Authenticiteit

- authenticatie van een boodschap betekent
  - nagaan of ontvangen boodschap afkomstig is van vermeende afzender en niet veranderd is
  - bescherming van de integriteit van de boodschap
  - valideren van de identiteit van de afzender
- wat zijn de 4 alternatieve manieren van authenticeren?
  - encryptie van de boodschap
  - gebruik van een message authentication code (MAC)
  - gebruik van een hash functie
  - gebruik van een *keyed* hash functie (recent)
- tegen welke aanvallen voorziet authenticatie?

Authenticiteit - handtekeningen 1

Waarom authenticeren? (1)

1. onthulling van de boodschap (encryptie)
2. analyse van verkeer (encryptie)
3. vermomming
   1. creatie boodschap afkomstig van “geauthenticeerde afzender”
   2. ontvangstbevestiging door iemand anders
4. wijziging van inhoud van de boodschap
5. wijziging van de volgorde van boodschappen
6. uitstellen of heruitzending van boodschappen

2

Waarom authenticeren? (2)

Non repudiation?

7. ontkenning door afzender dat boodschap van hem afkomstig is
   - digitale handtekening
   - *source repudiation*
8. ontkenning door bestemmeling dat boodschap door hem ontvangen is
   - extra protocol nodig
   - *destination repudiation*

3

Authenticering vereist 2 niveaus

- laagste niveau
  - authenticator = functie
  - geproduceerd bij aanmaak van boodschap
- hoger niveau
  - authenticator gebruikt als primitieve in authentiseringsprotocol bij controle
- 4 alternatieve functies als authenticator
  - geëncrypteerde boodschap
  - message authentication code (MAC)
  - hash functie
  - *keyed* hash functie

Authenticiteit - handtekeningen 4

Encryptie van de boodschap

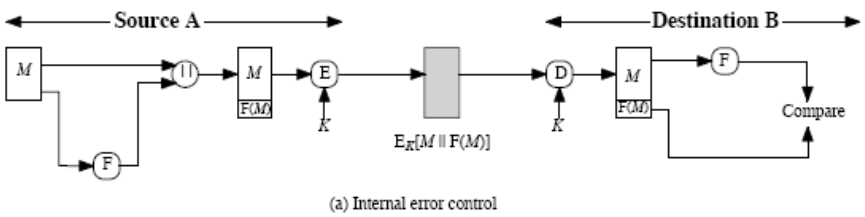
Conventionele encryptie

- boodschap A➔B, gecodeerd met gedeelde sleutel
  - geheimhouding verzekerd!
- is B verzekerd dat boodschap afkomstig is van A?
- ja omdat
  - A en B zijn enigen die de sleutel kennen
  - A is enige partij die boodschap kan gecodeerd hebben
  - B weet dus dat boodschap niet is gewijzigd
- hoe weet B dat gedecodeerde ciphertext=plaintext?
- kan dit geautomatiseerd worden?
- alleen als boodschap aangepaste **structuur** heeft
  - FCS of andere vorm (TCP-pakket, …)

Structuur in de boodschap: FCS

Frame check sequence

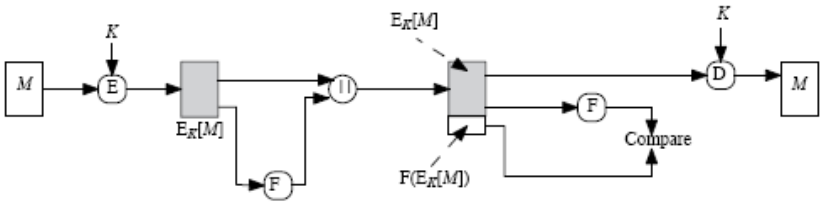
- foutdetectiecode



(a) Internal error control

Plaats van FCS

FCS moet berekend worden vóór encryptie !



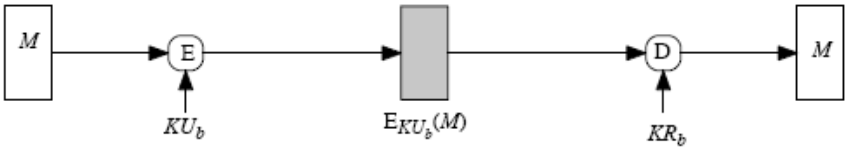
(b) External error control

hier kan gewijzigde ciphertext correcte FCS opleveren!!

Encryptie van de boodschap

Publiekesleutelencryptie

- boodschap A➔B, gecodeerd met publieke sleutel van B
  - geheimhouding verzekerd!
- is B verzekerd dat boodschap afkomstig is van A?
  - neen, want iedereen kan sleutel gebruiken



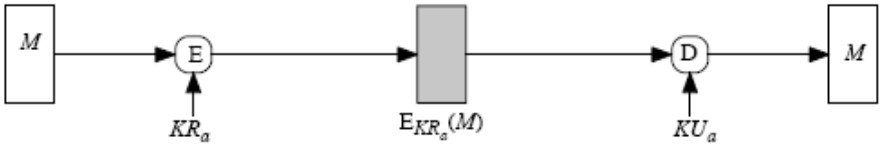
(b) Public-key encryption: confidentiality

Encryptie van de boodschap

Publiekesleutelencryptie (2)

boodschap A➔B, gecodeerd met private sleutel van A

- is de authenticiteit verzekerd ?
- alleen als boodschap aangepaste **structuur** heeft

