  + verkeer loggen
  + alerts triggeren
  + verleidelijk gemaakt worden door bewust met een wat lagere beveiliging te werken
  + niet gekoppeld zijn met ‘echt’ belangrijke data
  + ...

SNORT:

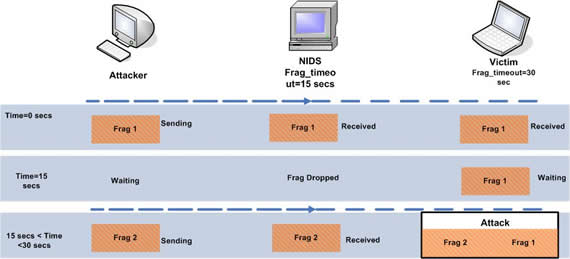


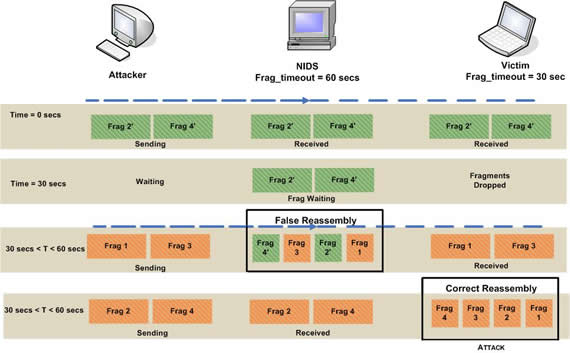
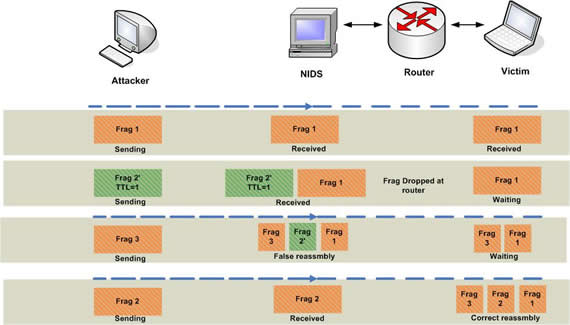
### IDS ontwijking

**Insertion technique**Aanvaller stuurt in een ‘aanval’ een specifiek pakket dat de bestemmeling niet leest, maar de IDS wel doorlaat. Bijvoorbeeld: bewust verkeerde checksums doorsturen, te lage ttl, …  
fragmentatie

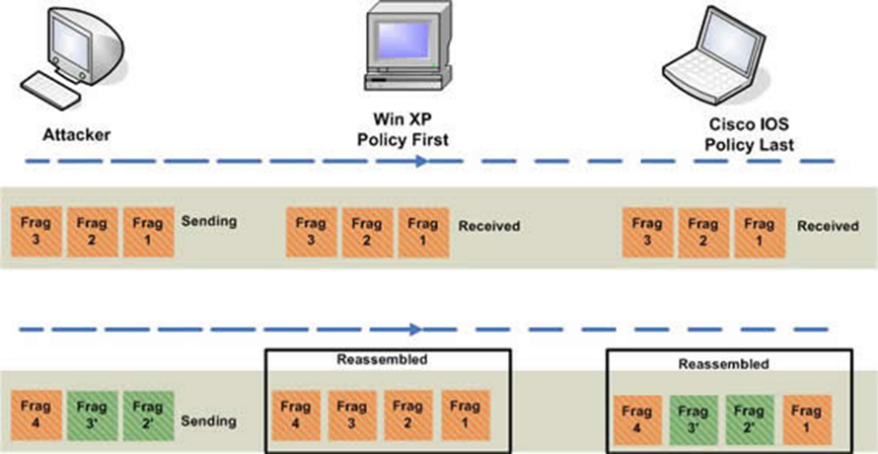
**of omgekeerd...**aanvaller stuurt in een ‘aanval’ een specifiek pakket dat de bestemmeling wel zou aanvaarden, maar de IDS niet doorlaat.

