Op examen vragen als: waarom dit. Waarom niet dat. Wat als je anders.. . waarom ooit die keuze gemaakt,...

Wat is Security? Authenticatie, encryptie, social media (dingen moeten afgeschermd zijn, privacy, cybercrime

Maar security is tradeoff met user friendly design

Waarom targetten ze gewone mens? Asbotnet, advertisements, mining, pc’s gebruiken voor bv DDOS

Zelf met goede beveiliging

* Kan user nog domme dingen doen door weinig kennis, incompetentie, paswoorden sharen nondigitaal,

Cyber criminality: duidelijke impact op echte wereld  
Cyber warfare: ook fysieke impact bij zulke activiteiten. (mensen fysiek aangevallen erdoor)

C02 Basic concepts

# Security Goals

Altijd per security measure in latere hoofdstukken aftoetsen (testvraag)

## Confidentiality / vertrouwlijkheid

Data kan enkel worden gelezen door juiste personen

Via secret key wordt data geëncryped voor het onleesbaar te maken voor Carol.

Traffic-flow confidentiality: Via privacy enhancing techniques (PET). Er mag niet geweten worden dat er gecommuniceerd wordt (bv. Tor)

## Authentication

Verschil met identificatie: identificatie is meer zoals oogscanner/stemherkenning. Via uw email te geven weet een website dat dat een goede authenticator is voor je binnen te laten. Maar het kan nog niet uw identificatie (wie je echt bent) garanderen.

Authenticatie gebaseerd op:

-Entity authentication: collectie van data (bv email, id)

-attribute authentication: Bv een dokter(een karakteristiek) heeft meer privileges dan andere.

-data-origin authentication: komt van juiste bron? Shit manier via message signature op basis van secret info die enkel alice heeft.

## Acces control / authorization

Welke gebruiker welke bronnen gebruiken.

## Data integrity

Ontvangen data = verzonden data. Kan gedeeltelijk via sequentienummer

## Non-repudiation

Zender kan niet ontkennen bericht verstuurd te hebben. Ontvanger kan niet ontkennen bericht gekregen te hebben.

## Availability

Syteem is beschikbaar voor geautoriseerde gebruikers.

# Security threads

Passief: Eavesdropping Traffic analysis

Actief: message insertion/modification impersonation

Replay DoS Hijacking( A stuurt naar C en C stuurt naar B)

Mogelijke aanvallen:

* Brute force
* Cryptanalysis: op basis kennis algoritme
* Side channel attack: fysieke factoren

Onvoorwaardelijk veilig: niks bereikt dit

Computionally secure: kost encryptie > value informatie  
 duurt langer om te cracken dan dat informatie geldig is

# Security meachanisms

## Encryption

Symmetrisch: Secret key gebruikt voor zowel encryptie als decryptie

Assymetrisch: Private key voor elke user zelf. En een public key voor iedereen voor te versturen. Kunt niks weten over private key op basis public key. Dan decrypted bij ontvangen met private key

Examenvraag: assymetrisch is veel trager. Dus niet voor bericht zelf te encrypten ma wel voor de sleutel te encrypten. Dus enkel voor authenticatie en sleutels en nooit data.

Data confidentieel != integriteit kan zijn dat hij kan wijzigen maar niet lezen. Dus niet verzekerd indien confidentieel

Encryptie (verzekert🡺data integriteit en authentiteit)

# Hash functies

Eerste schema met hash. Confidentialiteit niet verzekerd. Maar data integreiteit wel

Input messages of any length: bepaalde protocollen altijd bepaalde length headers. Mag je er niet uit kunnen afleiden

Hash moet identiek zijn. Mag geen ander bericht zijn me zelfde hash

Hash moet snel te berekenen zijn?  
**reden wel**: verkeer niet tegenhouden. Maar makkelijker te cracken  
**redden niet**: anders kan je snel waardes beginnen berekenen en vergelijken indien hashes gelekt zijn.

