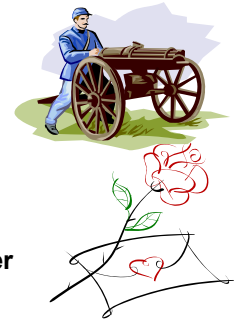
Kryptering og steganografi

EJHJUBM SFQSFTFOUBTKPO FS FU LVMU GBH

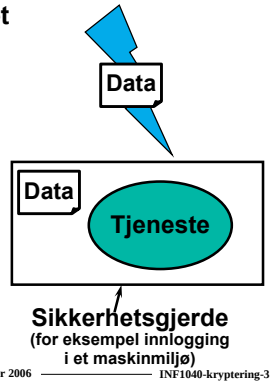
Hemmeligholdelse av budskap

- Vi er ofte interessert i å gjøre data uleselig for uvedkommende, eller å gjemme dem slik at uvedkommende ikke kan finne dem
- Dette har vært gjort i uminnelige tider, lenge før datamaskinen ble oppfunnet – særlig i forbindelse med krig og kjærlighet
- Like lenge har vi hatt en kamp mellom de som gjemmer og de som finner
- Datamaskinteknologien har åpnet nye muligheter for begge parter



Gjøre data uleselige eller usynlige

- Data kan gjøres uleselige for utenforstående (kryptografi) eller skjules (steganografi)
- Spesielt aktuelt for data som befinner seg i usikrede omgivelser, som for eksempel:
  - På vei over nettet fra et system til et annet (dataoverføringer, meldinger)
  - På en bærbar datamaskin (stjålet?)
  - På minnepinne, CD eller diskett
- Også aktuelt som en ekstra beskyttelse for kritiske data som befinner seg i sikrede omgivelser.



Kryptologi

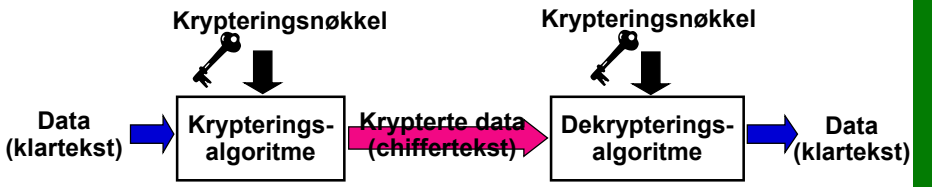
fra gresk

- cryptos: Hemmelig, gjemt
- graphos: Skriving
- kryptografi: Studiet (vitenskap/kunst) av hemmelig skrift
- kryptoanalyse: Studiet av metoder og prinsipper for å kunne tyde en kryptert tekst uten kjennskap til krypteringsnøkkelen (”knekke koden”)

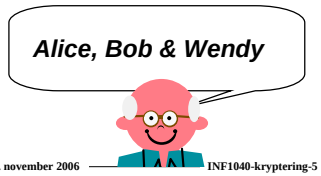
  Krypteringsnøkkel:
  Data som brukes for å kryptere og dekryptere meldinger

- kryptologi : Kryptografi + kryptoanalyse

Kryptering av data



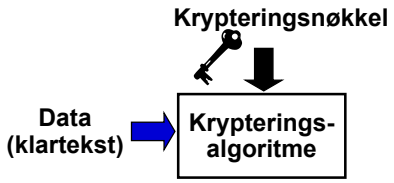
- Formål:
  - Gjøre data som sendes eller lagres uleselige for uvedkommende
- Utfordringer:
  - Finne tilstrekkelig gode krypterings- og dekrypteringsalgoritmer
  - Gjøre det praktisk umulig å finne krypteringsnøkkelen ("knekke koden")
  - Administrere krypteringsnøkler



Et enkelt eksempel: Cæsars kode

- Meget enkel krypteringsalgoritme:
  Bytt ut hver bokstav med en bokstav et bestemt antall plasser k lenger opp i alfabetet. (Ved slutten av alfabetet, "wrap around")
  Verdien av k er nøkkelen
  Eksempel:
  `DIGITAL REPRESENTASJON ER ET KULT FAG`
  `EJHJUBM SFQSFTFOUBTKPO FS FU LVMU GBH`
- Meget enkel dekrypteringsalgoritme:
  Bytt ut hver bokstav med bokstaven k antall plasser lenger ned i alfabetet. (Ved begynnelsen av alfabetet, "wrap around")
- Cæsars kode er et eksempel på kryptering med bruk av *substitusjonsprinsippet* med *syklisk permutasjon*
- Cæsars kode er et eksempel på *symmetrisk kryptering* – samme nøkkel brukes både for kryptering og dekryptering

Kerckhoffs' assumption



A. Kerckhoff, 1883:

"The security of a cipher must not depend on anything that cannot be easily changed"

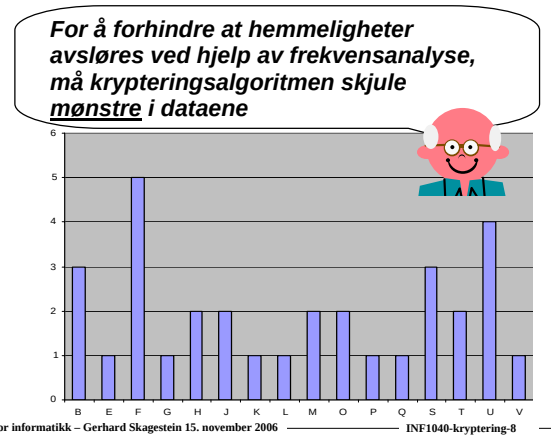
"The opponent is not to be underestimated. In particular, the opponent knows the encryption and decryption algorithms. So the strength of a cipher system depends on keeping the key information secret, not the algorithm"

se også http://en.wikipedia.org/wiki/Kerckhoffs%27_law

Hvordan bryte Cæsar's kryptering

- "Brute force":
  Prøv alle N muligheter for k der N er antall tegn i alfabetet og se hvilken forskyvning som gir et forståelig resultat.
  ("Exhaustive search of the key space")
- Frekvensanalyse:
  Lag et histogram for tegnene i den krypterte meldingen. Anta at bokstaven som forekommer oftest står for bokstaven E.