**Obfuscation**Encoding van payloads (vb naar unicode) zodat enkel nog de doelhost dit begrijpt  
voorbeeld: “\” kan voorgesteld worden door 5C, C19C, E0819C  
Polymorfe code ontwikkelen om signature based IDS systems te ontwijken  
Afwijkende bestandsnamen/paden/…  
Aanval uitvoeren over geëncrypteerde verbinding  
False positives genereren om het slachtoffer te verwarren  
Fragmentation  
Session splicing

**Fragmentation attack**IP pakketten kunnen opgesplitst worden als ze ‘ te groot’ zijn.   
dit maakt misbruik mogelijk door opzettelijk op te splitsen.  
om aanvallen te detecteren moet de IDS te fragmenten opnieuw samenstellen  
de aanvaller kan de timings vergroten tot de IDS z’n reassembly timeout overschreden wordt  
vb bij snort: 60 seconden

**Fragmentation attack: voorbeeld**voorbeeld: als de reassembly timeout 15 seconden is in de IDS en 30 seconden bij het target systeem, dan kan de aanvaller de fragmenten versturen na 30 seconden. De IDS zal niks detecteren, maar het target zal deze wel ontvangen.  
In dit voorbeeld is de timeout in de NIDS hoger dan die van het slachtoffer. Ook dan is een aanval mogelijk. Om deze aanval succesvol uit te voeren, moeten enkele valse fragmenten gebruikt worden. In het schema zijn dat fragmenten 2’ en 4’. Deze worden naar de NIDS gestuurd, die ze langer zal bewaren dan het slachtoffer. Als na de timeout bij het slachtoffer echte fragmenten 1 en 3 doorgestuurd worden, zal de ids proberen om 1,2’, 3 en 4’ opnieuw samen te stellen. Er wordt geen aanval gedetecteerd, dus worden de ‘echte’ fragmenten 1 en 3 naar het slachtoffer gestuurd.Als daarna ook nog echte fragmenten 2 en 4 gestuurd worden, dan heeft de nids dat niet door, omdat het voorbije samengestelde pakket reeds werd geflusht..

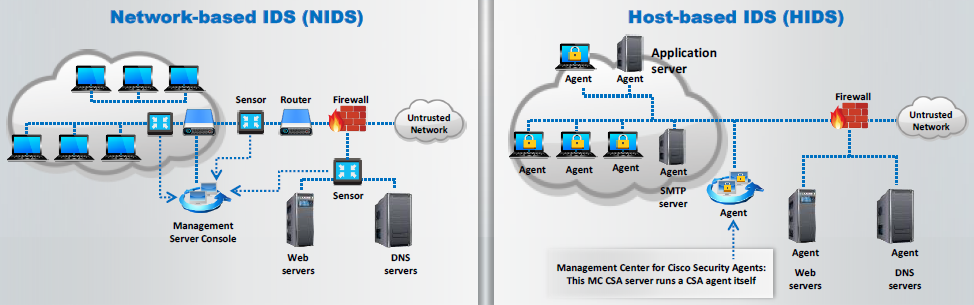
**Time-to-live Attacks**  
Hierbij worden opzettelijk pakketten geïnjecteerd met een te lage TTL om de IDS te verwarren.

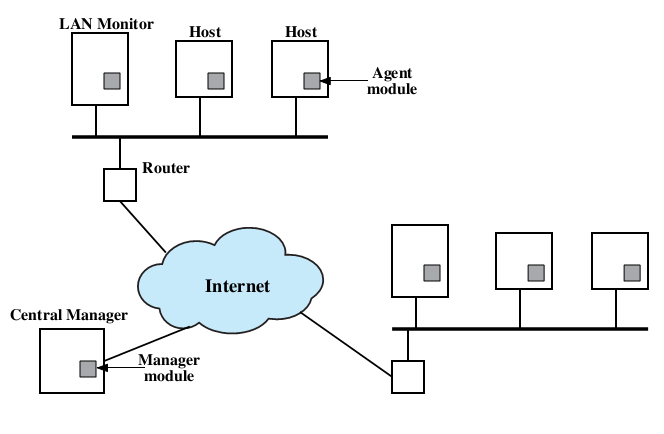
**Overlapping fragments attack**Maakt misbruik van de verschillende reassembly policies in toestellen. vb Windows geeft voorkeur aan originele fragmenten, cisco bewaart altijd laatste versie van bepaald fragment..

### Lokale malware-detectie

**Host intrusion detection systems**

Auditing van specifieke host  
bevat vaak ook virusdetectie  
Minder populair, omdat finetuning complex is & grote impact op performantie  
Auditing van vb file system permissies, file integrity (vb tripwire), registry, ...

