Forelesning 6

Hash-funksjoner
og
meldingsautentisering

Angrep

- Avsløring
- Trafikkanalyse
- Maskerade
- Modifikasjon av
  - Innhold
  - Rekkefølge
  - Tid
- Fornektelse

Meldingsautentisering

Digitale signaturer

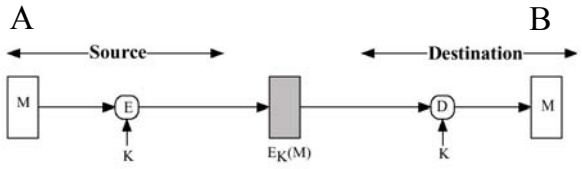
Autentiseringsfunksjoner

- Enhver meldingsautentiseringsfunksjon kan betraktes som to deler:
  - En funksjon som produserer en autentikator
  - En høyere-nivå protokoll som bruker denne funksjonen som et primitiv for å verifisere at en melding er autentisk

Autentikator

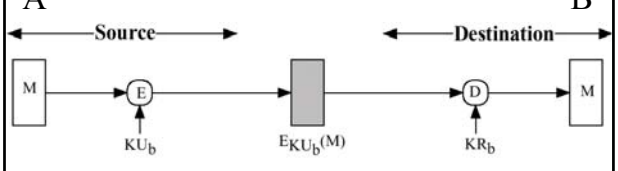
- Funksjoner som kan brukes til å generere en autentikator
  - Kryptering
  - Meldingsautentiseringskode – MAC
    Offentlig metode og hemmelig nøkkel
  - Hash-funksjon
    Offentlig metode (ikke nok alene!)

Konvensjonell kryptering



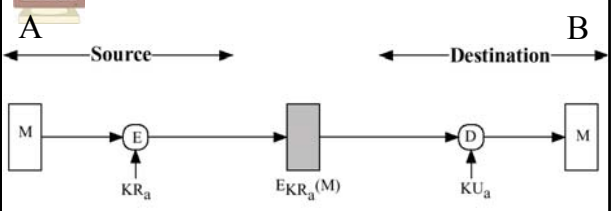
Konfidensialitet og autentisering

Offentlig-nøkkel kryptering



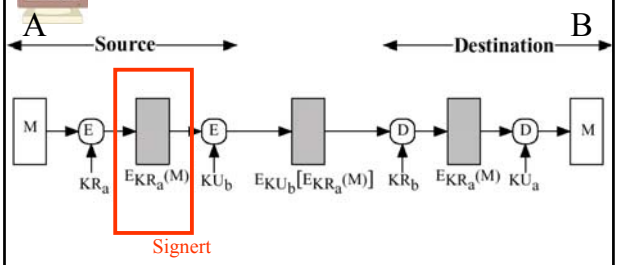
Konfidensialitet

Offentlig-nøkkel signering



Autentisering og signatur

Offentlig-nøkkel kryptering

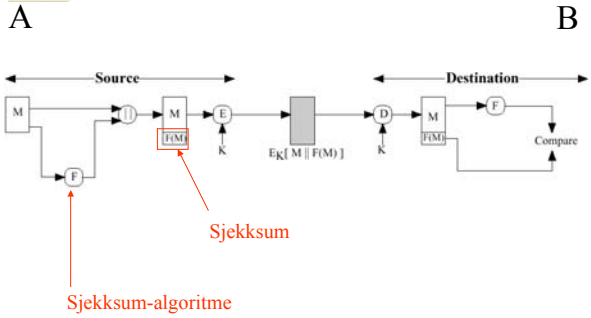


Konfidensialitet, autentisering og signatur

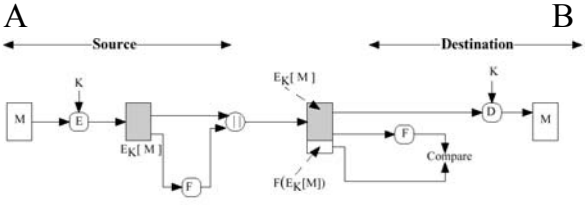
Kryptering som autentisering

- Symmetrisk kryptering gir implisitt autentisering, ettersom hvis en melding til A lar seg dekryptere til noe fornuftig med K_{ab}, vet A at den må komme fra B, siden bare B har en kopi av nøkkelen.
- Hvordan automatisk avgjøre om en dekryptert melding er "fornuftig"?