## Message authentication codes (MAC)

Cryptographics checksum = hash functie met ook geheime sleutel als input.

+: data integriteit op packet niveau(niet flow). Nu wel confidentieel.

Network and communication security

Ge moet ni persee router hacken. Ge kunt kabel doorknippen en hub tussensteken. (bij possible intruders schema)

Slide 5:   
Q: op welk van die lagen best beveiligen (bv encryptie.) ?   
A: Niet enkel transport laag want niet alles via TCP. Soms praten op transport niveau met andere methodes die geen encryptie hebben. DATALINK laag ook beveiliging nodig zodat niemand kan inloggen daar. En zien dat niemand ne plug kan insteken bij data link. Dus eigenlijk alle lagen beveiligen.

SSH => op applicatie laag en dus boven transport (wordt vaak verward).

# SSH

Q: Wordt gebruikt voor routers ook maar die hebben geen applicatie laag. Hoe dan?   
A: Routers volgen OSI laag niet strict.

## Architectuur

**Transport layer**: server authenticeren, confidentialiteit en integriteit.

Q: Waarom protocol en softwareversie meegeven.   
A: Ssh2 niet compatible met eerste. Soms komen bugs in software want kan zijn dat server of client weigert te communiceren met oudere versie.

## Packet

Aan payload random padding toegevoegt: cryptoanalyse moeilijk maken. Anders voorspelbaar bij kleine pakketjes. + veel encryptiealgoritmes hebben vast aantal bytes

MAC code voor integriteit toegevoegd. + sequentienuummer

Q: Kan ontvanger te weten komen op basis dit bericht(slide 16) welk sequentienummer gebruikt werd?  
A: Nee, want enkel voor MAC gebruikt en MAC is irreversible. Anders moest je dat ook encrypten

Man in the middle attack voorkomen: met private sleutel handtekening doen me alle voorgaande berichten vo te zien dat door hem gebeurde en ni iemand anders

## User authentication protocol

methodes

* Public key: eenvoudigst. Maar computationally complex
* Password: in plain tekst sturen mag. Waarom? Makkelijk te onderscheppen? Nop. Want TODO
* Host based

Slide 28: kanalen interessant want je kan instellingen aanpassen.

## SSH tunneling

**Wanneer?**

* Als je insecure TCP connectie hebt met bv oude email client.
* Om network restricties te bypassen.

**Local portforwarding**:

* lokaal op uw pc verkeer onderscheppen
* naar externe partij via SSH sturen. dus vanaf werk waar geblokkeerd wordt stuurt ge naar uw huis.
* uw huis ga naar yahoo. (poort 9001 wordt dan dus gehijackt door uw huis en ga naar yahoo en terug)
* Wat in uw browser dan intypen voor naar yahoo te surfen? => localhost:9001
* [Home:80 ipv yahoo.com:80 indien je naar iets van uw huis zelf wilt]

Q: Wat als poort 9001 geblokkeerd is?

A: Dan kunnen we niet SSH’en naar home. Oplossing: Remote port forwarding.

Dynamic port forwarding: voor om het even welke site

(Op examen goed die SSH commando’s bestuderen.)

# Exchanging keys

## Diffie hellman

Examenvraag: Assymetrisch encryptie waar gebruikt? Voor sleuteluitwisseling: niet juist. Want gebruikt meestal diffie hellman. Juist antwoord: TODO

Slides Example:

Publieke sleutel: 56%23 Alice: 8   
 Bob: 19   
shared key: 819%23 2

Examenvraag: leg delen van een diagram uit (nooit volledig) bv. Die key-exchange diagrammen

Les 4

Q: Why Is this useful? (slide 37)  
A: Zodat CA niet teveel requests moet afhandelen

## X.509

Slide59: gwn wa uitleg over die attributen

Wordt niet meer uitgebreid, niemand is nog geïnteresseerd in PKI (Public Key Infrastructure) omgevingen.

