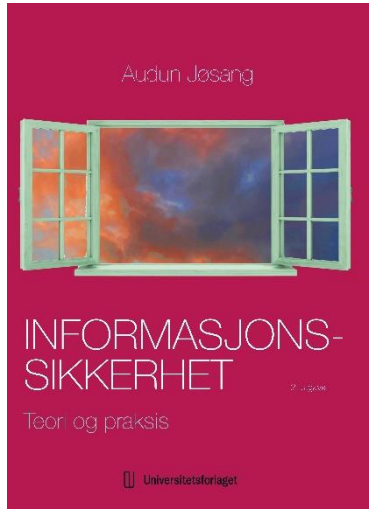
Kapittel 4: Kryptografi

Informasjonssikkerhet: Teori og praksis

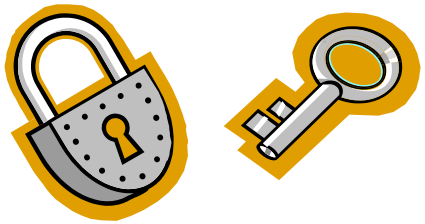
Audun Jøsang

2. utg. 2023

Universitetsforlaget

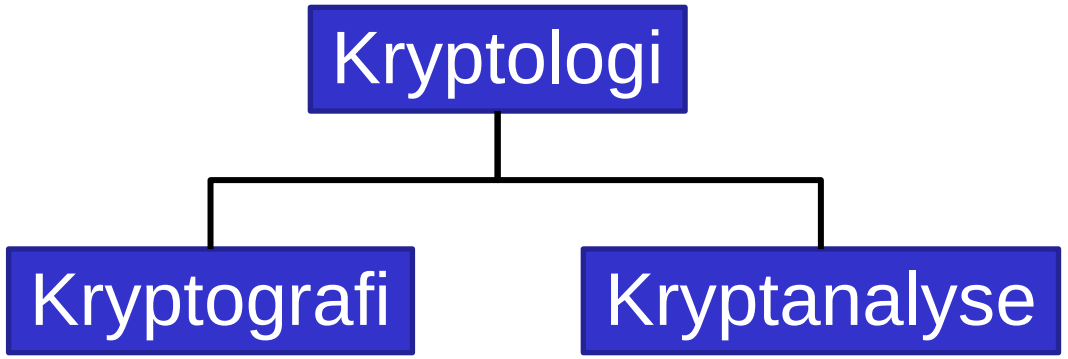


Oversikt



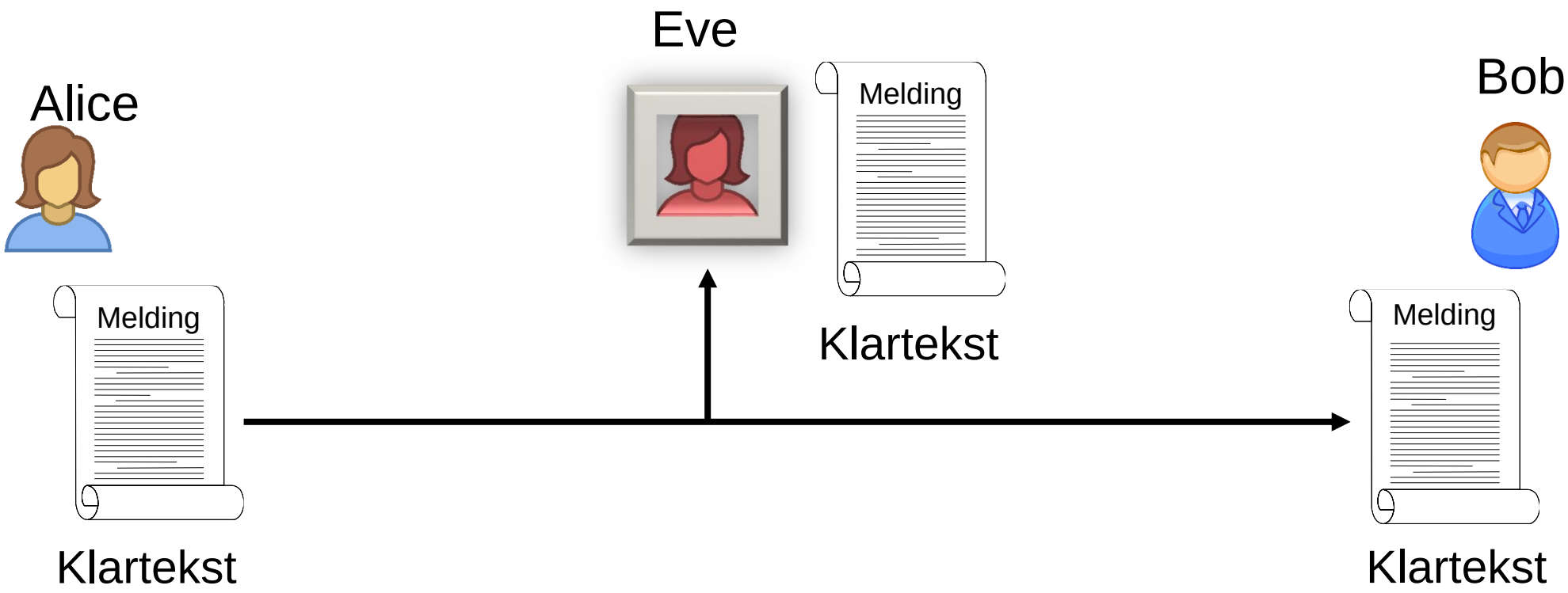
- Hva er kryptografi?
- Symmetrisk kryptografi
  - Kort kryptohistorikk
  - Strømchiffer
  - Blokkchiffer
  - Hashfunksjoner
- Asymmetrisk kryptografi
  - Asymmetrisk kryptering generelt
  - Diffie-Hellman nøkkelbytte
  - Digitale signaturer
- Post-Quantum Crypto