Intern meldingskontroll



Ekstern meldingskontroll



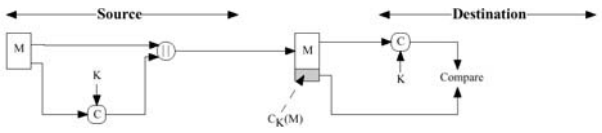
Gir ingen kontroll på om meldingsinnholdet er "fornuftig"!

Meldingsautentiseringskode

- Genererer en blokk av data av fast lengde fra en melding av variabel lengde og en hemmelig (symmetrisk) nøkkel K
- "Kryptografisk sjekksum"
- MAC = $C_K(M)$

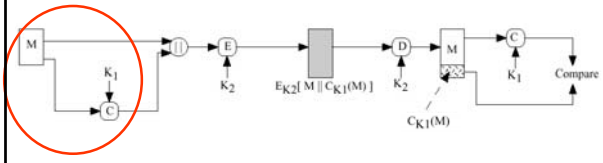
Meldingsautentisering

A B



... og konfidensialitet

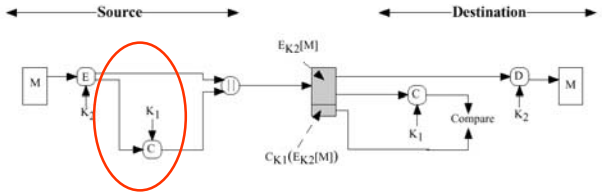
A B



Autentiseringen er koblet til klarteksten

... og konfidensialitet (2)

A B



Autentiseringen er koblet til chifferteksten

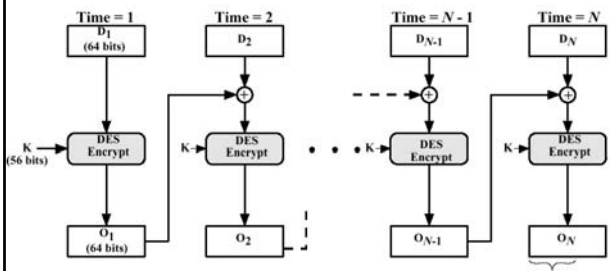
Hvorfor MAC?

- Kringkastingsmeldinger
- Høy belastning, stikkprøver
- Verifikasjon av programvare
- Hemmeligholdelse uviktig, autentisering viktig (SNMPv3)
- Fleksibilitet (konfidensialitet og autentisering på forskjellige protokollnivå)
- Autentisering etter mottak (verifisering av gamle meldinger)

Krav til MAC

- Gitt M og $C_K(M)$ skal det være "umulig" å konstruere en melding M' slik at $C_K(M) = C_K(M')$
- $C_K(M)$ må ha en uniform fordeling
- Hvis M'=f(M) skal det ikke være større sannsynlighet for at $C_K(M) = C_K(M')$ (ingen "svake punkter")

MAC vha. blokkchiffer (DES-CBC)



Data Authentication Algorithm

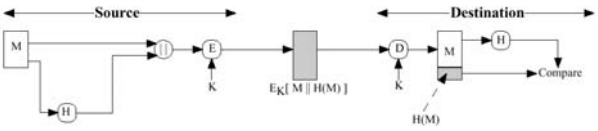
DAC= "Data Authentication Code"