  Aha! k = 1



Krypteringsprinsipper

- Krypteringsnøkkel
  - Symmetrisk kryptering
    Samme nøkkel brukes for både kryptering og dekryptering
  - Asymmetrisk kryptering
    To sammenhengende nøkler,
    den ene brukes for kryptering, den andre for dekryptering
- Håndtering av data
  - Stream-kryptering
    Krypterer bitene eller bytene etter hvert som de kommer
  - Blokk-kryptering
    Opererer på en blokk av biter – typisk fra 64 til 384 – ad gangen
- Algoritmer
  - Substitusjonsprinsippet
    Bytte ut biter eller bytes med andre biter og bytes
  - Transformasjonsprinsippet
    Endre rekkefølgen på biter eller bytes

Symmetrisk kryptering

- Kryptering av klartekst P med nøkkel k gir chiffertekst C:
  $E_k(P) \rightarrow C$
- Dekryptering av chiffertekst C med *samme* nøkkel k gir klartekst P:
  $D_k(C) \rightarrow P$
- dvs. $D_k(E_k(P)) \rightarrow P$
- Kommunikasjonspartnerne deler en hemmelighet: Nøkkelen
- Fordel: Brukbar effektivitet
- Problem: Distribusjon (og hemmeligholdelse) av nøkkelen

Vigenère kryptering

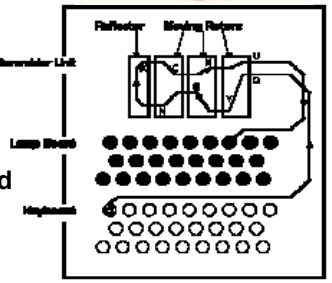
- Ligner Cæsar's kryptering,
  men bruker en *frase* som nøkkel.
  Symmetrisk stream-kryptering som bygger på substitusjonsprinsippet
- Eksempel:
  Klartekst `DIGITAL REPRESENTASJON`
  Nøkkel `HEI`
  Utdrag av Vigènere kodetabell
- Kryptering med Cæsar's kryptering for hver bokstav:

```
klartekst   DIGITAL REPRESENTASJON
nøkkel      HEIHEIH EIHEIHEIHEIHEI
chiffertekst KMOPXIS VMXVMAIVBEBQSV
```

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

Enigma - Rotormaskin

- Brukt bl.a. av nazi-regimet under andre verdenskrig
- Symmetrisk stream-kryptering som bygger på substitusjonsprinsippet
  - Hver rotor gjør en substitusjon
  - Rotorene er forskjellige og kan bytte plass
  - Et tastetrykk bevirker at tegnet sendes gjennom alle rotorene og tilbake (én rotor virker som reflektor)
  - Rotorene dreies automatisk etter hvert tastetrykk
- Nøkkelen er startposisjonen for rotorene
- Tilsvarer en nøkkellengde på 380 biter (med "plugboard"
- Flere detaljer på
  http://www.codesandciphers.org.uk/enigma/



http://www.math.miami.edu/~harald/enigma/enigma.gif

Hva medfører datateknologien?

- Mulighet for mer raffinerte krypterings- og dekrypteringsalgoritmer
- Vi krypterer biter og bytes, ikke tegn
  – krypteringsalgoritmene ser altså ikke forskjell på tekst, bilder og lyd
- Kraftigere verktøy for kryptering…
- …men også kraftigere verktøy for kryptoanalyse ("knekking" av kryptering)

Faktisk har kryptoanalyseorganisasjoner vært med på å finansiere utvikling av datateknologi

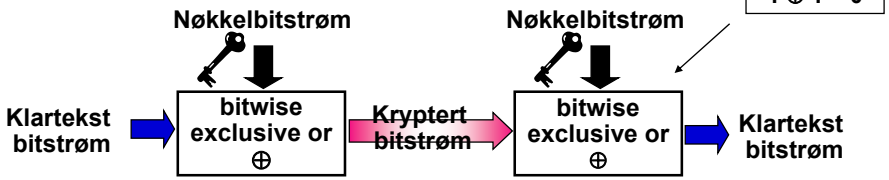
Kryptoanalysens absolutte begrensning

- Kraftige datamaskiner gjør det tenkelig å utføre "brute force attacks"…
- …men jo lengre nøkler, jo mer regnekraft trengs
- Et enkelt regnestykke:
  - Nøkkellengde 128 biter, antall mulige nøkler er 2^{128}
  - La oss anta at det tar 0,001 sekund å sjekke ut en nøkkel…
  - …da blir tidsforbruket i gjennomsnitt
    $0{,}5 * 0{,}001 * 2^{128}$ s = $1{,}7 * 10^{35}$ s = $5{,}4 * 10^{27}$ år
- Men vi kan jo bruke en mengde maskiner og prosessorer i parallell?
- Vi møter imidlertid en absolutt begrensning:
  - Termodynamikkens lover sier at det trengs en viss energi for å "flippe" en bit
  - Man kan estimere hvor mange biter som må flippes i en kryptoanalyse
  - Hvis den estimerte energien overskrider tilgjengelig energimengde mottatt fra sola, har vi et problem…
- Altså må vi utnytte svakheter i krypteringen for å gjøre det mer effektivt!

Vernam-kryptering

- Den enkleste formen for symmetrisk stream-kryptering
- Oppfunnet av Gilbert Vernam i 1917 for telex
  se http://en.wikipedia.org/wiki/Vernam_cipher

$0 \oplus 0 = 0$
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$
$1 \oplus 1 = 0$



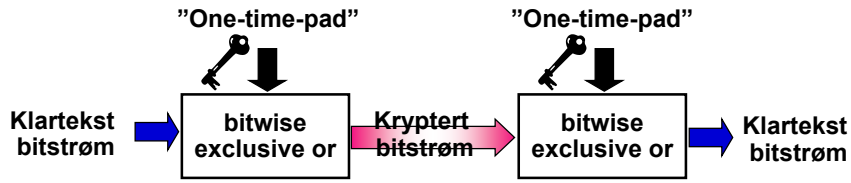
Eksempel:
'to og to ' i UTF-8: 74 6F 20 6F 67 20 74 6F 20
= 01110100 01101111 00100000 01101111 01100111 00100000 01110100 01101111 00100000