**Revocation** (certificaten terugtrekken) wordt niet altijd goed afgehandeld. Op certificaat staat ook metadata zoals naam, email,… dus dit is ook verloren bij het terugtrekken van een certificaat.

(bv wijzigen naam van bedrijf -> nieuw certificaat aanvragen, dus heel het process opnieuw)

#### Beperkingen

* certificaten kunnen te groot worden (meer overhead bij het ophalen van deze certificaten)
* Wordt niet door alle software ondersteund

Beperkingen overkomen:standaardprofielen (bv PKIX en FPKI). bijkomend probleem => teveel verschillen tussen profielen (geen compatibiliteit)

# TLS/SSL: origins

Hoe kunnen we op transportniveau beveiligen?

## TLS (Transport Layer Security)

* Ontworpen voor HTTP verkeer, dus wordt HTTPS.
* Een vroegere versie was SSL (Secure Socket Layer), ontworpen by Netscape.
* Hedendaags wordt TLS gebruikt en gebruikt poort **443**.
* SSL is een meer algemene naam waaronder TLS valt.

Het verschil met SSH:

|  |  |
| --- | --- |
| SSH | TLS/SSL |
| Ontwikkeld als applicatieprotocol | Ontwikkeld als transportprotocol (voor TCP) |
| Eigen format | X.509 certificaten |

**TLS Connectie =** ??

**TLS Sessie =** metadata over de TLS connectie. Is handig bv indien connectie interrupt wordt (computer wordt uitgeschakelt ofzo), dan kan de connectie verdergezet worden via het sessiebestand.

Vooraleer we kunnen communiceren moeten de protocollen afgesproken worden. Het handshake protocol gaat 1 byte doorsturen:

* messagetype
* 3 bytes dat de messagegrootte vastlegt
* dan het bericht

#### Fase 0

* Synchronisatie tussen Client en Server.
* De eerste aantal berichten zijn TCP berichten en zijn perfect afluisterbaar/wijzigbaar
* dus data-origin is niet gegarandeerd

#### Fase 1

* Client en server wisselen versies uit van de mogelijke veiligheidsmogelijkheden(bv welke encryptiemethoden kunnen gebruikt worden).
* De client stuurt een **client\_hello** bericht en de server stuurt een **server\_hello**¸dit bevat ongeveer dezelfde informatie als een client\_hello bericht. Het server\_hello bericht bevat ook de geselecteerde algoritmen, of kan de sessie weigeren.
* Eventueel bevatten deze berichten ook info indien er een sessie verder gezet moet worden.
* Hier zijn de eerste aantal berichten ook TCP berichten, maar daarna zijn het TLS berichten.

Zo een bericht kan vrij groot zijn aangezien alle securityalgoritmen moeten gespecifieerd worden. (lijst op P14 niet snappen, wel weten of het een assymetrisch of symmetrisch protocol is, kan gevraagd worden op examen)

#### Fase 2

* De server stuurt een certificaat naar de client.
* Er kan **optioneel** gevraagd worden aan de de client om certificate request te geven
* Ook kan er gekozen worden door de server om eventueel gegevens van de sleuteluitwisseling te sturen naar de client.
* Op het einde stuurt de client een **server\_done** bericht.

#### Fase 3

* De client moet zijn sleutel doorsturen naar de server.

**Q : Waarom is client key exchange verplicht en niet de server key exchange ?**    
A : Heeft al server key van vorige fase etc

#### Fase 4

Q: Waarvoor dient **change\_cipher\_spec ?**A :De server moet de authenticatie van de client controleren en een sleutel kan wijzigen tijdens een sessie. Dit bericht zal de overgang van de ene sleutel naar de andere sleutel optimaal maken.