Hash-funksjoner

- Tar en melding av variabel lengde, og generer en *hash* av fast lengde
- En hash kalles også ”message digest”
- h=H(M)
- Enveis-funksjon: ”Umulig” å finne en **M’** som genererer en gitt **h**
- Hash-funksjonen er ikke hemmelig, så hashen må beskyttes på andre måter

Autentisering og konfidensialitet

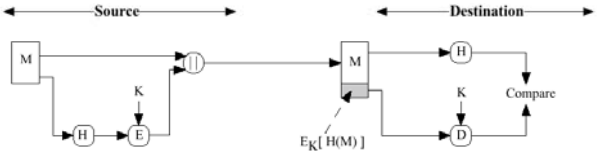
A B



Konvensjonell kryptering - hash bidrar til å avgjøre om innholdet er ”fornuftig”

Autentisering

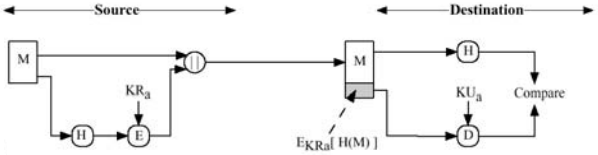
A B



Dette er egentlig en MAC!

Autentisering og signatur

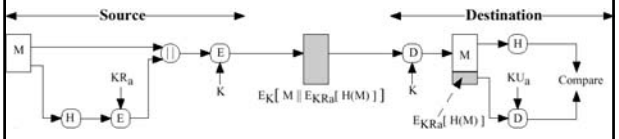
A B



Dette er en digital signatur!

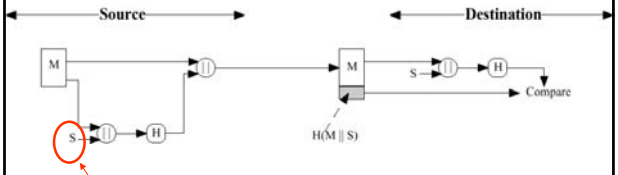
Signatur og konfidensialitet

A B



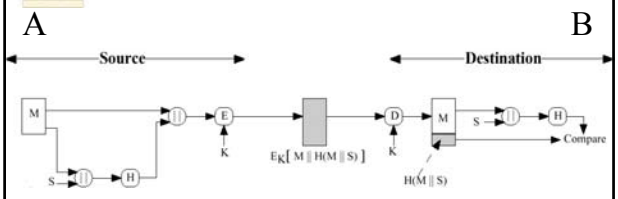
Autentisering uten kryptering

A B



Hemmelig verdi som bare A og B kjenner

... og med konfidensialitet



Krav til Hash-funksjoner

- Kan anvendes på data av vilkårlig størrelse (*)
- Genererer output av fast lengde
- H(x) lett å beregne
- Enveis-egenskap
- Gitt x, umulig å finne y≠x| H(x)=H(y) (svak kollisjonsmotstand)
- Umulig å finne (x,y)| H(x)=H(y) (sterk kollisjonsmotstand)

Kollisjonsmotstand

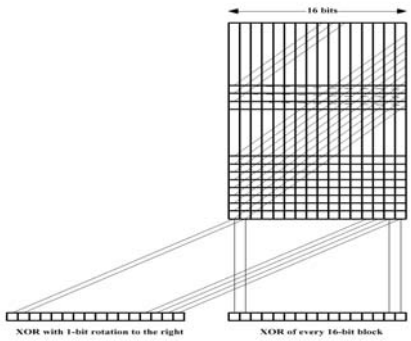
- Siden en hash-funksjon vanligvis er en transformasjon fra en STOR data-mengde til en LITEN, *må* det nødvendig-vis forekomme kollisjoner, dvs. at to for-skjellige input mapper til samme output. Poenget er at det skal være vanskelig å *finne* to slike input!