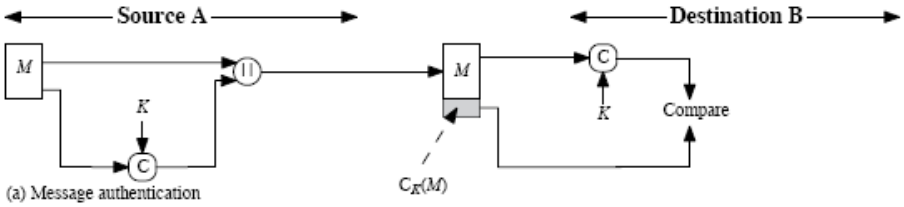


(c) Public-key encryption: authentication and signature

Vier mogelijke authenticatoren

- encryptie van de boodschap
- message authentication code (MAC)
- hash functie
- keyed hash functie

Message authentication code (MAC)

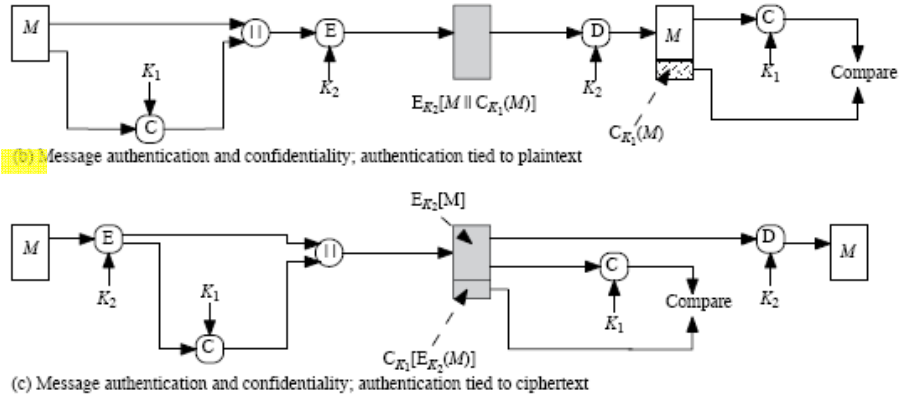


- algoritme genereert klein datablok met vaste lengte
- blok = cryptografische controlesom = MAC
- afhankelijk van boodschap en geheime gedeelde sleutel
- vergelijkbaar met encryptie doch niet omkeerbaar
- blok toegevoegd aan boodschap (handtekening?)
- ontvanger berekent MAC opnieuw en vergelijkt met ontvangen MAC

Message authentication code (MAC)

- correcte MAC verzekert dat
  - boodschap niet is gewijzigd
  - boodschap afkomstig is van afzender
  - boodschap een volgnummer heeft dat het juiste is
- indien ook geheimhouding gewenst
  - bijkomende encryptie noodzakelijk
  - bijkomende sessiesleutel noodzakelijk
  - codering kan voor of na plaatsen van MAC
  - zie volgende figuren

MAC: authentisering en geheimhouding



(b) Message authentication and confidentiality; authentication tied to plaintext

(c) Message authentication and confidentiality; authentication tied to ciphertext

Waarom MAC gebruiken?

in vergelijking met coderen van hele boodschap

aanbevolen in volgende gevallen:

- zelfde boodschap → meerdere bestemmelingen
  - vb. militair controlecentrum
  - 1 ontvanger controleert authenticiteit en verwittigt andere in geval van fout
- authenticiteit van computerprogramma
  - nagaan van MAC op plaintext eenvoudiger
  - vergelijk met decryptie van heel programma
- vergelijkbaar met encryptie doch niet omkeerbaar

Sterkte van MAC (1)

- N→1 functie
  - potentieel kunnen vele boodschappen zelfde MAC hebben
  - dergelijke boodschappen vinden moet moeilijk zijn
- indien geen geheimhouding gebruikt
  - opponent beschikt over
    - **blokken plaintext en bijhorende MAC's**
- onderstel
  - k = sleutellengte
  - n = grootte MAC
  - n < k

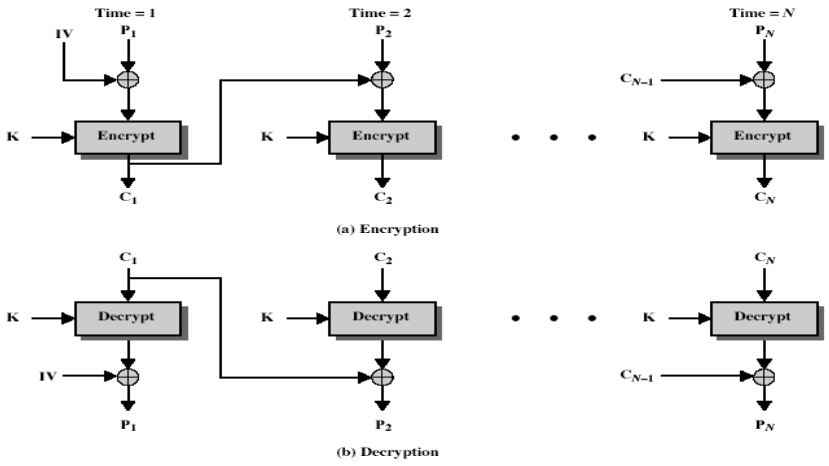
Sterkte van MAC (2)

- gegeven is gekende
  - M_1
  - $MAC_1 = C_{K1}(M_1)$ (met C = MAC-algoritme)
- brute force attack
  - bereken $MAC_i = C_{Ki}(M_1)$ voor alle mogelijke sleutels Ki
- minstens 1 sleutel i garandeert $MAC_i = MAC_1$
- productie van 2^k MAC-waarden (k = sleutellengte)
- slechts 2^n zijn verschillend (n = lengte MAC)
- $2^n < 2^k$ (want n < k)
- meerdere sleutels produceren zelfde MAC-waarde
- welke is de correcte sleutel?