Het Change Cipher Spec Protocol kan slechts 1 byte versturen. **todo**

Het Alert Protocol kan slechts 2 bytes bevatten. 1 byte : type alert 1 byte: description

**Q: Wat is de beste grootte voor een buffer (slide 22)?**    
A: Best wijzigen afhankelijk van de connectie. Begin met kleine fragmenten bij trage verbinding en gebruik grotere fragmenten bij snellere verbindingen.

Q : Wat is anders tegenover SSH op schema op slide TODO  
A : De MAC wordt bij TLS **VOOR** de encryptie toegevoegd. Bij SSH is dit omgekeerd, eerst wordt geëncrypteerd, daarna wordt het MAC toegevoegd. Het ene systeem is niet beter dan het andere, ze zijn algemeen even veilig.

Slide 23. Voor ieder packet wordt er een TLS Record Header toegevoegd.

Slide 24. De gedeelde sleutel (384 bits groot) wordt afgeleidt van **pre\_master\_secret** (pseudorandom generator). Dit wordt gegenereerd door een client

# Heartbeat

Extensie voor TLS met het **heartbeat protocol**. Het doel van dit protocol dient eerder voor mobiele toepassingen, waarbij sessies vaker uitvallen. Er zijn twee berichten:

**HeartbeatRequest:**

**HeartbeatResponse:**

Gaf aanleiding tot het **heartbleed bug.** Dit protocol werd fout geïmplementeerd door openSSL. De data in de server checkte niet de lengte van het bericht, maar gebruikte lengte die nog in het geheugen van de server zit. Bv client can berichtje van 4 bytesdoorsturen, maar eigenlijk is het 20 bytes. De server zal dus 16 bytes van zijn geheugen terugsturen (servervariabelen, …). Er was ook geen logging, dus er kan niet achterhaald worden wie dat de requests stuurde.

**Q : Hoe vergelijken SSH en TLS nu?**   
A : TLS is iets efficiënter aangezien er minder berichten worden verstuurd en zijn ook volledig geauthenticeerd (met behulp van certificaten). De pakketlengte van TLS wordt wel niet geëncrypteerd. Er is **geen** goede reden om **GEEN** TLS te gebruiken

Slide 36 , nog eens een overzicht van al de fasen

nadeel TLS: roundtrip time tussen client en server (heen en weer communicatietijd), vooral in het begin van een TLS connectie. **5** manieren om dit te verbeteren:

1. Gebruik maken van sessies
2. de client veronderstelt dat de server wel akkoord zal gaan ( de client voert zijn cipher\_change\_spec eerder uit dan de server).
3. gebruik van lokale proxies. Een client moet dan enkel met die proxy praten. De verbinding tussen de proxy en de client is in het beste geval ook beveiligd met TLS. Proxy kan enkel generieke data bewaren, niet user specifiek. Dit wordt heel complex in de praktijk.
4. In het begin van een TLS connectie de fragmentgrootte klein maken, zodat de verbinding snel kan gelegt worden.
5. Stuur niet enkel eigen certificaat, maar ook het certificaat dat het getekend heeft. Dit bespaart een DNS lookup, een request naar de server enz. Best wel dat de certificaatlijst niet te groot wordt.

Les 5

# Ipsec

* meestal vpn network genoemd.
* Network niveau bv. IP encryptie, sourcespoofing (andere ip gebruiken), packet replay
* data integreigeit en confidentialiteit garanderen op IP laag.

LAN to LAN: 2 netwerken op gespreide locaties. Maar willen zelfde netwerk kunnen gebruiken veilig.

Client to LAN: client kan zelf ipsec verbinding opzetten. Alsof hij deel maakt van dat subnet. Maar voor andere dingen gebruikt hij zijn normale dingen

Eigenlijk geen VPN. Maar woord wordt vaak veel gebruikt.

Voordeel bedrijf:

* Applicatie onafhankelijk
* Moet maar paar acces points configureren
* Transparant. VPN zelf opzetten is moeilijk

Nadeel:

* Heel complex, heel wat features en incompatibel tussen verschillende versies.