Terminologi

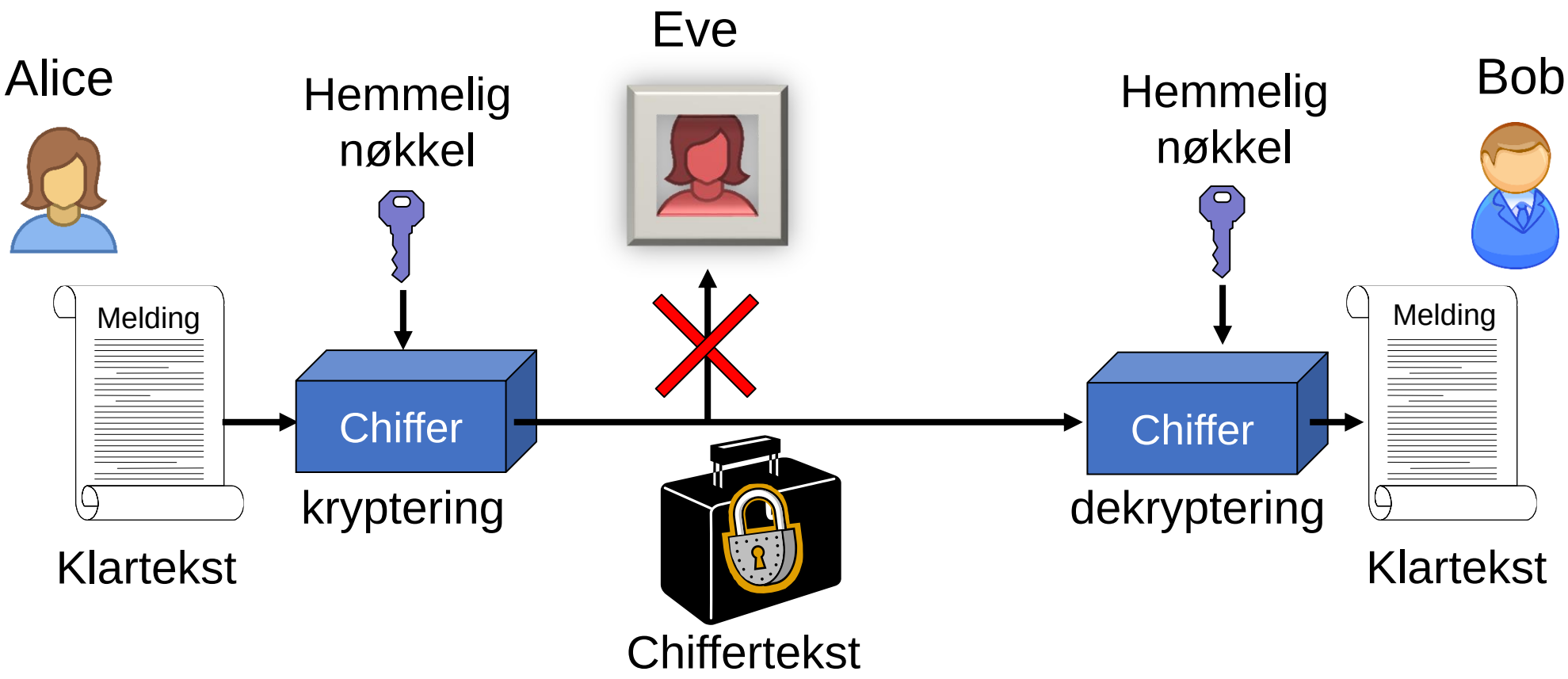


- **Kryptografi** er vitenskapen om hemmelig skrift med det formål å skjule betydningen av en melding.
- **Kryptanalyse** er vitenskapen om å knekke kryptografi.
- **Kryptologi** dekker både kryptografi og kryptanalyse.

Kryptografi i et blikk



Kryptografi i et blikk



Hva kan kryptografi brukes til?

- Kryptografi støtter følgende sikkerhetsmål:
  - **Konfidensialitet:**
    - Gjør data uleselige for enheter som ikke har de riktige kryptografiske nøklene, selv om de har dataene.
  - **Dataintegritet:**
    - Enheter med riktige kryptografiske nøkler kan bekrefte at data er korrekt og ikke er blitt endret av uautoriserte.
  - **Autentisering:**
    - Entiteter som kommuniserer kan få visshet om at identiteten til den andre brukeren/entiteten eller avsenderen av en melding er det den påstår å være.
  - **Digital Signatur og PKI (Public-Key Infrastructure):**
    - Sterkt bevis på dataautentisitet som kan verifiseres av tredjeparter.
    - Skalerbar (til hele Internett) sikker distribusjon av kryptografiske nøkler.

Kryptografiske funksjoner

Kryptografiske funksjoner

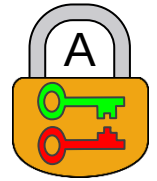
Krypterings-algoritmer

Hash-funksjoner

Også kalt *"offentlig-nøkkel kryptografi"*

Symmetriske chiffer
Samme hemmelig nøkkel brukes for både kryptering og dekryptering

Asymmetriske algoritmer
Bruker nøkkelpar med offentlig nøkkel og privat nøkkel



Blokkchiffer

Strømchiffer

Kryptering
Offentlig nøkkel brukes for kryptering og privat nøkkel brukes for dekryptering

Digital signatur
Privat nøkkel brukes for signering og offentlig nøkkel brukes for validering