Berekening van MAC

- Data Authentication Algorithm (DAA)
  - FIPS (Federal Information Processing Standards)
  - ANSI standaard
- Boodschap = N blokken D_i van 64 bits
  - D_N aangevuld
- Gebruik van DES (E_K) in CBC-mode (fig. volgende slide)
  - $O_1 = E_K (D_1)$
  - $O_2 = E_K (D_2 \text{ xor } O_1)$
  - …
  - $O_N = E_K (D_N \text{ xor } O_{N-1})$

CBC: cipher block chaining



Berekening van MAC

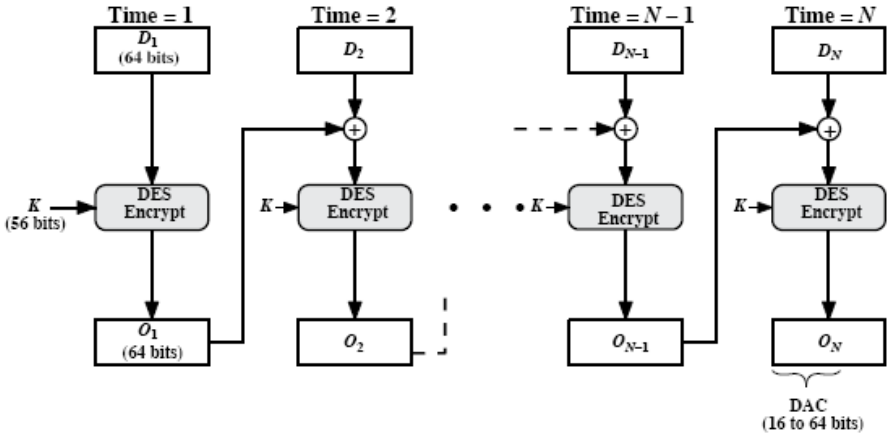


Figure 11.6 Data Authentication Algorithm (FIPS PUB 113)

Vier mogelijke authenticatoren

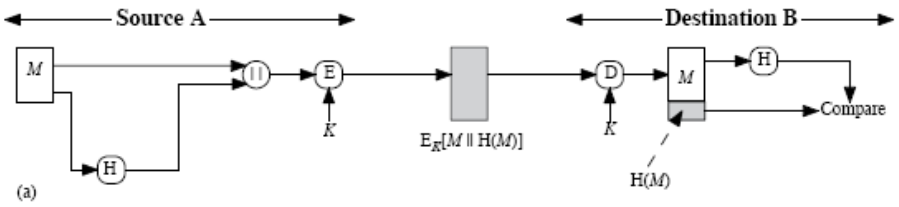
- encryptie van de boodschap
- message authentication code (MAC)
- hash functie
- keyed hash functie

Hash-functie

- condenseert boodschap tot vaste lengte
- functie is publiek en gebruikt geen sleutel
- vergelijkbaar met MAC (doch deze gebruikt wel sleutel)
- 1 bit wijzigen in boodschap ➔ wijziging van hashcode
- hash laat toe om wijziging in boodschap te detecteren
- kan op verschillende manieren worden gebruikt
- bijna altijd gecombineerd met encryptie (waarom?)
- meest gebruikt om digitale handtekening te maken

Hash-functie: gebruik (1)

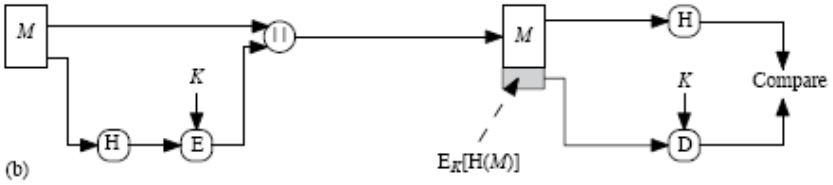
- conventionele encryptie
- hash = structuur die toelaat authenticiteit te controleren (cfr. FCS)
- ook confidentialiteit d.m.v. de sleutel



- waarom is authenticiteit verzekerd?

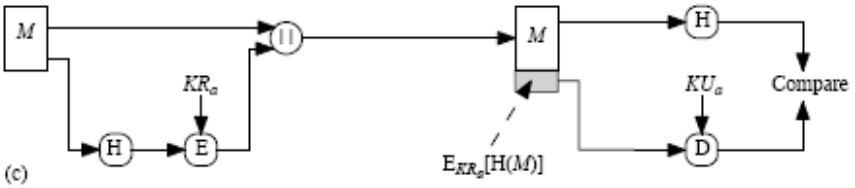
Hash-functie: gebruik (2)

- conventionele encryptie alleen bij hashfunctie
- kleinere verwerkingstijd
- geen confidentialiteit
- vergelijkbaar met MAC



Hash-functie: gebruik (3)