## Gebruik

User wilt communiceren naar pc in privaat subnet.

Bericht w in nieuwe header gezet. Gateway haalt da eraf en doet naar juiste pc.

Mode 1: layer 2 tunnel mode  
Mode 2: transport mode: Geen IP aanmaken maar alle IP authenticeren/encrypten

Verschillen met andere protocollen: die modussen en headers

SSH: voor korte connectie. TLS tussenin. IPsec voor langdurige verbindingen (dagen,maanden,jaren)

# VPN

* Voor iedere veilige tunnel van site naar site.

Examen: Ipsectrace matchen op voorgaande protocollen: verschillen interessant

# Data link layer

Waarom datalink layer ook beveiligen: stel u voor switch. Automatisch packetten routeren naar juiste poort. Content address memory houd da bij, beperkte hoeveelheid memory. Constant valse MAC adressen sturen dus switch blijft toevoegen en kan dan niet meer toevoegen en dan ??? en dan maak je van switch een UP en kan je afluisteren. Of man in the middle attack.

Hidden node attack: bij wifi weet je nooit of je mag sturen. A wilt naar B sturen en C wilt naar B sturen. Bij wifi ga ni omda je ni kan zenden en ontvangen tegelijk en je weet ook ni of bij ontvange ere en collision is. Oplossing: request to send: heel kort packet, als b da ontvangt stuurt hij agreed naar A. dan weet C da ie ff moet wachten. C kan nu constant RTS berichten sturen. Vo te reserveren voor onbestaand persoon. Dus C is jammer die niet moet jammen (ma om de zoveel tijd opnieuw sturen) dus moeilijk te detecteren.

WEP is slecht, heel onveilig. Makkelijk te kraken

WPA2 is the way to go

EAP: voorbeeld gebruik: tijdelijke paswoorden geven aan vrienden vo op uw network te gaan thuis ipv uw netwerkkey te geve

# Firewalls

Les 6

# Encryptie

Steganografie: verbergen info door mengen andere info

Cryptografie: dan ziet iedereen de info direct maar is het encrypted

In foto verborgen foto: met least significant bit: want die zien we toch niet goed. Interessant voor watermerk to te voegen (via ruis)

# Substitution ciphers

Alfabet verschuiven

Niet veilig: grote sleutel maar in menselijke taal heel wat redunantie/voorspelbaarheden. En niet elke letter even veel gebruikt. Sommige combinaties worden ook vaker gebruikt

Entropie: lage entropie=dan ga je veel te weten komen

Voorbeelden met probability  
1: Entropie=Ongeveer 0,2-0,3  
2: Entropie=1

Vigenere cipher: beetje lek symmetrische encryptie: iedereen kent die schijf maar de sleutel moet je zelf onderling beslissen.

Digraph

1. 5X5 bord maken
2. Ge schrijft message in
3. Rest vul je aflabetisch aan
4. I/J combo als 1 symbool
5. Letters opdelen in groepjes van 2 en dan in bord kijken. 2 na elkaar wordt placeholder ‘X’ bv
6. Beide in andere rij andere kolom. (diagonaal kijken)
7. In zelfde rij:
8. In zelfde kolom:

# Transposition cipher

Letters niet vervangen maar volgorde veranderen.

Rail fence cipher

1. Rij en kolommen maken
2. Diagonaal schrijven eerst
3. Dan per rij kijken voor wa je doorstuurt
4. De sleutel is aantal rijen

Columnar transposisiton cipher: (foutje in slide ergens daar)

1. In rijen alles schrijven
2. Dan kijk je naar kolommen. Eerst 3de kolom, dan 4de kolom,...

# combination ciphers

combinatie van beide

# modern crypotography

block cipher: datablock 8-128 bytes anders padding

stream cipher: je wilt niet wachten tot block klaar is en stuurt al stukken door

feistel encryption: per ronde aantal bitbewerkingen. F block is publiek bekend. Private grote sleutel wordt opgedeeld in kleine sleutels die per ronde worden gebruikt. Handig voor snelheid want hardware kan dit snel in beide richtingen van F is reversibel. In F block gebeuren substituties en permutaties.   
Diffusion: ook al maar 1 bit verschilt in plain. Dan mag je na cipher geen verschil meer zien.  
Confusion: Je mag op basis van cipher ook niks kunnen afleiden vd sleutel.