Terminologi

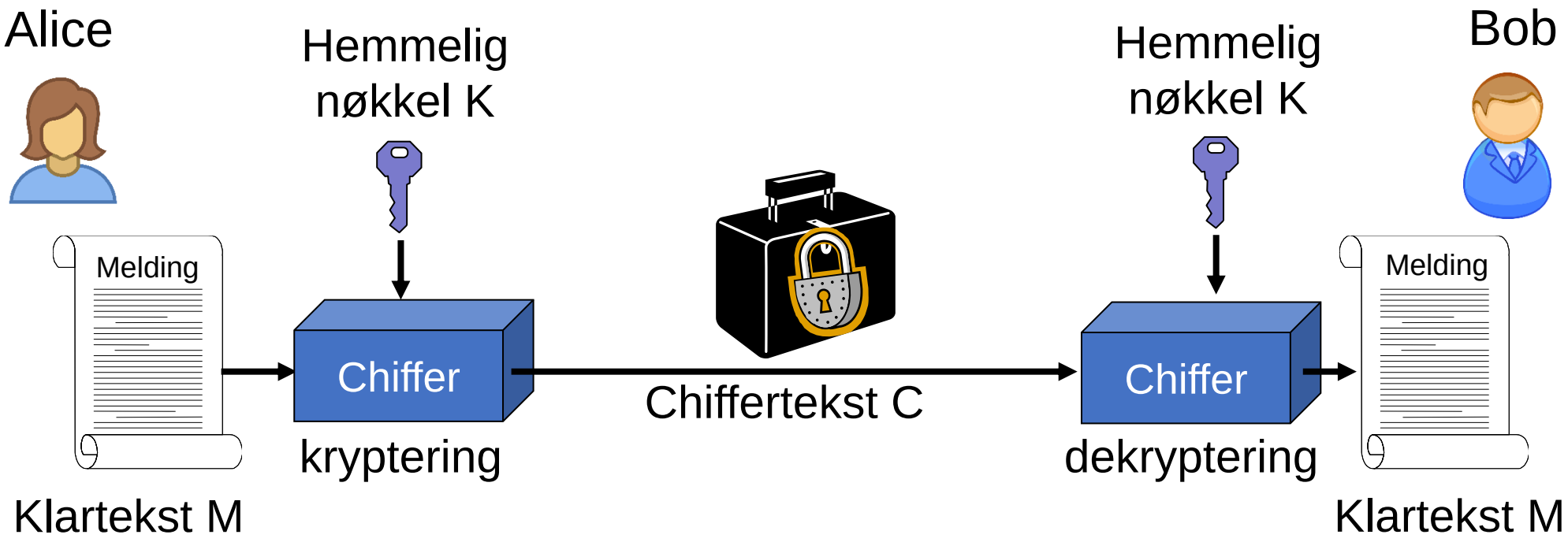
- Kryptering: klartekst *M* transformeres med krypteringsfunksjon E til chiffertekst *C* styrt av krypteringsnøkkel *k*.
  - Formell skrivemåte: $C = E(M, k)$.
- Dekryptering: chiffertekst *C* transformeres med dekrypteringsfunksjon D til klartekst *M* styrt av krypteringsnøkkel *k*.
  - Formell skivemåte: $M = D(C, k)$.
- Symmetrisk chiffer: samme hemmelige nøkkel brukes både for kryptering og dekryptering.

- Asymmetrisk chiffer: Nøkkelpar med en privat og en offentlig nøkkel.
  - Kryptering med offentlig nøkkel og dekryptering med privat nøkkel
  - Digital signatur med privat nøkkel og verifisering av signatur med offentlig nøkkel

Privat nøkkel

Offentlig nøkkel

Symmetrisk kryptering (med hemmelig nøkkel)



M ⟶ E(M,K) ⟶ C ⟶ D(C,K) ⟶ M

- "Hemmelig nøkkel" betyr at nøkkelen er delt *i hemmelighet* mellom alle entiteter som er autorisert til å kryptere/dekryptere.