Enkel Hash funksjon

B1	1	0	0	0	1	1	1	1	0	0	0	1	0	1	0	0
B2	0	0	0	0	1	1	0	1	0	0	1	1	0	0	1	0
H	1	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0

Del meldingen opp i blokker, og
foreta bitvis XOR mellom blokkene

Enkle hash-funksjoner



Eksempel enkel hash-funksjon

Steg 1

B1	0	1	0	
B2	1	1	1	
B3	0	1	1	
H'	0	0	0	
H	0	1	0	

Steg 2

B1	0	1	0	
B2	1	1	1	
B3	0	1	1	
H'	1	0	0	
H	0	1	1	

Steg 3

B1	0	1	0	
B2	1	1	1	
B3	0	1	1	
H'	1	1	0	
H	1	0	1	

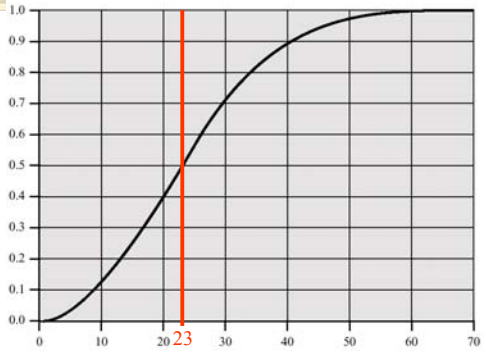
H= første blokk XOR H'

H' = roter H ett bit

Birthday Paradox

- Hvor mange elever må det være i en klasse for at det skal være 50% (eller mer) sannsynlighet for at to elever har samme fødselsdag?

Birthday Paradox forts.



Neida, vi går ikke inn på matten

- Det er en lav sannsynlighet for at noen har samme fødselsdag som en tilfeldig valgt elev
- Det overraskende lave tallet elever (23) for å få sannsynlighet 50% kommer av at man ser på *alle mulige* par av elever
- Dette er også forskjellen på svak og sterk kollisjonsmotstand!

Birthday Attack

- A "signerer" en melding M ved å generere en n-bits hash som krypteres med hemmelig nøkkel. M sendes i klartekst, med kryptert hash heftet ved.
- Har fra Birthday Paradox at man med en n-bits hash-kode vil finne en kollisjon med sannsynligheten 0,5 etter $2^{n/2}$ input

Birthday Attack, forts.

- Motstander generer $2^{n/2}$ variasjoner av M, og beregner hash for disse
- Motstander lager en falsk melding, genererer $2^{n/2}$ variasjoner av denne, og beregner hash for alle disse også
- Vil med sannsynlighet større enn 0,5 finne 2 meldinger som genererer samme hash!

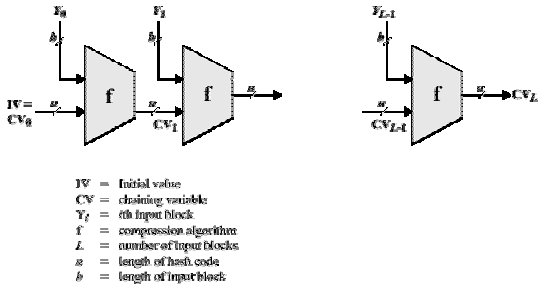
Lærdom fra Birthday Attack

- Hash-funksjoner må generere hasher som er *dobbelt så lange* som man skulle tro når man sammenligner med symmetriske nøkler som er robuste mot brute-force

Meet-in-the-middle for hasher

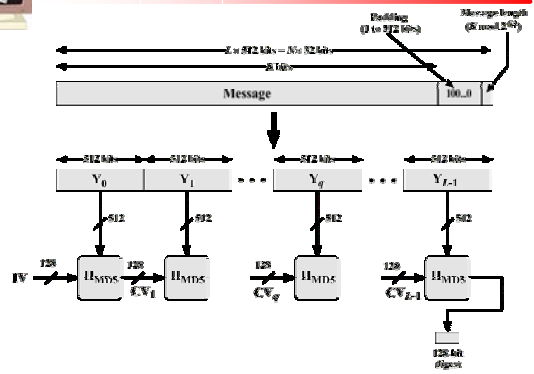
- CBC-baserte hash-varianter er sårbare for m-i-t-m-angrep
- Gjør det mulig å legge til informasjon uten at det forandrer hashen til slutt
- Har bidratt til fokus på å finne andre former for hash-funksjoner