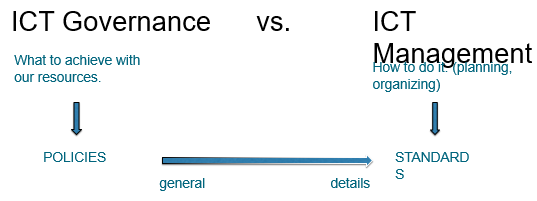
* Signature
  + verdachte bitpatronen definiëren
* Anomalie-detectie
  + parameters monitoren
    - tijd tussen aanmelden, gebruikte resources, grootte transfers
  + treshold detectie
    - eenvoudig, veel false positive/negative
  + profile based detection
    - gedrag uit verleden in kaart brengen
    - afwijkingen detecteren
    - niet gebaseerd op bekende zwakheden
* Sandbox technologie:
  + verdachte bestanden worden uitgevoerd in geïsoleerd systeem om gedrag te analyseren
* Toekomst: machine learning

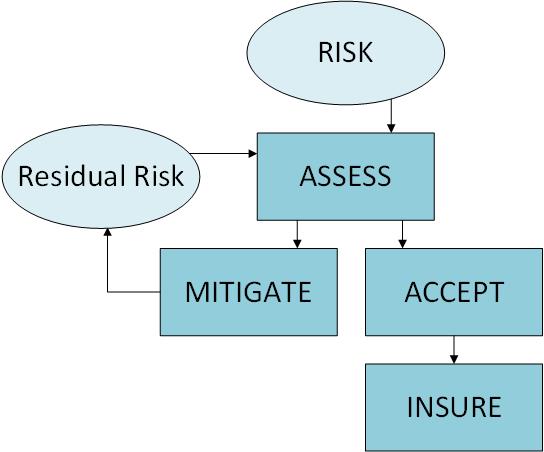
**terminologie: malware**

* virus: malware dit moet uitgevoerd worden
  + infecteert andere bestanden
  + trigger & payload
  + soms polymorf (verandert bij elke infectie) of metamorf (herschrijft zichzelf)
* worm:
  + verspreid zichzelf door actief op zoek te gaan naar slachtoffers
  + multimorf/metamorf
* rootkit:
  + malware die zich in het operating system zelf nestelt
  + erg moeilijk te detecteren
* trojan:
  + malware die zich anders voordoet

## Toreon

**ICT Governance**“IT Governance primarily deals with connections between business focus and IT management. The goal of clear governance is to assure the investment in IT generate business value and mitigate the risks that are associated with IT projects.”

Chief Information Security Officer



“The potential that a given threat will exploit vulnerabilities of an asset or group of assets and thereby cause harm to the organization. It is measured in terms of a combination of the probability of an event and its consequence.”

= *Expected Value of Cost*

**ALE = probability of event \* cost**  
**probability = combination of the perceived threat and measured vulnerability**

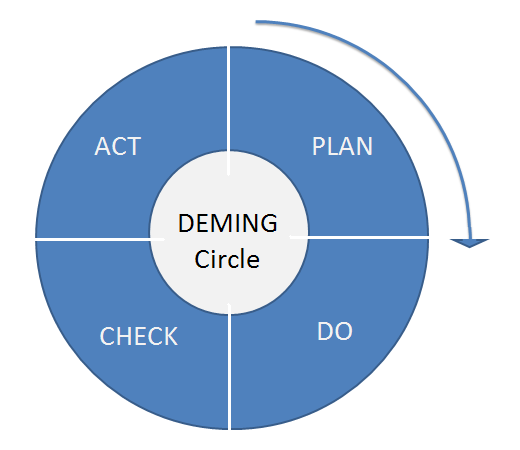
**(ALE: annualised loss expectancy)   
if (cost of countermeasure < ALE) then GO!**