Tilfeldig valgt nøkkel: 01110011 01100101 01111000

klartekst	01110100 01101111 00100000 01101111 01100111 00100000 01110100 01101111 00100000
nøkkel	01110011 01100101 01111000 01110011 01100101 01111000 01110011 01100101 01111000
chiffertekst	00000111 00001010 01011000 00011100 00000010 01011000 00000111 00001010 01011000
nøkkel	01110011 01100101 01111000 01110011 01100101 01111000 01110011 01100101 01111000
klartekst	01110100 01101111 00100000 01101111 01100111 00100000 01110100 01101111 00100000

"One-time-pad"

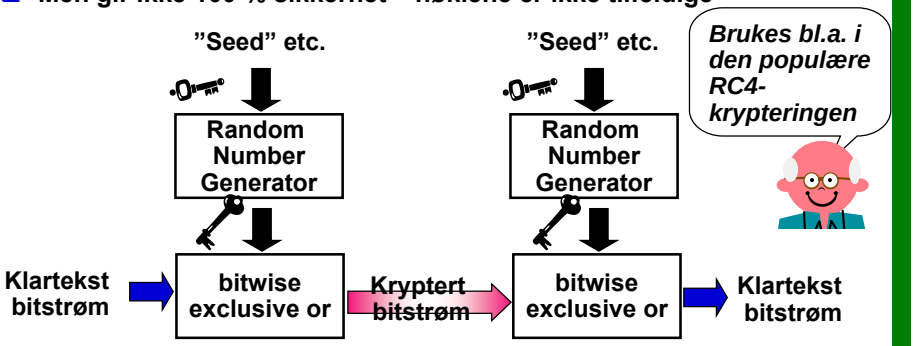
- Vernam-kryptering med en tilfeldig nøkkel minst så lang som meldingen, *og som brukes bare en gang*
- Oppfunnet av Vernam og Mauborgne på 1920-tallet
  se http://en.wikipedia.org/wiki/One-time_pad
- Bevist av Shannon å være 100 % sikker
  Intuitivt: Enhver tekst med et gitt antall tegn kan genereres med en passende nøkkel



Når "one-time-pad" er 100 % sikker, hva er da problemet?

Generert "One-time-pad"

- Problemet med "one-time-pad" er distribusjonen av et stort antall meget lange tilfeldige nøkler (men dette problemet er kanskje minkende?)
- Et alternativ er å erstatte "one-time-pad" med en sekvens av *pseudo-tilfeldige* biter fra en "random number generator" (se neste lysark)
- Nøkkelen blir da meget kort (består av "seed" og evt. konstanter)
- Men gir ikke 100 % sikkerhet – nøklene er ikke tilfeldige



Pseudotilfeldige tall

- Mange moderne krypteringsteknikker bygger på sekvenser av pseudotilfeldige tall
- Pseudotilfeldige tall ser ut som tilfeldige tall, men hvert tall i sekvensen er beregnet på grunnlag av det forrige.
  Det første tallet er beregnet på grunnlag av en startverdi ("seed").
- Nøkkelen består av "seed" og eventuelle konstanter (se neste lysark).
- Med denne nøkkelen og tilgang til algoritmen for beregning av de pseudotilfeldige tallene kan tallsekvensen regenereres av mottakeren av meldingen!

Algoritmer for pseudotilfeldige tall

- Eksempel på algoritme (ikke særlig velegnet for kryptering)
  Linear Congruential Pseudo-number Generator

  $x_{n+1} = (C*x_n + D) \% M$

  der `C`, `D` og `M` er konstanter valgt slik at:
  - `C` og `D` er relativ prim (ingen felles faktorer)
  - `C-1` er delbar med alle primfaktorer i `M`
  - hvis `M` er delelig med 4, er også `C-1` det.
- Da gir algoritmen alle tall fra 0 til M-1 i "tilfeldig" rekkefølge før sekvensen gjentas.
- Eksempel: C=5, D=3, M=16, seed=5 gir
  12, 15, 14, 9, 0, 3, 2, 13, 4, 7, 6, 1, 8, 11, 10
- For å få en noenlunde sikker nøkkel må M være stor! (> 128 biter)
- se også http://en.wikipedia.org/wiki/Pseudo-random_number_generator

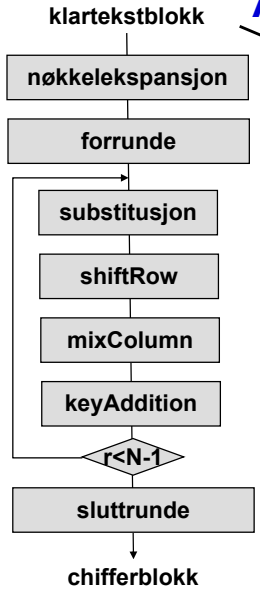
Blokk-kryptering

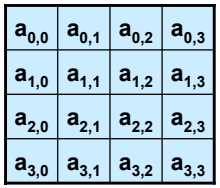
- Blokk-kryptering opererer på en blokk av biter – typisk fra 64 til 384 – ad gangen
- Blokk-kryptering åpner for permutasjon av bitene i blokken
- To hovedprinsipper
  - "Confusion"
    Sørge for en så komplisert sammenheng mellom nøkkel og chiffertekst at kryptoanalytikeren ikke kan få nyttig informasjon om nøkkelen ved statistisk analyse av chifferteksten
  - "Diffusion"
    Skjule mønstre i klarteksten ved å spre blokkene over hele chifferteksten – en endring av en bit i klarteksten bør gi endring i mange biter i chifferteksten.
    Løses med ombytting og xor'ing av blokker
- se også http://en.wikipedia.org/wiki/Block_cipher

Advanced Encryption Standard – AES

- Vinner av en konkurranse utlyst av NIST (National Institute of Standards and Technology) i USA – vinneren kåret i oktober 2000
- Konstruert av to belgiere, Joan Daemon og Vincent Rijmen, under navnet *Rijndael*
- Blokk- og nøkkellengde 128, 192 eller 256 biter
- Ingen patenter, kan benyttes av alle
- Motstandsdyktig mot alle kjente kryptoanalysemetoder
- Enkel, rask, lett å implementere i maskin- og/eller programvare
- Avløser DES – Data Encryption Standard

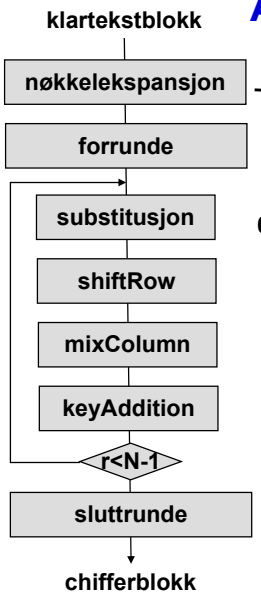
AES – virkemåte (1)