# Symmetrisch

## DES (data encryption standard)

Volgorde hier van alles nog eens herschrijven. Leerkracht zegt alles door elkaar

Op examen inzichtsvragen: schets heo substitutie blokken er uit zien en hoe die werken of leg blok uit. Je moet niet des manueel kunnen uitvoeren of in detail alles kennen. Per 1 blok wel ke met de hand kunnen uitvoeren

1. Plaintext input. 64 bit text
2. Dan aantal rondes. Met permutaties
3. Elke ronde subsleutel nodig die wordt ge left shift voor de sleutel te maken voor volgende ronde: 56 bit sleutel als input. (Hardware kon fouten bevatten, en sleutel moest manueel ingegeven worden, daarom pariteitsbits toegevoegd). Tegenwoordig echter geen probleem meer dus 56 bit ipv 64 bit want ze laten die weg.   
   Sleutel werkt op basis van transformatie matrix. Bv eerst bit 50 dan 43,...
4. Left shift: sleutel opdelen in 2 delen en shiften. (zie animatie)
5. PC-2: permutatie
6. Encoding:
7. IP: Eerst permutatie gebruiken (standaard, altijd zelfde)
8. Rnd-1: begint bij expansiebox. 32 bit naar 48 bit want we hebben 48 bit sleutels. Makkelijk combineerbaar zo. Sommige bits worden hier gedupliceerd.
9. Dan sleutel combineren met die block DATABLOK + SLEUTELBLOK = RESULTBLOK
10. Dan S box: opdelen: 8 keer 6 bits: 6 bits telkens input adressen van kleine lookup table
11. Komt maar 4 bits uit zo een adres box
12. Men kijkt hierbij naar eerste an laatste bit voor rij te kiezen (1 . . 1) => 4
13. En dingen ertussen voor de kolom (NOT SURE; nog eens checken)
14. P-Box: nog eens een permutatie die standaard gedefinieerd is.
15. Boxen zijn allemaal zo gemaakt daj geen data verlies hebt.
16. Dan na F functie nog ge xorped met linker blok en dan heb je rechterblok. Nieuwe linkerblok is oude rechterblok.
17. Als je dit allemaal volgt heeft 1 bit verandering invloed op alle andere bits dus kan niets afgeleid worden van dingen dat maar 1 bit verschillen
18. Iedere box 16 keer gebruikt.
19. Elke box makkelijk door hardware te implementeren.

Block cipher mode: grote hoeveelheden data encrypteren efficient.

Electronic code book. Opdelen in blocken, encrypteren en doorsturen. Niet vaak gebruikt want niet veilig: iedere block op veel dezelfde soort data, wordt veel data ontdekt. Parallelisatie is makkelijk, voor snel te encrypten.

Chaining: ook al blokken gelijkaardig. Toch nog ciphertext heel verschillend  
Veel last van bit fouten en parallelisatie niet meer mogelijk want je kan niet encrypten zolang je het vorige niet weet. Bij decryptie parallisatie wel mogelijk

Examenvraag: Bv encryptie van harde schijf: welke encryptie methode beste? Kijken naar => bitfouten ?parallisatie?

s-bit Cipher feedback:

1. Shit doorsluizen
2. Bit error helpt alles omzeep. Dus transmissiemedium moet error vrij zijn
3. Parralisatie encryptie: neen
4. Paralisatie decryptie: nee, want output nodig van het ene vo het andere

Output feedback:  
 klein verschil: in dit geval komt sbit voor xorp. Dus bits komen sneller. Dus minder bit errors en op voorhand kan je keys berekenen.