- vergelijkbaar met (2), doch public key encryptie
- hashfunctie gecodeerd met private sleutel van A
- = digitale handtekening
- alleen A kan hash-code geëncrypteerd hebben
- iedereen die publieke sleutel van A bezit, kan controleren

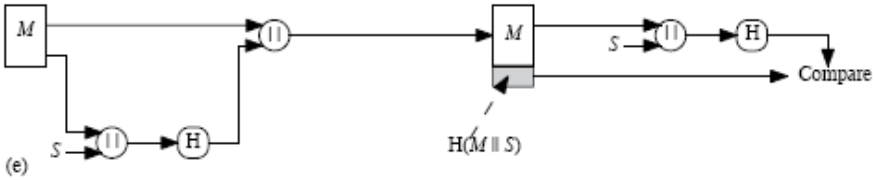


Hash-functie: gebruik (4)

- (3) + conventionele encryptie
- ook geheimhouding

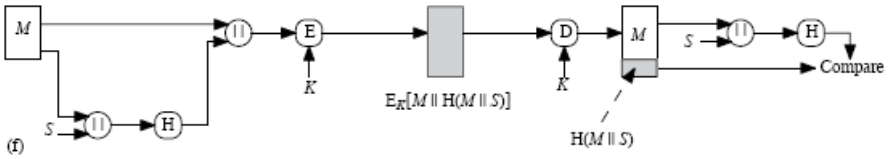
Hash-functie: gebruik (5)

- wel hash, geen encryptie
- A en B delen geheime waarde S (niet doorgestuurd)
- opponent kan boodschap niet veranderen



Hash-functie: gebruik (6)

- (5) + conventionele encryptie
- confidentialiteit



Hash-algoritmen: vereisten

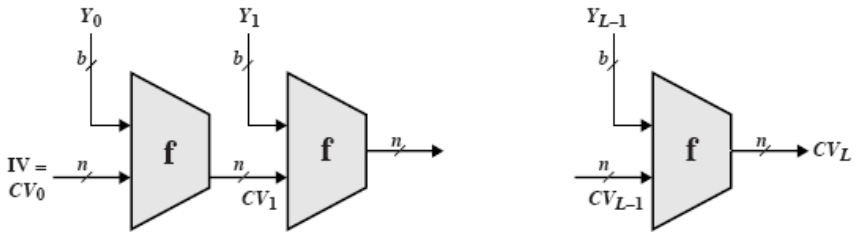
Vereisten voor hash H

- H kan toegepast worden op datablok van willekeurige lengte
- H(x)
  - heeft vaste lengte
  - is gemakkelijk te berekenen (soft- en hardware)
- computationeel ondoenbaar om
  - voor gegeven hash h, x bepalen zo dat H(x)=h
  - voor gegeven x een y bepalen zo dat $y \neq x$ en H(x) = H(y)
  - om een paar (x,y) te vinden zo dat H(x) = H(y)

Hash-algoritmen

- MD5 (message digest 5)
- SHA-1 (secure hash algorithm)
- RIPEMD-160 (RACE Integrity Primitves MD)
- evolueren om zich te verdedigen tegen brute force attack
  - grotere hash-lengten
- eigenschap: gebruiken net als block-ciphers functies die herhaald worden (zie figuur op volgende slide)

Hash-algoritmen: principe



IV = Initial value
CV = chaining variable
Y_i = ith input block
f = compression algorithm
L = number of input blocks
n = length of hash code
b = length of input block

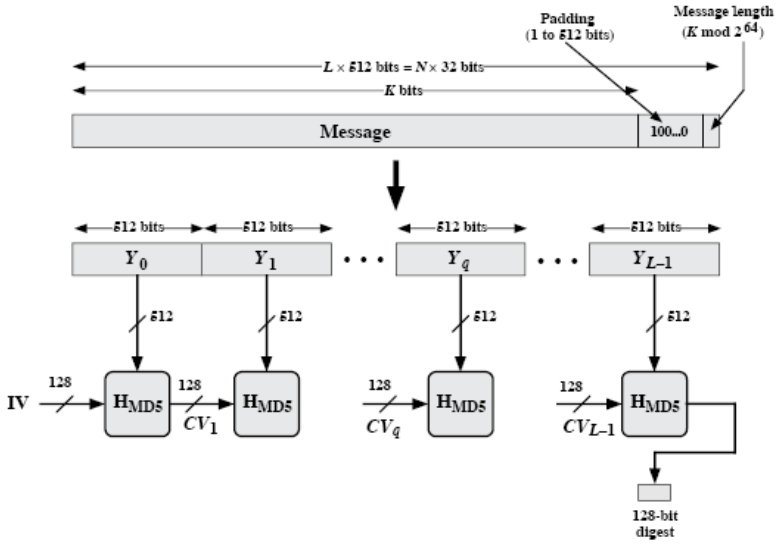
MD5: algemeen

- ontworpen door Ronald Rivest (R in RSA)
- laatste in een reeks MD2, MD4
- produceert een hash-waarde van 128 bits
- tot voor kort meest gebruikt algoritme
- is Internetstandaard RFC 1321