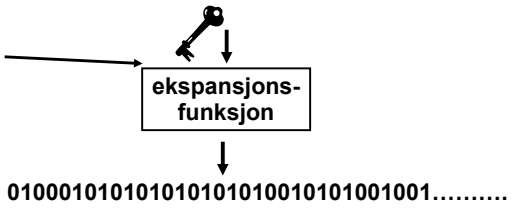




- Klartekstblokken oppfattes som en 4 * 4 tabell (matrise) med bytes (AES-standarden)
- Generelt kan større blokker og matriser kan benyttes
- Antall runder (10, 12 eller 14) avhenger av blokkstørrelsen

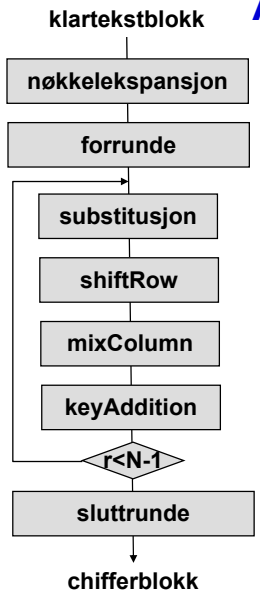
AES – virkemåte (2)

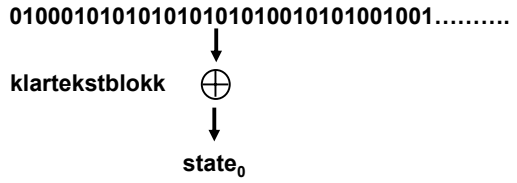




- Nøkkelen ekspanderes ved hjelp av en algoritme (Rijndael's key schedule) til et pseudo-tilfeldig bitmønster av tilstrekkelig lengde

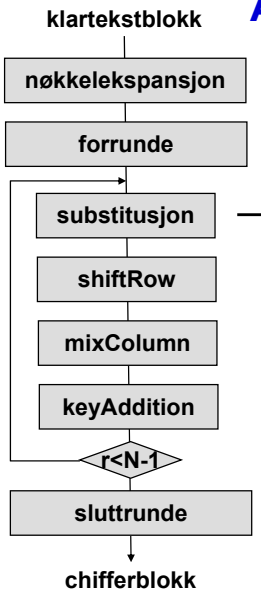
AES – virkemåte (3)

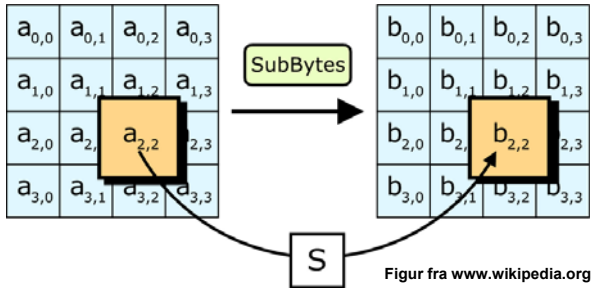




- Klartekstblokken xor'es med den første delen av den ekspanderte nøkkelen.
  Dette gir utgangstilstanden for iterasjonene – $state_0$

AES – virkemåte (4)

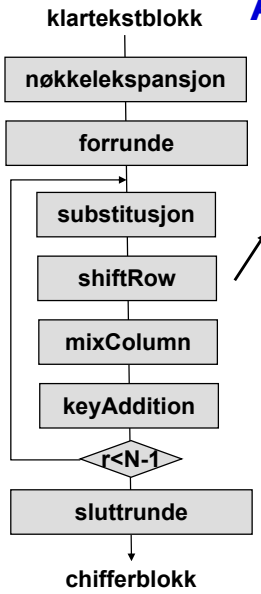


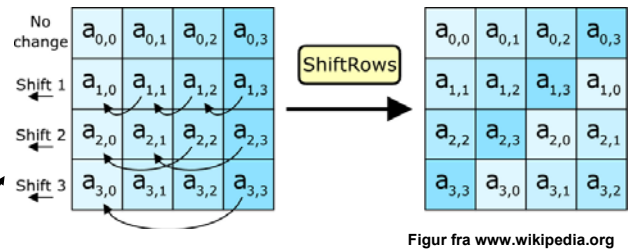


Figur fra www.wikipedia.org

- Hver enkelt byte erstattes med en annen byte som finnes ved å slå opp i en tabell S med 2^8 elementer
- Tabellen er konstruert slik at vi får best mulig "confusion"

AES – virkemåte (5)

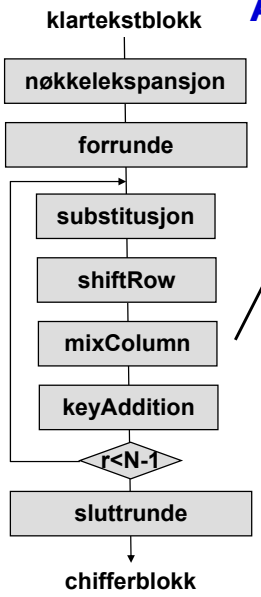


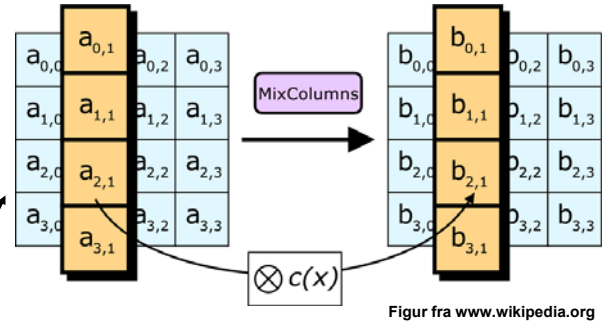


Figur fra www.wikipedia.org

- Linjene i tilstandsmatrisen roteres
- Dette er første ledd i "diffusion"

AES – virkemåte (6)