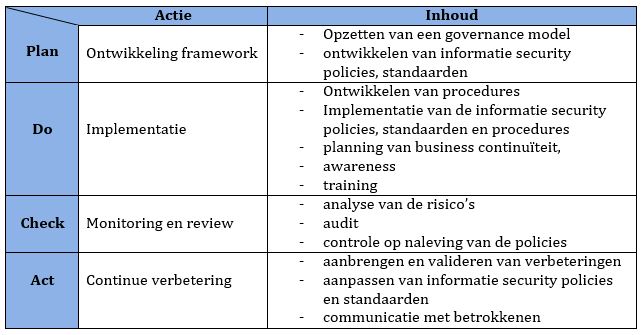
**Problems with Risk decisions..**

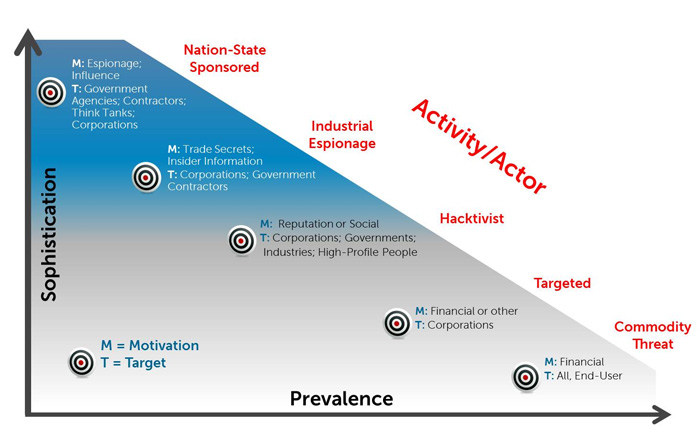
* not enough data (incidents, cost of incidents)
* cost is hard to calculate (reputation?)
* threat moves too quickly
* vulnerabilities can be unexpected and big
* problem with rare, expensive events

Many vulnerabilities are obvious:

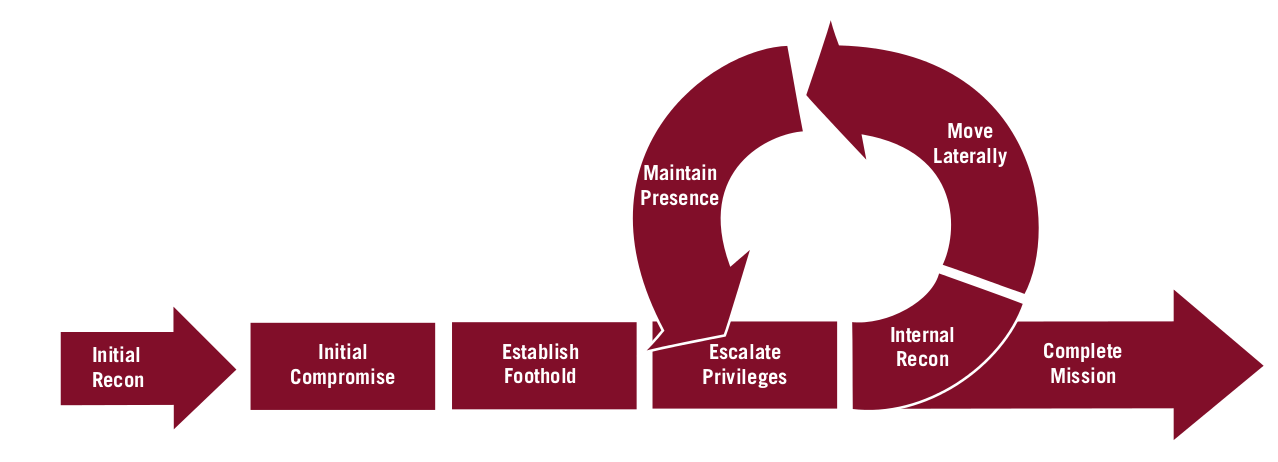
* no patching of OS or apps
* bad user management/pwd mgmt/admin-user mgmt.
* bad perimeter design (e.g. firewall setup)
* flat networks
* lack of auditing and logging
* no technical vulnerability checking, appsec testing
* bad user and ICT admin habits
* remote access with passwords (no strong authentication)
* easy 3rd party access to inner network

**Information Security Management System (ISMS): Deming Cycle**I: (ne cirkel i.p.v. een 3-hoek)



**Advanced Persistent Threats APT**

|  |  |
| --- | --- |
| Advanced   * + Social Engineering   + Malware (known and 0-day)   + Phishing campaigns   + Exploitation of vulnerable applications (web and others) | The goal is to have long-term access to the target  “Low and slow” approach   * + No regular or predictable attacks or communications   + Stay under the radar   + Be patient   “I’ll be back!”  An average advanced attack persist around 243 days before being detected. (Source: Mandiant) |

**APT ‘Kill chain’**

**Initial compromise: methodology**

Phishing mails  
Generic (“Low hanging fruit”)  
 Widely usable subject and message content (ex. “Invoice Notice”)  
 Simple malware attachment or URL to malicious site  
 .exe, .scr, … sometimes in a .zip  
 usually detected by AV