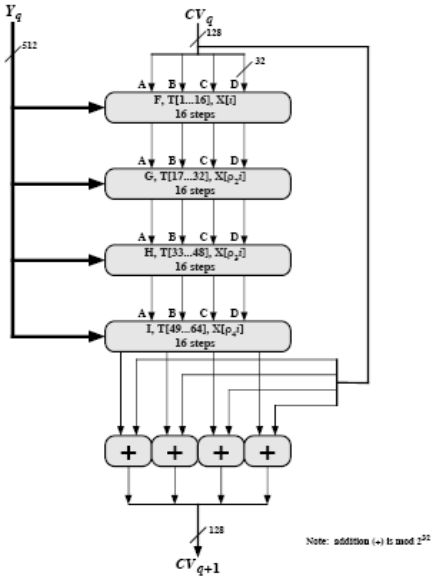
MD5: overzicht

1. boodschap opgevuld zo dat
   lengte = 448 mod 512 (bits)
   - opvulling = 1000…000
   - lengte is 64 bits minder dan veelvoud van 512
2. $(\text{originele lengte van boodschap})_2 \bmod 2^{64}$
   - minst beduidende 64 bits worden achteraan toegevoegd
3. totale lengte boodschap = L x 512 bits
   - L blokken van 512 bits
   - elk blok voorgesteld door 4 x 128 bits
   - elke 128 bits voorgesteld als 4 x 32 bits: A B C D

MD5: overzicht (2)



MD5: compressiefunctie



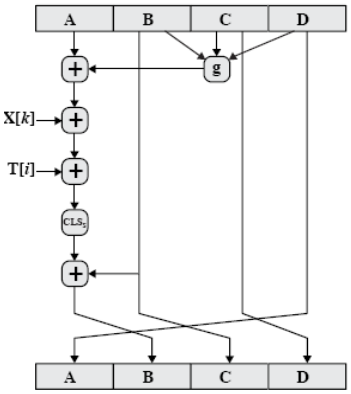
MD5: overzicht (3)

4. 4 registers van 32 bits krijgen initiële waarde:
   - A = 0x67452301
   - B = 0xefcdab89
   - C = 0x98badcfe
   - D = 0x10325476
5. compressiefunctie = 4 ronden

- elke ronde heeft zelfde structuur
- elke ronde heeft andere primitieve functie: F, G, H, I
- elke ronde gebruikt ¼ van tabel T[1..64]
  - T[i] = 2^{32} x abs(sin(i)) ➔ getal van 32 bits want 0<= sin <= 1
  - genereert 64 willekeurige bitpatronen
- na laatste ronde optelling mod 2^{32}

MD5: elementaire stap (16x per ronde)

ronde	fctie	g(b,c,d)
1	F	(b&c)|(!b&d)
2	G	(b&d)|(c&!d)
3	H	b xor c xor d
4	I	c xor (b|!d)



- CLS_S = left shift over s bits
  - s afhankelijk van stapnummer
- X[k] is k^{de} 32 bits woord van het q^{de} 512 bits blok van boodschap
  - 16 x 32 bits = 512
- + optelling mod 2^{32}

Hash-algoritmen

- MD5 (message digest 5)
- SHA-1 (secure hash algorithm)
- RIPEMD-160 (RACE Integrity Primitves MD)

Andere hash-algoritmen

- gebruiken vergelijkbare schema's als in MD5
- SHA-1 (secure hash algorithm)
  - ontworpen door NIST & NSA in 1993, gereviseerd 1995 SHA-1
  - US standaard voor gebruik met DSA handtekeningschema
  - standaard is FIPS 180-1 1995, en Internet RFC3174
  - produceert 160-bit hash-waarde
  - nu algemeen geprefereerd algoritme
  - methode vergelijkbaar met die van MD5
  - uitgebreid naar SHA-256, SHA-384 en SHA-512
- RIPEMD-160 (RACE Integrity Primitves MD)
  - 160 bits hash-waarde

Authenticering: overzicht

- 4 alternatieve functies als authenticator
  - encryptie van de boodschap
  - message authentication code (MAC)
  - hash functie
  - Keyed hash functions: NMAC en HMAC
- MAC
  - berekening van digest op basis van **sleutel** en conventioneel encryptiealgoritme
- hash
  - berekening van digest zonder gebruik te maken van sleutel
- Keyed hash
  - berekening van digest met gebruik van **sleutel**

Berekening van MAC

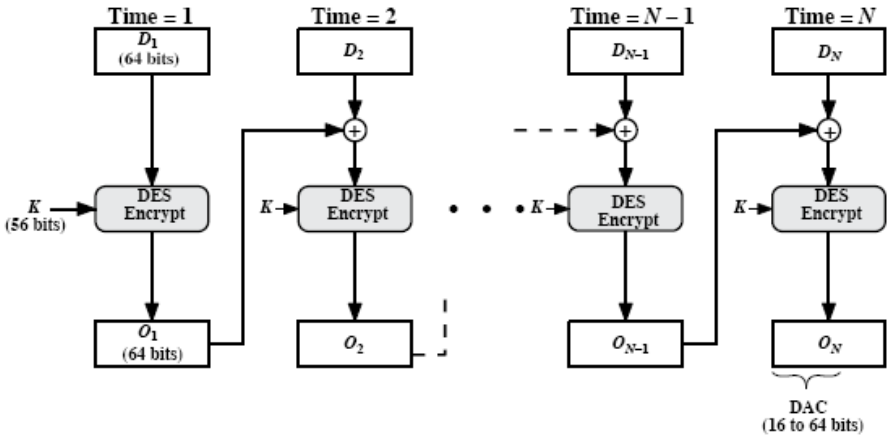


Figure 11.6 Data Authentication Algorithm (FIPS PUB 113)