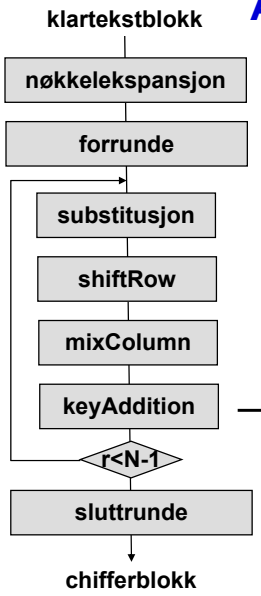


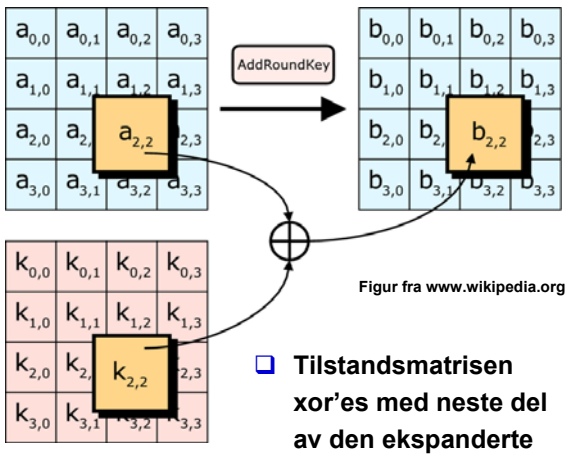


Figur fra www.wikipedia.org

- Kolonnene matrisemultipliseres med en fast vektor c(x)
- Det innebærer at innholdet i en a-byte påvirker innholdet i 4 b-bytes
- Dette er andre ledd i "diffusion"

AES – virkemåte (7)

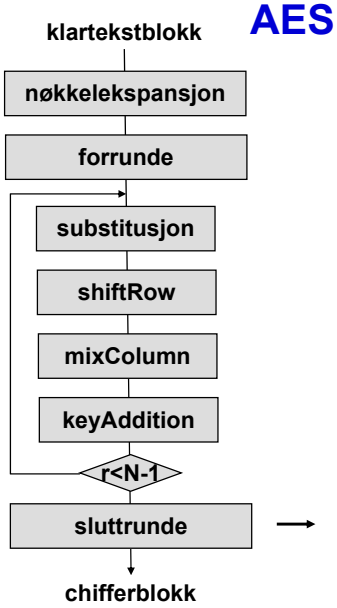




Figur fra www.wikipedia.org

- Tilstandsmatrisen xor'es med neste del av den ekspanderte nøkkelen

AES – virkemåte (8)



- Sluttrunden er som alle de andre, bortsett fra av skrittet mixColumn er utelatt

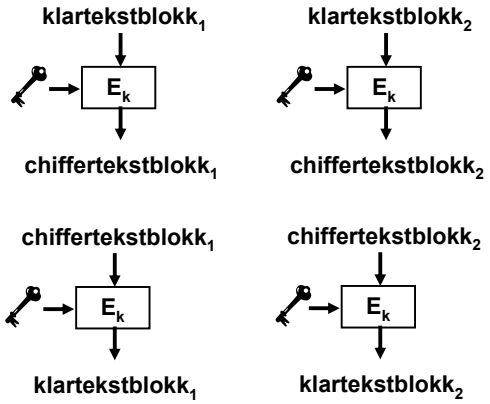
Bruksmåter (”modes of operation”)

- For å kryptere bitstrømmer lengre enn blokklengden, bakes blokk-krypteringen inn i ulike bruksmåter (”modes of operation”):
  - Electronic Code Book – ECB
  - Cipher feedback – CFB
  - Cipher Block Chaining – CBC
  - Output feedback – OFB
  - Counter mode – CTR
- Noen av disse utfører ”diffusion” utenfor blokklengden
- Ulike egenskaper med hensyn på sikkerhet og effektivitet
- For detaljer, se de følgende lysark
- se også http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

Derfor snakker vi for eksempel om AES-CBC

Electronic Code Book - ECB

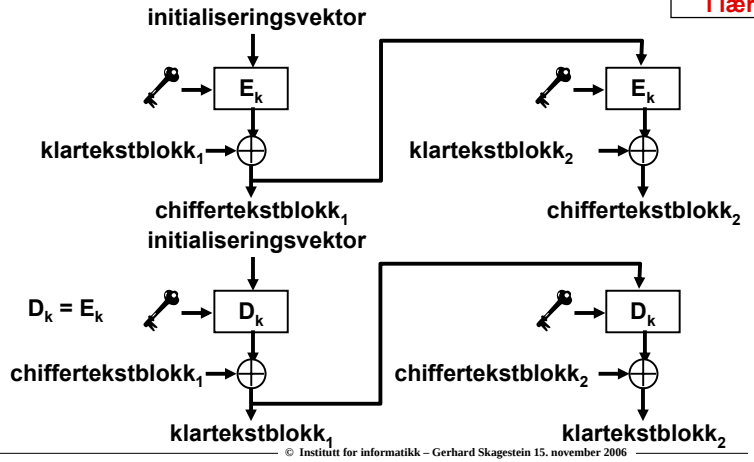
- Hver blokk krypteres for seg
- Ulempe: Like klartekstblokker gir like chiffertekstblokker (dårlig ”diffusion”)



Cipher Feedback - CFB

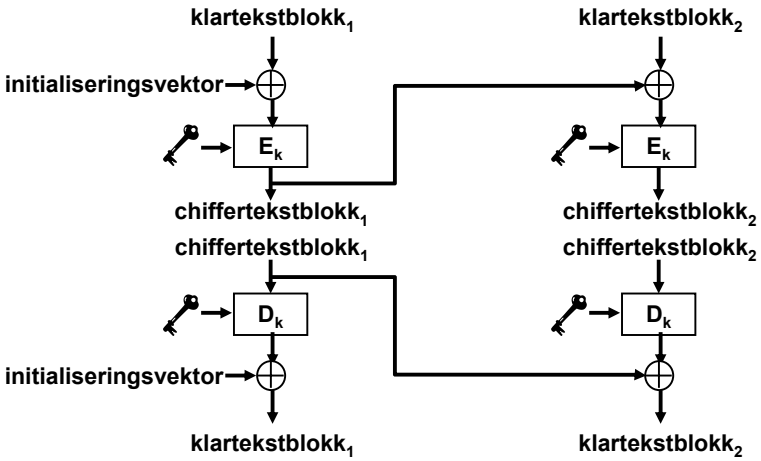
- Foregående chiffertekstblokk krypteres med nøkkelen – første gang krypteres initialiseringsvektoren
- Blokken xor’es med resultatet