CTR: goed voor smart card die weinig computional power heeft. Weinig impact bit errors, parallisatie eenvoudig. Nadeel, manipulatie mogelijk. Probleem want headers kunnen zo van 0 naar 1 gezet worden. Of ergens in getal 1 bijzetten voor waarde X10 te doen.

Is DES veilig? Niet veilig: want zie tabel met 56 bit tijd required.

## 3DES

## AES

Verschil tegenover DES: Hier geen beperking blokgroote/sleutelgrootte.

Les 7

# Modern cryprography: assymetric encryption algoritmhms

* Moderner dan symmetrische

# RSA

* Op basis discrete wiskunde
* Om te encrypteren
* Authenticatie

# ECC (elliptic curve cryptography)

Heel goed. Kies je best ajet zelf doet

Wat voor type algoritme is da en waarvoor wordt dat gebruikt: (slide 177) gaat hij vragen, en relatieve voor en nadelen. Niet de wiskundige details 🡺

RC4: symmetrische encryptie algortime, vaak voor confidentialiteit

SHA1: Message authentication

ECDHE; voor sleuteluitwisseling

# Hash

* Vast aantal bits als output
* Modification detection: replay attacks / ip spoofing / ...
* Niet cryptografisch veilig, moet je dus combineren met symmetrische of assymetrische vo zeker te zijn da hash niet gewijzigd wordt onderweg
* Dikwijls hash over groot bericht (grote blokken). Voor betere efficientie en weerstand tussen key attacks (zelfde hash zelfde output als de uwen)

# MD5

Indien mogelijk beter SHA1 SHA2 SHA3 nemen

# SHA1

* Veiliger dan MD5.

# SHA2

* Want SHA1 al oud
* Meerdere versies: afhaneklijk hash grootte

Meest gebruikte hash in netwerkwereld bbinnnen netwerksecurity (nog niet voor wachtwoorden)

Hoe het werkt niet vanbuiten leren. Maar ongeveeeer weten hoe die blokken p125 in elkaar zitten

* Input 1024, output 512
* Iedere blok als input gebruikt
* Initialisatievector in begin per blok
* Vaste inputvector: niet afhankelijk van een concrete sleutel
* In inputvector: eerst data expanderen. Eerste 1024 bits kopie van input. Dan woorden combineren met bitoperaties.
* Die woorden worden dan samengeteld en worden dan bv woord 16.
* En dat een aantal keer tot ge 5120 bits hebt
* Dan 80 keer compressie functie (per 8 bytes)
* (Maj = majority, grootste nemen)
* ...
* Output dan optellen met vorige en output is dan 512 bits

# Sha3

* Websites beginnen langzaam te shiften ernaar
* Browsers veranderen niet graag voor compatibiliteit dus gaat sloom vooruit
* Sponge function: dingen contacteren (absorberen) en verwerken
* Beter dan SHA2 maar het moet ondersteund zijn in uw omgeving. Enkel recentste dingen ondersteunen het maar

# MAC algoritmes

Eerst hash functie en dan encryptie bij hash algoritmes. Nu op plain text MAC functie toepassen.

## CBC MAC

Enkel laatste blokje gebruik je.

Nadeel: veel encryptie ertussen voor dan enkel laatste te gebruiken

## Hmac

Hash functie die sleutel hashed.