HMAC versus MAC

- beide maken gebruik van een sleutel
- MAC op basis van
  - Data Authentication Algorithm
  - gebruik van DES
- HMAC op basis van hashfunctie
- Argumenten pro:
  - hashfuncties sneller uit te voeren in software dan DES
  - bibliotheekroutines voor hashfuncties wijd verspreid
  - geen exportrestricties vanuit de US
- hoe sleutel invoeren en toch hash gebruiken?
  - aantal voorstellen, HMAC meest aanvaard
  - RFC 2104
- HMAC gebruikt in IPSec en SSL

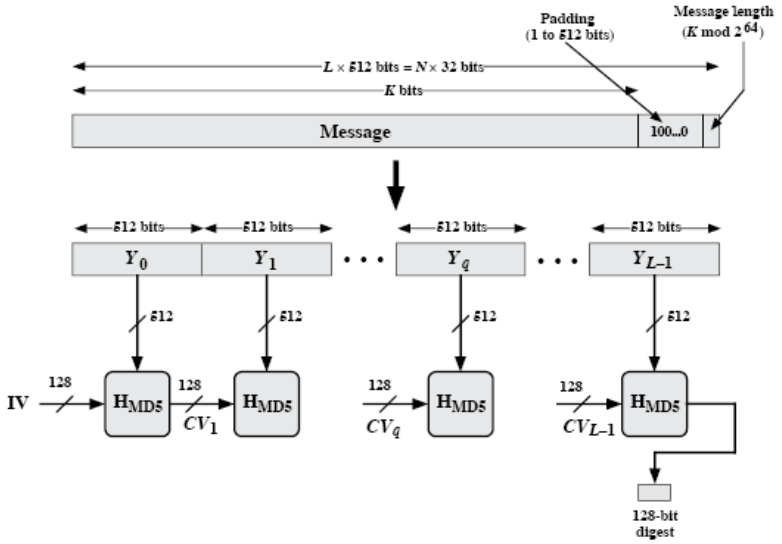
41

HMAC: objectieven

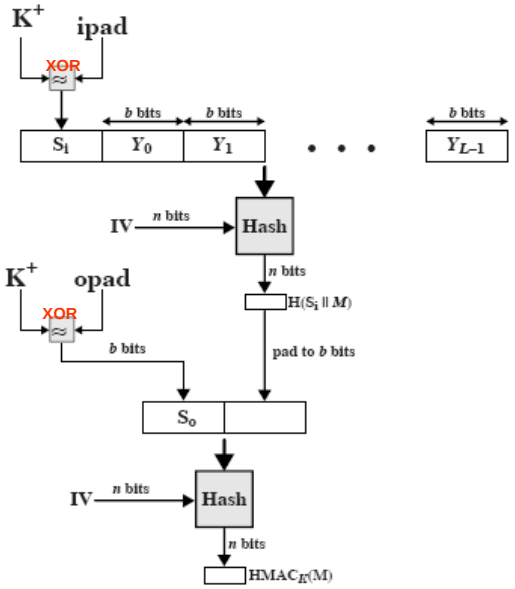
- gebruik van hashfuncties
  - goede performantie in software, vrij bruikbaar
  - zonder aanpassingen
- hashfunctie makkelijk te vervangen door andere
- behoud van oorspronkelijke performantie van hash
- sleutel gemakkelijk te behandelen
- veiligheid vergelijkbaar met die van de hashfunctie
- hashfunctie in HMAC is black box
  - grootste deel van code is direct bruikbaar
  - kan vervangen worden

Authenticiteit - handtekeningen 42

MD5: overzicht (2)



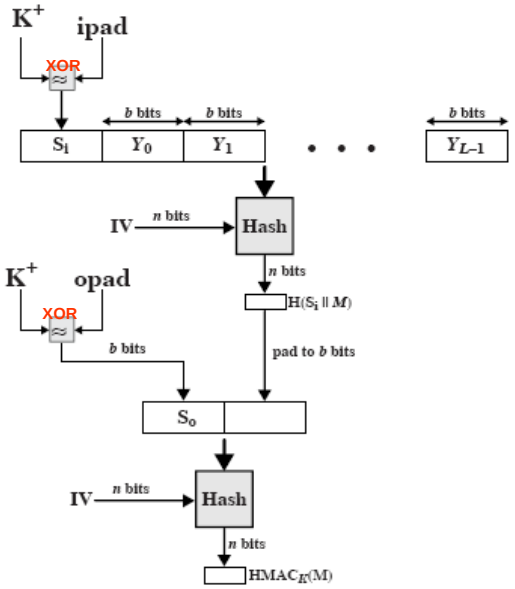
HMAC: schema



HMAC: algoritme

- H: embedded hashfunctie (MD5, SHA-1, RIPEMD-160, …)
- IV: initiële inputwaarde van hash
- M: boodschap als invoer voor HMAC, incl padding zoals voorgeschreven door hash
- b: aantal bits in blok
- L: aantal blokken in M
- Y_i: ide blok van M, 0<= i<L
- n: grootte van de digest geproduceerd door hashfunctie