OBS: Feil i figuren i læreboka



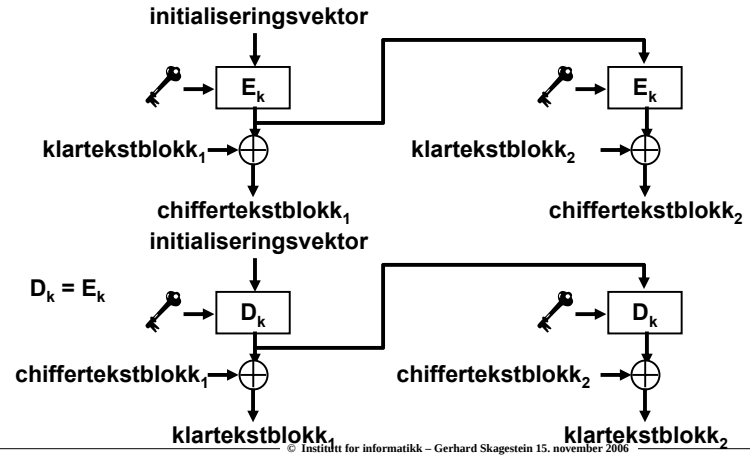
Cipher Block Chaining – CBC

- Klartekstsblokkene xor'es med foregående chiffertekstblokk
- Resultatet krypteres med nøkkelen



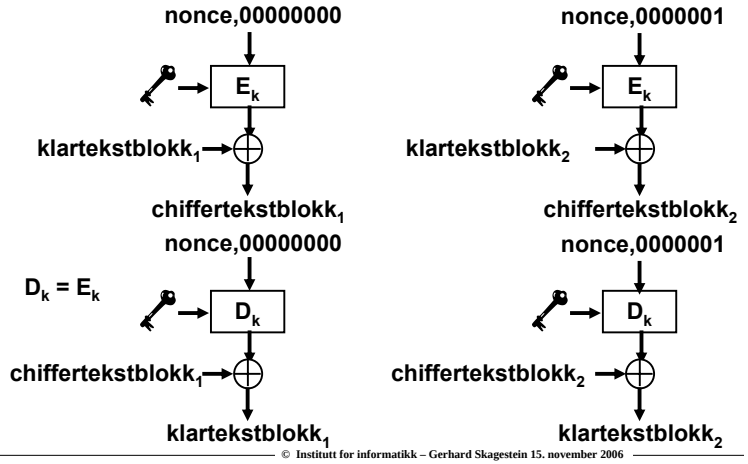
Output Feedback - OFB

- Ved første gjennomløp krypteres initialiseringsvektoren
- Ved senere gjennomløp krypteres resultatet av foregående kryptering
- Klartekstblokken xor'es med resultatet



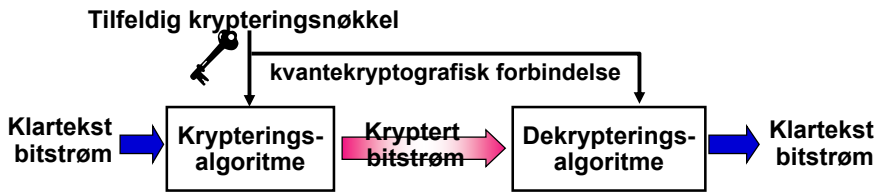
Counter mode – CTR

- En "nonce" ("number used once") etterfulgt av en teller krypteres med nøkkelen.
- Klartekstblokken xor'es med resultatet



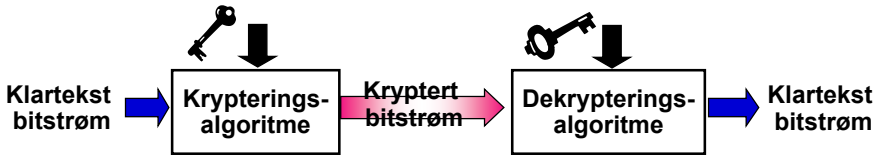
Kvantekryptografi

- Symmetrisk kryptering har et stort problem: Oversending av nøkler på en sikker måte
- Her kan den såkalte kvantekryptografien komme til hjelp
- Bygger på at polariseringsretningen for fotoner i lys kan avleses bare én gang (BB84-protokollen)
  - Derfor er det umulig å lytte ubemerket på linjen
- Fungerer bra over opptil 50 km optisk linje
- Foreløpig få kommersielle anvendelser – men mye forskning
- se f.eks. http://www.fysikknett.no/optikk/anvendelser6.php?menuid=3



Asymmetrisk kryptering

- Asymmetrisk kryptering arbeider med et *nøkkelpar* k1 og k2 som er matematisk relatert til hverandre – den ene nøkkelen brukes for kryptering, den andre for dekryptering
- Kryptering av klartekst P med nøkkel k1 gir chiffertekst C: $E_{k1}(P) \rightarrow C$
- Dekryptering av chiffertekst C med nøkkel k2 gir klartekst P: $D_{k2}(C) \rightarrow P$
- Asymmetrisk kryptering forenkler nøkkeladministrasjonen, men algoritmene er tunge!



Praktisk bruk av asymmetrisk kryptering (1)

- Alice vil sende en hemmelig melding til Bob
- På forhånd har Bob fått generert et nøkkelpar Bobk1 og Bobk2
  - Bobk1 offentliggjør han som sin *offentlige nøkkel*
  - Bobk2 holder han strengt hemmelig som *privat nøkkel*
- Alice krypterer meldingen med Bobk1, altså Bobs offentlige nøkkel
- Bob er den eneste som kan dekryptere meldingen, fordi bare han er i besittelse av Bobk2, den tilhørende private nøkkelen

Praktisk bruk av asymmetrisk kryptering (2)

- Alice vil sende en melding til Bob på en slik måte at Bob kan forsikre seg om at den virkelig er fra henne
- På forhånd har Alice fått generert et nøkkelpar Alicek1 og Alicek2
  - Alicek1 offentliggjør hun som sin *offentlige nøkkel*
  - Alicek2 holder hun strengt hemmelig som *privat nøkkel*
- Alice krypterer meldingen med Alicek2, altså sin egen private nøkkel
- Bob ser at meldingen trolig kommer fra Alice, og dekrypter med Alicek1, dvs. hennes offentlige nøkkel. Går det bra, kan han gå ut fra at meldingen virkelig kommer fra Alice.