Notatie: K||m key staat na message H[X] hash nemen van x

# Software and system security

* Applicatie laag

# Secure applications

## PGP

* Voor email verkeer: authentication, confidentialiteit
* Data op uw pc ook.
* Iedere eigenaar een sleutelpaar: 1 voor handtekeningen, 1 voor encryptie
* Identieit, houdbaarheidsdatum en paswoord nodig voor key te maken
* Kunt zelf sleutel aanmaken op naam van een ander met uw email. Niks controleerd dat
* Plain text wordt gehashed. Dan encrypted me private sleutel alice. Toegevoegd terug aan hash. Dan ge compresseerd. Dan omgezet in RADIX64 (soms wordt witruimte toegevoegd door de mailshit enzo, RADIX64 fixed dat probleem).
* Ontvangen: terug oorspronkelijk formaat, de compresse, hash functie bereknen en decrypteren met sleutel van alice. Als die zelfde is dan weetje da da van alice is en geen wijzigingen op uitgevoerd.
* Dan nog assymmetrische versie voor confidentialiteit
* Beide schema’s combineren voor zowel confidentialiteit en authenticatie
* Grootste probleem: welke sleutel moet gebruikt worden? Iedereen die pgp stuurt geeft zijn geheime sleutel mee. Je zou dan denken da je elke keer diezelfde sleutel wilt gebruiken. Dan automatiseren kan maar niet aan te raden voor ip spoofing.
* Bericht: heeft data, time stamp, file name::: dat deel gehashed. Dan handtekeing erboven met time stamp wanneer die gezet wordt. Dan key toegevoegd van alice voor wie die gemaakt heeft. Dat wordt gencrypt met private sleutel van alice. Dat stuk voor authenticatie. Dan alles gezipped, dan geencrypt met sessiesleutel, en publieke key van ontvanger, dan ook ID van ontvanger. Dan heel dat deel met R64 verstuurd
* Sessie sleutel bovenaan voor overhead met meerdere ontvangers te beperken
* Paszin beter dan paswoord. Cryptografisch veel veiliger (bv geen dictionary attack mogelijk). Typt eenvoudiger ook.
* 1- 1 nog niet 1q 1 nog niet genoeg informatie ,.....
* Pgp.cs.uu.nl
* Revocation indien er probleem was
* Trust model vindt hij interessant

# S/MIME

* S/MIME stuff op traditioneel mail systeem (mime) veilig te maken
* Native support voor email clients, terwijl ge me PGP extra modules moet installeren. Niet alle mail clients comptatibel met elkaar tho.
* MIME zorgt ervoor dat je in jaren 90 ook jpegs enzo kon versturen in mail, maar had geen security.
* Headers > lege lijn > bericht in MIME bericht
* Multipart/mixed bv. Allemaal bijlagen in elk 1 stukje na elkaar
* multipart/parallel moet gelijk afgespeeld worden(bv. Allemaal deel van 1 liedje)

S/mime

* Envelopped data: encrypted
* Signed data: heeft signature. Zodat content niet gewijzigd worden onderweg, maar nadeel: kan enkel gelezen worden door iemand dat s/mime ondersteund
* Clear signed data: standaard signature dus mensen da S/mime ni ondersteunen kunnen het ook lezen
* Signed and enveloped data: combo
* Pkcs profiel: nodig voor settingsstuff te weten (lek bij vorige dingen paddinggrootte enzo)

Q: Welke data kan je als listener wel nog lezen?  
A: content is encrypted, maar recipientInfo niet(als die ooit gehackt geweest is eje klodden). **Berichttitel** kan je ook lezen. Alles voor content type kanje lezen op pagina 35: Mime, from ,to, date ,subject

* S/mime ook bruikbaar voor automatisatie van certificatie. “is deze persoon betrouwbaar”. 🡺 s/mime bericht sturen met vraag over die usernaam+sleutel. Server zegt dan Ja/nee , die is geregistreerd met die x
* Signed receipts: aan ontvanger vragen voor bericht terug te sturen. Zodat zender weet dat het ontvangen werd
* Security labels:
* Secure mailing list: outsourced naar een server ervoor
* s/mime heeft stndaard niet certificaten maar heb je ook attribuut voor.