HMAC: schema



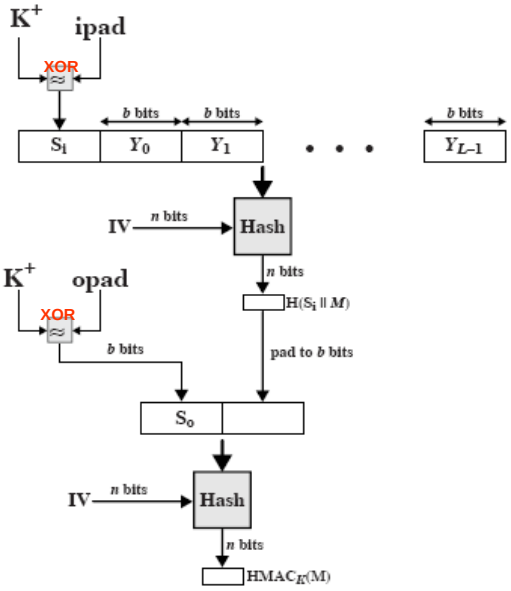
HMAC: sleutel en padding

- K: geheime sleutel
- indien lengte K > b blokgrootte (vb. 512)
  - K is invoer voor hashfunctie ➔ produceert n-bit sleutel
  - minimum lengte voor K is de hash-grootte n (vb. 128)
- K^+: voeg nul-bits toe aan K tot lengte = b
  - vb. lengte K =160; b=512; voeg 44x8=352 nullen toe
- ipad (inner pad) = 0x36 wordt b/8 keer herhaald → b bits
- opad (outer pad)= 0x5C wordt b/8 keer herhaald
- resultaat:
  - HMAC = H ((K^+ xor opad) || H ((K^+ xor ipad) || M))
- gebruik van ipad en opad resulteert in twee sleutels die pseudorandom gegenereerd zijn (helft v. bits gewijzigd)

HMAC: efficiëntere implementering

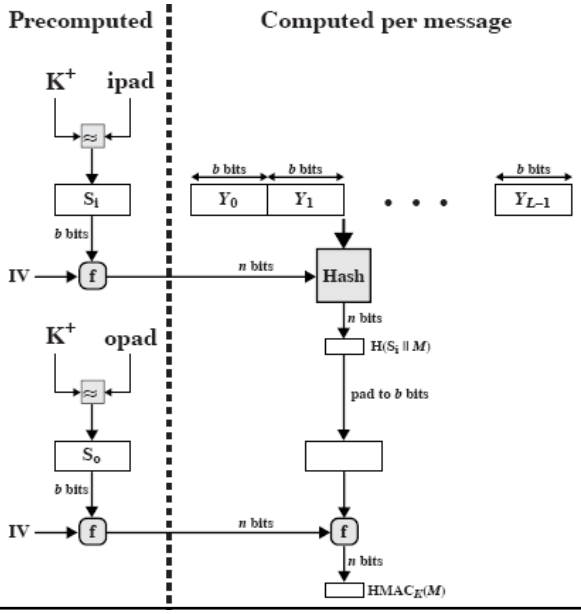
- deel wordt vooraf berekend
  - f(IV, (K^+ xor ipad)) en f(IV, (K^+ xor opad))
  - f = compressiefunctie van de hashfunctie
- herberekening alleen als sleutel verandert
- vervangt IV in hashfunctie
- zie figuur op volgende dia
- slechts 1 bijkomende toepassing van compressiefunctie

HMAC: schema



49

HMAC: algoritme



50

HMAC: veiligheid

- gebaseerd op de veiligheid van de ingebouwde hashfunctie
- er is bewezen dat
  waarschijnlijkheid op succesvolle aanval op HMAC
  ≈
  waarschijnlijkheid op succesvolle aanval op hasfunctie in 1 van volgende gevallen
- de aanvaller kan een output bereken, zelfs met een IV die random, geheim en onbekend voor de aanvaller is: vereist inspanning van 2^n
- de aanvaller vindt collisions in de hashfunctie, zelfs met een IV die random en geheim is.

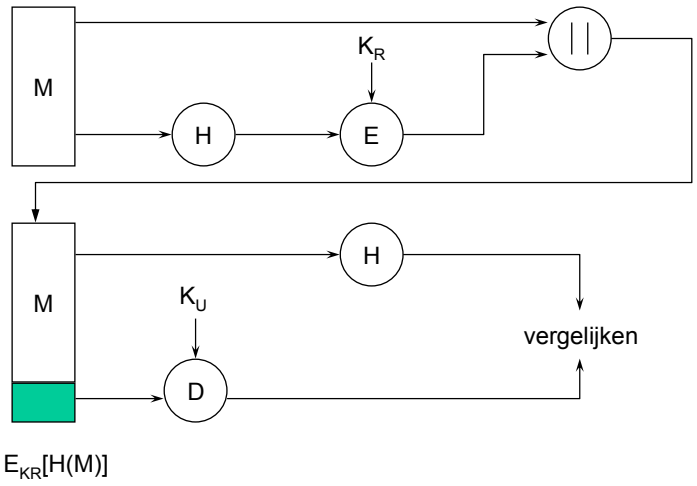
HMAC: veiligheid

- tweede aanval = birthday-attack:
  - zoek 2 boodschappen M en M' met zelfde hash
  - inspanning van $2^{n/2}$ → 2^{64} voor MD5 is doenbaar?
- is MD5 niet geschikt voor HMAC?
- toch wel:
  - offline zouden collisions kunnen gezocht worden voor verschillende waarden van M (IV en algo is gekend)
  - maar aanvaller kent K niet
  - observatie van HMAC-resultaten nodig met zelfde sleutel
  - 2^{64} blokken = 2^{73} bits → over 1Gbps link → 299.000 jaar