Kryptografiens historie

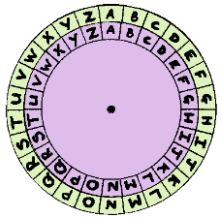
Klassiske chiffer

Transposjon

Scytale

Substusjon

Cæsar-chiffer



Middelalder chiffer

Poly-alfabetisk substitusjon og transposisjon

Vigenère 1566

Pre-WW2 chiffer

Engangs-nøkkel

Vernam 1916

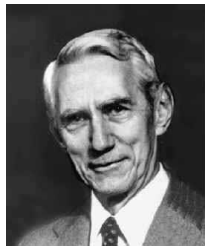
WW2 chiffer

Kompleks mekanikk

Enigma

SP-nettverk Info-teori

Shannon



Pre-2000 chiffer

DES

Feistel

Asymmetrisk krypto

Diffie Hellman

Post-2000 chiffer

AES

Rijmen & Daemen

SHA-3

Post-kvante asymmetrisk krypto

→ BC | AD → 1799 | 1800 → 1939 | 1940 → 1975 | 1976 → 2000 | 2001 →

Styrken av et chiffer

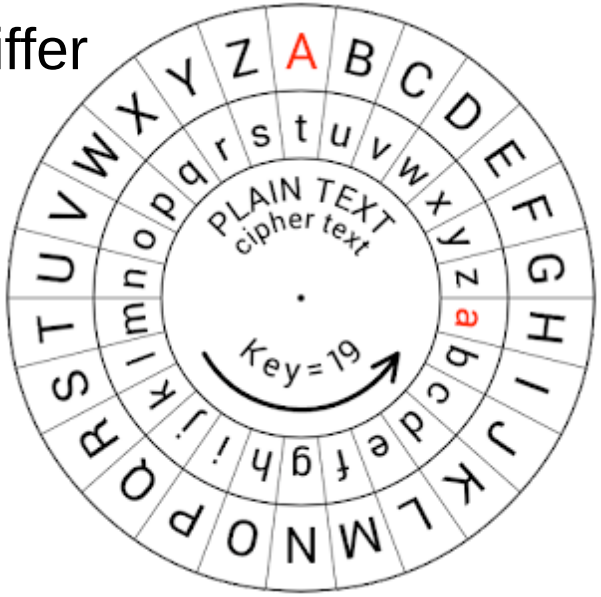
Faktorer som bestemmer chifferstyrke:

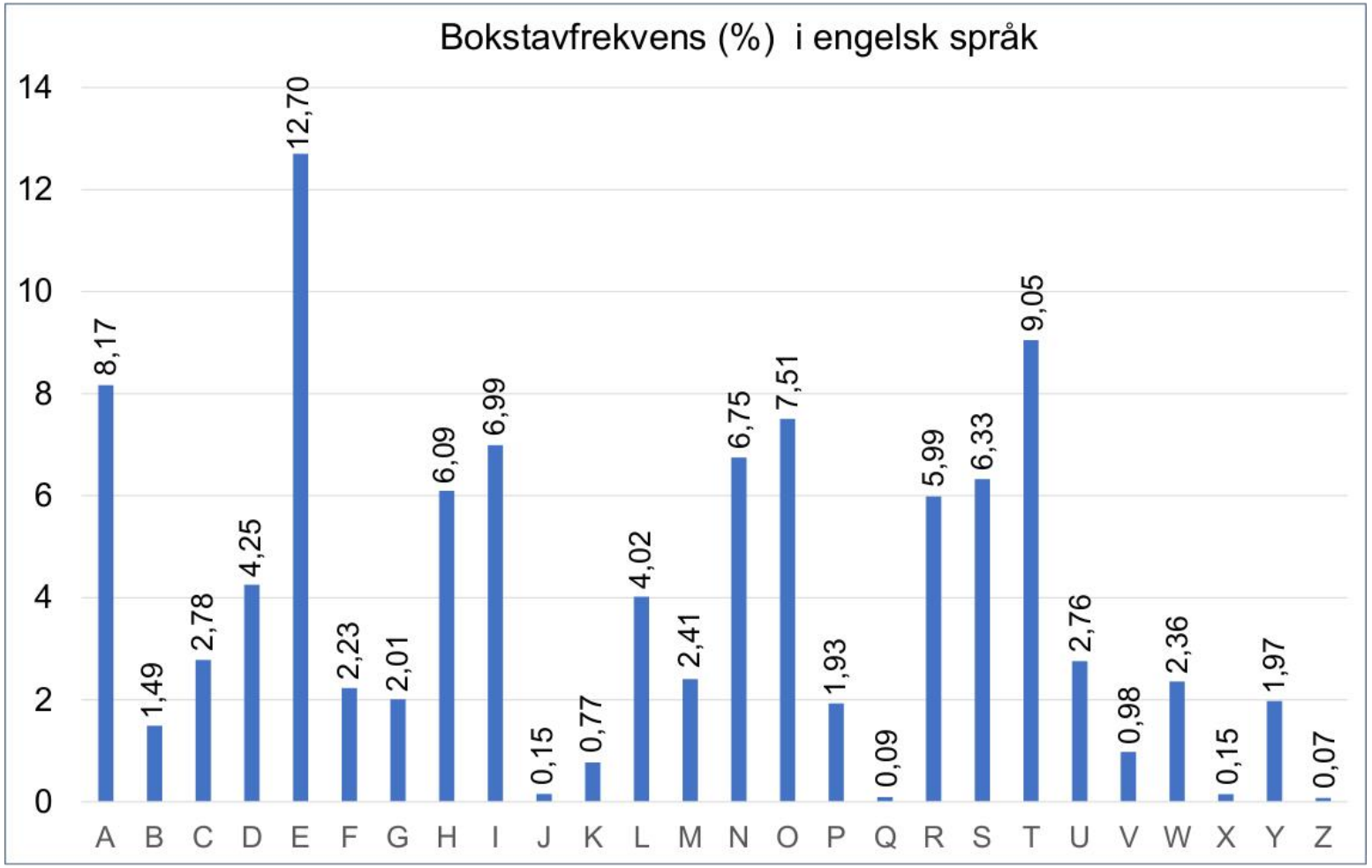
- **Nøkkelstørrelse**.
  - Nødvendig tid for fullstendig søk blant alle nøkler avhenger av størrelse.
  - Typisk størrelse for symmetrisk blokkchiffer er 256 bits.
  - Angriper må gjennomsnittlig prøve $2^{256}/2$ forskjellige nøkler for å finne den riktige, noe som ville ta millioner av år og som derfor ikke er praktisk.
  - Hvis det fins N ulike nøkler vil nøkkelstørrelse være: $\log_2(N)$.
- **Algoritmens styrke**.
  - Å finne nøkkelen ved kryptanalyse kan utnytte statistiske ujevnheter i chifferteksten.
  - For å forhindre kryptanalyse bør bitmønstrene / tegnene i chifferteksten ha en uniform/jevn fordeling, det vil si at alle bitmønstre / tegn skal være like sannsynlige.