Targeted  
 Content and attachment are related to business activities of the recipient  
 Attachment is a document containing exploit code (possibly 0-day)  
 .pdf, .docx, .xlsx, …  
 Possibly undetected by AV  
 Copy of previously sent mail, slightly modified (“this time including the attachment”)

Web-based infection

* Water holing attack
* Drive-by-downloads

Compromise of a (web) application

Other external connections

* Partner connections
* Remote access VPN

**Establish foothold**  
Malware will contact C2 (Command&Control) server to receive further instructions

* Via HTTP/HTTPS, direct or via proxy
* Via email body, attachment, encoded inside images
* Could be any other protocol: FTP, DNS, SMTP
* Encrypted, hidden in headers...

Instructions include

Create files, directories

* Create Remote session
* Launch applications
* Transfer files
* …

**countermeasures**

Restrict external connections to the bare minimum

* But good luck restricting all web and mail traffic!

Use whitelisting for URL filtering

* Only business critical sites and categories are allowed
* Procedure to quickly add sites
* Effective but very time and resource intensive to implement

Block low hanging fruit

* Block certain downloads and attachments types: .exe, .scr, .pif, …
* Traditional antivirus is not completely useless

Some are better at behavior-based detection than others

**Advanced Threat Protection technologies**

* Execute files in a virtualized environment or sandbox
* Look for suspicious behavior

Patch Management

* Java/flash are usually a problem 🡪 delete!

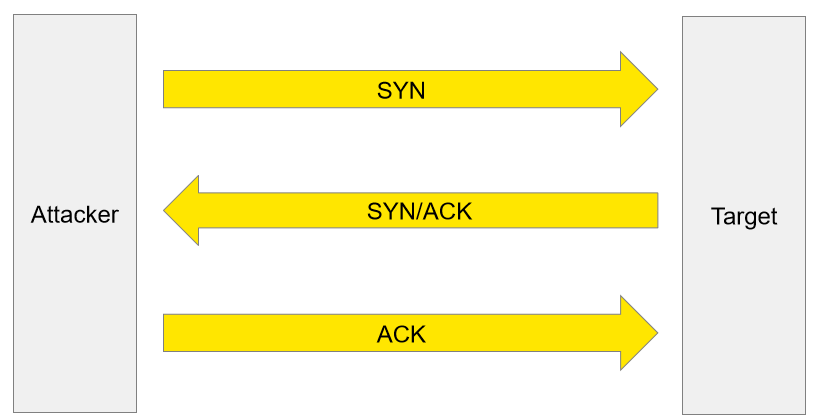
### EY

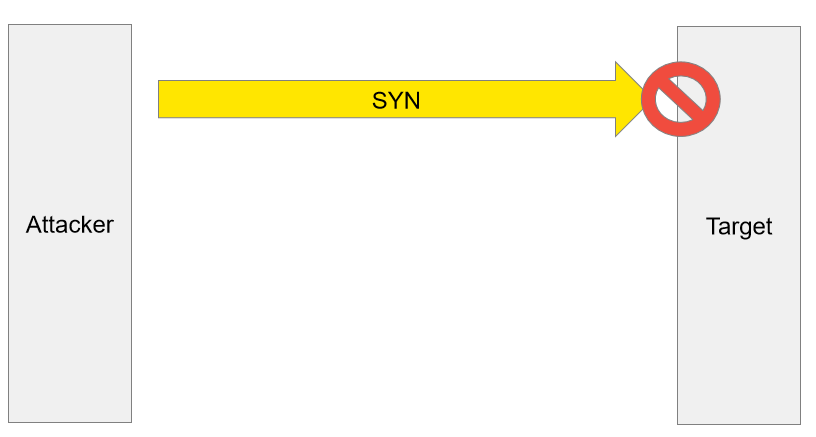
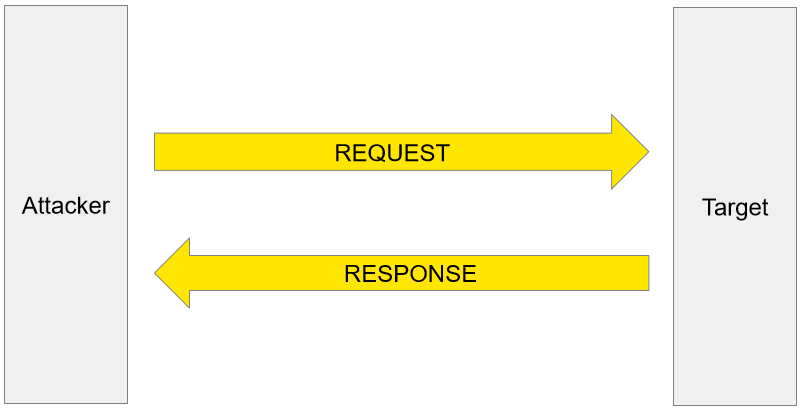
Pentesting is:

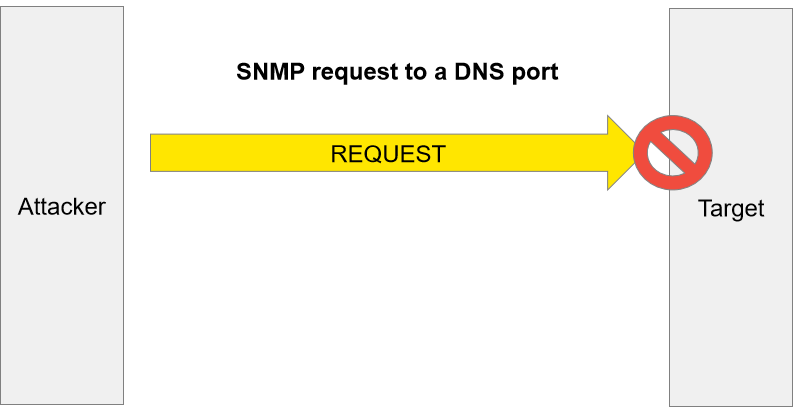
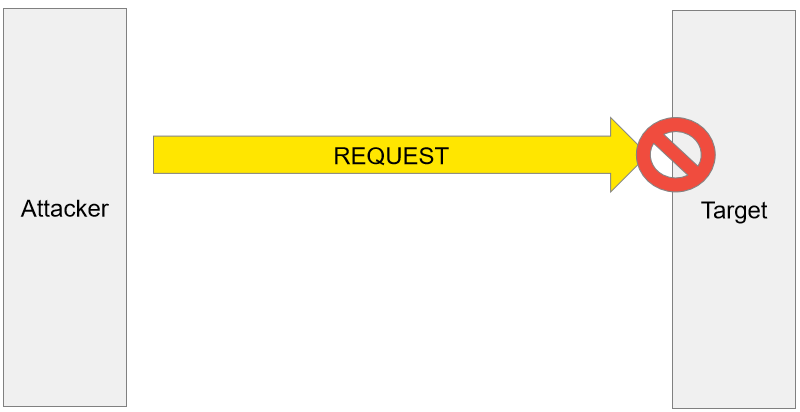
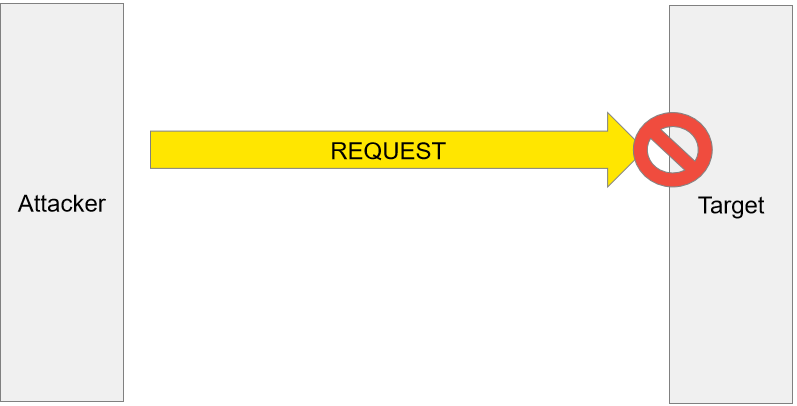
Discovering vulnerabilities in   
► Applications   
 ► Any platform   
 ► Any language   
► Infrastructure   
 ► Anything with any sorts of processing power   
► Reporting these vulnerabilities to the client   
► Help him resolve these vulnerabilities   
► Ultimately increasing his overall security   
► See step 1

White Box ► Receive information from the client ► What is “in scope” ► What can we do and what not

► Black Box ► “Figure it out yourself”