Digitale handtekeningen

- schema volgens RSA
- Digital Signature Standard DSS

Handtekening volgens RSA



Handtekening volgens RSA

Zender van boodschap
- hashcode van vaste lengte (meestal MD5)
- encryptie van hashcode met private sleutel K_R = handtekening
- versturen van boodschap + handtekening

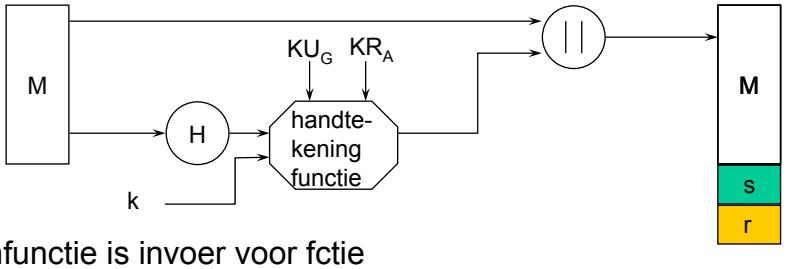
Ontvanger:
- bereken hashcode van boodschap
- decrypteer handtekening
- vergelijk

RSA ook voor encryptie en sleuteluitwisseling

Digital Signature Standard

- Opgesteld door NIST 1991-1993-1996 (FIPS PUB 186)
- SHA + Digital Signature Algorithm (DSA) = DSS
- DSS: alleen digitale handtekening
- discrete log berekenen is moeilijk (cfr Diffie-Hellmann)
- Methode: handtekeningfunctie met als invoer
  - hashcode van boodschap
  - random getal *k* (éénmalig voor deze handtekening)
  - private sleutel van zender
  - globale publieke sleutel (gekend door communicerende personen)
- Uitvoer functie twee delen: r en s

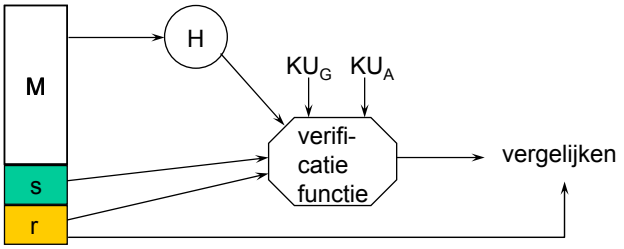
DSS – plaatsen handtekening (A)



- hashfunctie is invoer voor fctie
- k is random getal voor deze handtekening
- KR_A is private sleutel van “afzender A”
- KU_G globale publieke sleutel
  - set van parameters
  - gekend door groep spelers die onderling communiceren
- handtekening bestaat uit 2 delen: r en s

57

DSS – verifiëren handtekening



- hashcode wordt berekend
- verificatiefunctie heeft o.a. als invoer publieke sleutel van A
- r wordt vergeleken met resultaat van verificatiefunctie
- r is onafhankelijk van boodschap
- handtekening alleen geldig indien geplaatst met behulp van correcte private sleutel van A

58

Sleutels (1)

3 publieke parameters (groep gebruikers) vormen KU_G

- gekozen priemgetal q (160 bits)
- priemgetal p: bitlengte van 512 tot 1024 bits in stappen van 64 bits,
  zo dat *q deler is van (p-1)*
- $g = h^{(p-1)/q} \bmod p$
  g>1 en h geheel en gekozen tssn 1<h<(p-1)

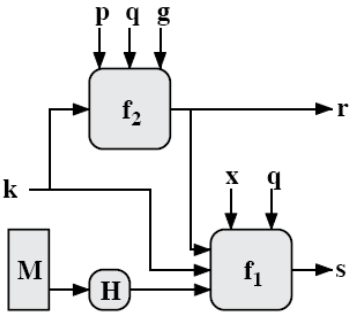
Sleutels (2)

Elke gebruiker

- kiest private sleutel x met $0 < x < q$ (random)
- berekent publieke sleutel : $y = g^x \bmod p$
- computationeel ondoenbaar om x te berekenen indien y gekend:
  x is discrete logaritme van y in basis g modulo p

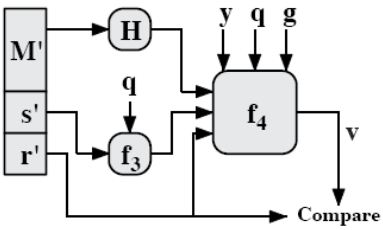
Functies: modulo-rekenen

Plaatsen handtekening: (r,s)



- $r = (g^k \bmod p) \bmod q$
  onafhankelijk van boodschap
- $s = ((H(M)+xr)\ k^{-1}) \bmod q$

Verificatie handtekening: (r,s)



- $r = (g^k \bmod p) \bmod q$
- $s = ((H(M)+xr)\ k^{-1}) \bmod q$

- $w = f3(q,s') = s'^{-1} \bmod q$
- $v = ((g(H(M')w) \bmod q\ y\ r'\ w \bmod q) \bmod p) \bmod q$
- v == r ?? bewijs zie gonzo

DSA: besluit

- r onafhankelijk van boodschap
- enig zware berekening, eventueel vooraf te maken
- r bepaalt mee de handtekening s
- ontvanger kan r bepalen uit s en hash van M

- x niet te bepalen uit y
- k niet bepalen uit r
- x niet te bepalen uit s