Modell for hash-funksjoner

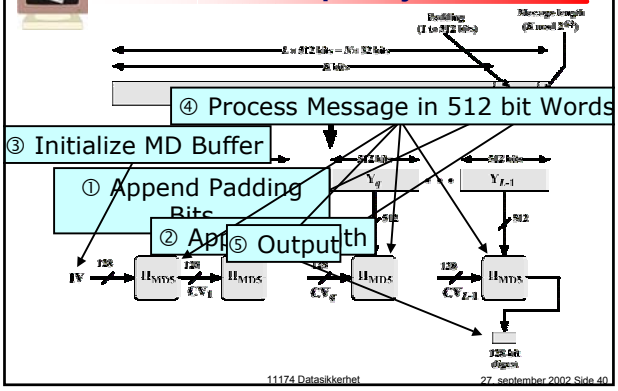


IV = Initial value
CV = chaining variable
Y_i = ith input block
f = compression algorithm
L = number of input blocks
n = length of hash code
b = length of input block

MD5



MD5 operasjon



Andre hash-algoritmer

- MD4 – forløper til MD5
  - Ikke like sikker, men raskere
- Secure Hash Algorithm 1 (SHA-1)
  - Bygget over samme lest som MD5
- RIPEMD-160

Sammenligning

	MD5	SHA-1	RIPEMD-160
Digest length	128 bits	160 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	64 (4 rounds of 16)	80 (4 rounds of 20)	160 (5 paired rounds of 16)
Maximum message size	∞	$2^{64} - 1$ bits	$2^{64} - 1$ bits
Primitive logical functions	4	4	5
Additive constants used	64	4	9
Endianness	Little-endian	Big-endian	Little-endian

Algorithm	Mbps
MD5	32.4
SHA-1	14.4
RIPEMD-160	13.6

HMAC

- MAC vha hash-funksjon og hemmelig nøkkel
- RFC 2104
- Brukes bl.a. av Microsoft i NTLMv2-autentisering (HMACT-64)

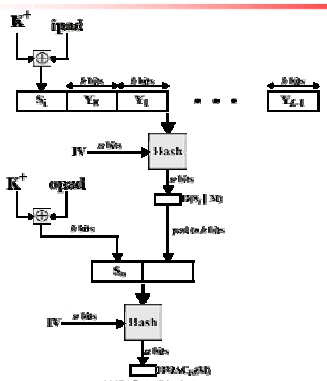
HMAC notasjon

- H – hash-funksjon (MD5, SHA-1, etc.)
- M – melding (inkludert padding)
- Y_n – n'te blokk av M
- L – antall blokker i M
- b – antall bit i en blokk
- n – lengde av hash generert av H

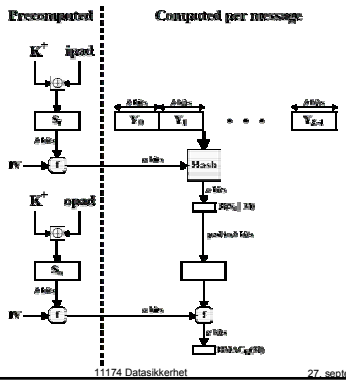
HMAC notasjon, forts.

- K – hemmelig nøkkel
- K^+ – K paddet med nuller til b bit
- ipad – 00110110 gjentatt b/8 ganger
- opad – 01011100 gjentatt b/8 ganger
- IV – initialiseringsvektor for hash-funksjonen (konstant for gitt hash-funksjon)

HMAC



HMAC Optimization



Dagens website

http://www.sans.org

- Newsbites
- Bulletins