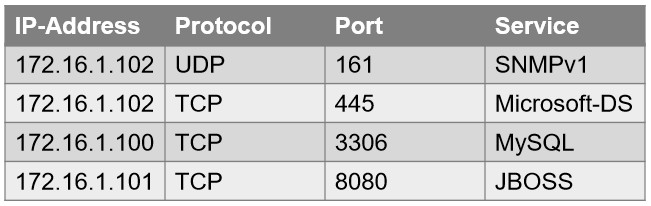
TCP 3-Way handshake TCP port scan: Closed port  
   
 TCP firewalled port UDP Open port

UDP Open port, wrong protocol UDP Closed port  
   
 UDP firewalled port  


NMAP

Quick checks to determine if a host is alive  
► Nmap default ► ICMP Echo Request ► Connections to TCP 80 & 443  
► Can be customized ► Additional ICMP types, TCP & UDP ports, protocol flags,…  
► Timesaver ► Try networks with 26000+ potential hosts

Uit demo:  
SNMP ► Simple Network Management Protocol ► Provides information about configuration settings of a system ► v1/v2 are plain text! ► Default “community” strings ► See it as a password ► “public” ► Usually read-only ► “private” ► Can provide read-write

► Metasploit has modules to do this

► JBOSS ► JavaBeans Open Source Software Application Server ► Application Server  
► History of vulnerabilities ► Especially default configurations! ► Unauthenticated management console access ► Allows us to deploy malicious Web ARchive (WAR) files  
► Leverage metasploit modules  
► Exploitation of identified vulnerabilities  
► Communicate with client ► System could go down ► You are probably testing in production! ► Be very careful with this!

Meterpreter: ► Dynamically extensible payload  
► Can be included as payload for your exploit  
► Provides a complete interactive environment ► Build-in commands ► Load additional modules (e.g. post-exploitation) ► Save time

► Let me introduce “Pass The Hash” ► Use the hash to log in as “local Administrator” on a different server

Mimikatz?► Windows needs to do single-sign on ► Means also HTTP Digest authentication ► Does not accept LANMAN/NTLM password hashes ► Passwords are stored somewhere in-memory

### Social engineering

Social engineering : psychologische manipulatie van mensen om ze acties te laten uitvoeren of (confidentiële) informatie vrij te geven.

LND: Lui • Nieuwsgierig • Dom / Naief / Onfeilbaar

Design fase (Professionals zijn hier de oorzaak)  
– Auto – ARP – SMTP – Buffer Overflow   
Uitvoeren / Gedrag   
– Helpdesk medewerker – System Engineer – SCADA(Supervisory Control And Data Acquisition)

Helpdesk medewerkers zijn van nature hulpvaardig • Hun functie is helpen – antwoorden geven • Getrained om vriendelijk te zijn • Niet getrained om de echtheid van elke oproep te controleren – Minimaal tot niet getrained rond security – Job zonder veel andere uitdagingen dan : oplossen van het probleem – Objectief : zo snel mogelijk oplossen en afhandelen • Goudmijn voor de social engineer

Fingering the Mark

Verzamelen van zo veel mogelijk gegevens om het verhaal dat we gaan ophangen te omkaderen , om authentiek en geloofwaardig over te komen • Lijsten – namen - functies – email adressen - telefoonnummers – gebouwen – computersystemen • Via – whois – De website van het bedrijf – Google – Documenten op papier

**Phising – Vishing – Spear Phising**   
– Boodschap zodanig verpakken dat target gevoel krijgt iets legitiem te doen, dikwijls in combinatie met malware   
– Angst, bedreiging maw sense of urgency om target snel te doen handelen   
– Meestal gericht op verzamelen van logon credentials   
– Hoe meer verrijking hoe beter   
**Pretexting**   
– In scene zetten van een situatie die geloofwaardig genoeg is om target tot handelingen te laten overgaan   
– Niet gebasseerd op angst of bedreiging eerder op het krijgen van vertrouwen   
– Kan telefonisch of met het juiste pakje ook fysiek en ter plaatse  
**Baiting**   
– Handeling uitlokken door target iets voor te houden • Muziek of film download • Foto • USB stick • DVD   
– Gebasseerd op de nieuwsgierigheid van de target   
**Quid Pro Quo**   
– Handeling uitlokken door target iets terug te geven • Doe dit even voor mij en dan krijg je....   
– Gebasseerd op ....het willen hebben van iets (Naief)