Asymmetrisk krypteringsteknikk

- Vi ønsker at samme algoritme E brukes for både kryptering og dekryptering: dvs. $E_{k2}(E_{k1}(P)) \rightarrow P$
- Mest kjente algoritme: RSA (Rivest, Shamir & Adleman) (se detaljer neste lysark)
- RSA er basert på at det er lett å multiplisere to primtall, men vanskelig å faktorisere dem (bare empirisk vist!)
- se også http://en.wikipedia.org/wiki/Rsa

Multiplikasjon av to primtall er et eksempel på en matematisk enveisfunksjon:
Det er enkelt å beregne y = f(x),
og samtidig meget vanskelig å beregne $x = f^{-1}(y)$

Det er en matematisk utfordring å finne ut hvor "enveis" en gitt funksjon virkelig er!

Virkemåte av RSA-algoritmen

La meldingen P være et tall mindre enn N

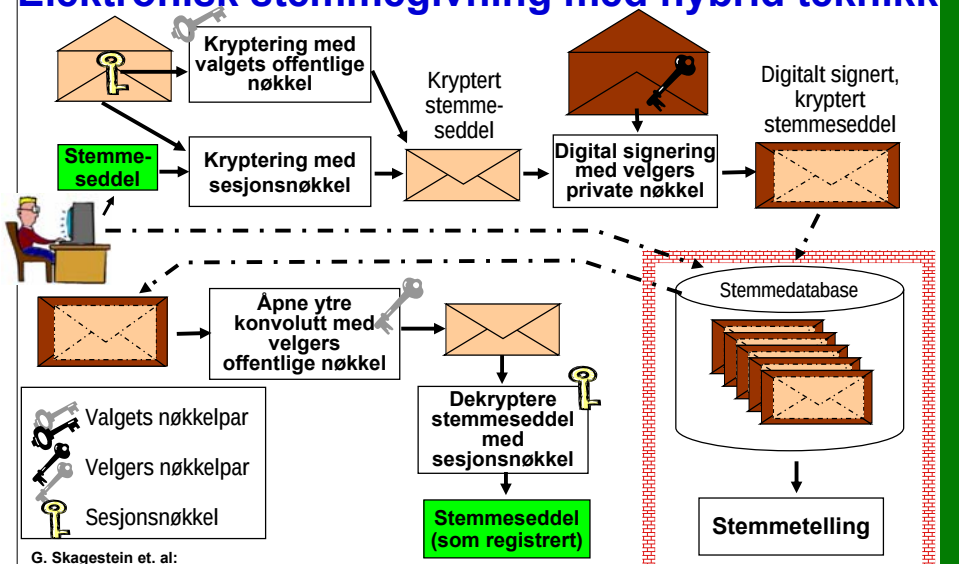
- Kryptering: $C = E_{k1}(P) = P^{k1} \% N$
- Dekryptering: $P = E_{k2}(C) = C^{k2} \% N$
- N og k1 er den offentlige nøkkelen
- k2 er den private nøkkelen
- N, k1 og k2 må være omhyggelig valgt etter følgende regler:
  - $N = p * q$, der p og q er store primtall
  - k2 er et stort tall relativt prim til $(p-1)(q-1)$ (dvs. ingen andre felles faktorer enn 1)
  - k1 er valgt slik at $k1*k2 \% (p-1)(q-1) = 1$
- Eksempel:
  - p = 3, q = 5, k2 = 7, k1 = 23, N = 15
  - Meldingen P er tallet 3:
  - Kryptering: $C = E_{23}(3) = 3^{23} \% 15 = 12$
  - Dekryptering: $P = E_7(12) = 12^7 \% 15 = 3$

Hvis p og q velges slik at N blir tilstrekkelig stor (mer enn 100 siffer), er det i praksis ikke mulig (selv med datamaskin) å finne k2, selv om du kjenner k1 og N.

Hybride teknikker

- Kombinasjon av symmetriske og asymmetriske teknikker
  – mye brukt i praksis
- Hvordan det virker:
  - velg en tilfeldig hemmelig engangsnøkkel
  - bruk en symmetrisk teknikk for å kryptere meldingen med denne hemmelige engangsnøkkelen
  - bruk en asymmetrisk teknikk for å kryptere den hemmelige engangsnøkkelen
  - send kryptert melding og engangsnøkkel til mottaker
- Gir god sikkerhet hvis engangsnøkkel velges tilfeldig

Elektronisk stemmegivning med hybrid teknikk



G. Skagestein et. al:
How to create trust in electronic voting over an untrusted platform.
In Krimmer, R. (Ed.): *Electronic Voting 2006*, GI Lecture Notes in Informatics, P-86, Bonn, 2006.

Hvordan distribuere offentlige nøkler?

- Hvorfor det ikke nytter å publisere dem i Aftenposten . . .
- En løsning: Digitale sertifikater (delegering av tillit)
  - Top Key Authority med kjent offentlig nøkkel (Verisign, Thawte, BT etc.)
  - Top Key Authority har som oppgave å sertifisere et underliggende nivå med Key Authorities x1, x2, …
  - . . . og så videre nedover i hierarkiet
  - En Key Authority sender deg et *sertifikat* i form av en melding kryptert med Key Authority private nøkkel
  - Sertifikatet bekrefter at bruker NN har offentlig nøkkel NNoff
- En mindre byråkratisk løsning: ”Web of trust”

Hvordan oppdage endring av meldinger

- Kryptering forhindrer at utenforstående kan lese dataene, men forhindrer ikke at utenforstående (inklusive støy på linjen) ubemerket endrer dataene
- Mottiltak er å la en algoritme h beregne et såkalt digitalt segl eller "message digest" s ut fra meldingen P.
  - Avsender beregner s = h(P) og overfører s sammen med P
  - Mottaker beregner s = h(P) en gang til og sammenlikner med den overførte s

Beregning av digitalt segl

- En "hash-funksjon" h kan beregne en tilsynelatende mystisk bit-sekvens (dvs. et tall) fra enhver melding P
- h(P) har alltid samme verdi for samme P
- Meldingen P kan bestå av navn, passord, dato, og selve klarteksten
- Ønskede egenskaper for en god hash-funksjon:
  - Alle biter i P inngår i beregningen av h(P)
  - Uansett innholdet av de aktuelle P gir h(P) en jevn fordeling over hele resultatområdet, dvs. alle verdier h(P) er like sannsynlige
  - Beregningen er rask

En god hash-funksjon

For den matematisk interesserte!

- h(P) kan beregnes som verdien av et polynom
  $(t_1 * x^n + t_2 * x^{n-1} + \ldots t_n * x^0) \% M$

  der koeffisientene $t1, t2 \ldots$ er de enkelte bytes i P

  x har verdien 256

  M er verdiområdet for h(P)
- Eksempel
  Forutsetter at tegnstrengen representeres i UTF-8:

  $h(\text{"UiO"}) = (55_{16} * 256^2 + 69_{16} * 256^1 + 4F_{16} * 256^0) \% 2^{16}$
  $= (85 * 256^2 + 105 * 256^1 + 79 * 256^0) \% 2^{32} = 5597519 \% 65536 = 26959$

Metode for sikker meldingsformidling

- Senderen må
  - Beregne et digitalt segl for meldingen
  - Kryptere seglet med egen privat nøkkel – gir digital signatur
  - Kryptere meldingen med en tilfeldig valgt nøkkel (sesjonsnøkkel)
  - Kryptere sesjonsnøkkelen med mottakerens offentlige nøkkel
- Mottakeren må
  - Dekryptere sesjonsnøkkelen med egen privat nøkkel
  - Dekryptere meldingen med sesjonsnøkkelen
  - Dekryptere signaturen med avsenders offentlige nøkkel – gir seglet
  - Beregne seglet på nytt og sammenlikne med det oversendte seglet

Steganografi

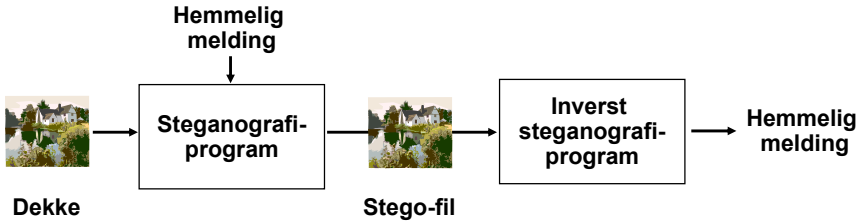
- fra gresk
  - stegano: skjult, dekket, å dekke
  - graphos: skriving
- Steganografi: A gjemme en representasjon av noe i en representasjon av noe annet – et såkalt dekke (”cover”)

Mitt ønske til mor er glemt,
kan lillebror se etter katten selv?

For et mer avansert eksempel (bilde skjult i bilde), se for eksempel http://niels.xtdnet.nl/stego/abc.html

Dekket

- Dekket er som oftest et bilde eller en lyd (mange biter!)
- Den beste måten å holde noe hemmelig er å holde det hemmelig at det er noe å holde hemmelig
  → dekket må ikke endres påfallende gjennom at noe er skjult



Et eksempel på steganografi

- Steganografi med ” Digital Invisible Ink Toolkit ”

se http://diit.sourceforge.net/index.html

Originalbilde	Dekke	Stegobilde	Gjenvunnet bilde
7 kb jpeg	40 kB jpeg	332 kB png	7 kB jpeg

Vannmerking

- Utvide en tekst-/bilde-/lydfil med ekstra data som ikke kan la seg fjerne uten å ødelegge originalen
- Deler av vannmerket kan være observerbart i teksten/bildet/lyden, men vanligvis er det ikke observerbart ved direkte observasjon
- Brukes for tilleggsopplysninger, som rettighetshaver, copyrightmarkering og eksemplarnummerering.
- Et vannmerke skal ikke kunne fjernes uten å ødelegge tekst/bilde/lyd
- Vannmerking ligner teknisk sett på steganografi, hovedforskjellen er intensjonen

Hvordan gjemme data

- ”Trikset” er å gjemme det hemmelige bitmønsteret på bestemte steder i bitmønsteret for dekket
- Eksempel:
  - Et bilde (dekke) er representert som en sekvens av RGB-verdier, hver på 24 bit.
  - Vi skal gjemme tegnet ”A”, Unicode 41 (hex) = 0100 0001
  - Vi sørger for at den siste biten i for eksempel hver 64de RGB-verdi har verdiene 0100 0001
  - Av og til fører dette til at RGB-verdien blir endret, av og til ikke
  - En eventuell endring i intensitet i blått hist og her vil antagelig ikke være synlig
- Variasjoner:
  - Endre i R- eller G-verdi istedenfor
  - Endre i de minst signifikante bitplanene i et bilde
  - Endre hvor det allerede er store endringer (bildederivasjon!)
  - Endre i andre verdiområder (for eksempel frekvensdomenet)
  - Manipulere fargetabellen for bildeformater med fargetabell

Robusthet

Men farer truer det skjulte budskap:

- Klipping av bilde og lyd
- Formatkonvertering
- Komprimering

Derfor:

- Metoden med å bruke de minst signifikante bitene er meget tvilsom hvis dekket kan bli utsatt for komprimering basert på at uvesentlige data kastes vekk
- Vannmerker bør dupliseres og spres over hele dekket

Stegoanalyse

- Å konstatere at et dekke inneholder et skjult budskap
  – og eventuelt gjenvinne budskapet
- Vanskelig å gjøre ved direkte observasjon
- Analyseprogrammer er mye mer effektive
  – er det noe uvanlig i teksten/bildefilen/lydfilen?
- Tilgang til dekket letter stegoanalysen betraktelig
  – bruk ikke kjente digitale bilder som ligger på Internett som dekke!

Oppsummering

- Data kan gjøres uleselige ved hjelp av kryptering
- Urettmessig endring av data kan oppdages med digitale segl
- Avsender kan verifiseres ved hjelp av asymmetrisk kryptering
- Kobling mellom individer og nøkler gjøres med sertifikater
- Ved kryptering brukes oftest en kombinasjon av symmetriske og asymmetriske krypteringsteknikker
- Vurderinger av sikkerheten mot ”knekking” av krypteringer er kun basert på antagelser og empiri, intet er bevist
- Steganografi brukes for å skjule en melding i et dekke
- Vannmerker brukes for å gi tilleggsopplysninger som ikke kan fjernes uten å ødelegge dekket