Bokstavfrekvenser→ Statistisk kryptanalyse

- Klassiske chiffer, som Cæsar-chifferet, er svake fordi de ikke klarer å skjule statistiske ujevnheter i chifferteksten.

Cæsar-chiffer

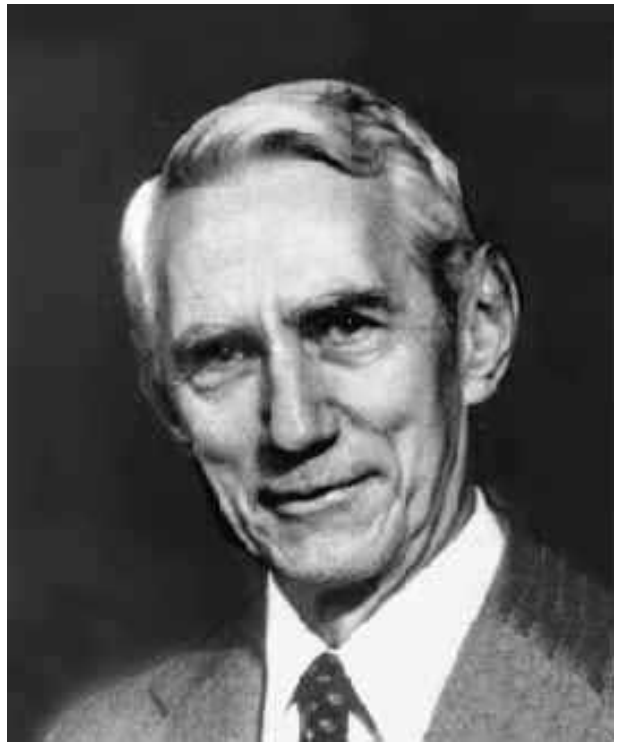




Claude Shannon (1916 – 2001)

Informasjonsteoriens far – MIT / Bell Labs

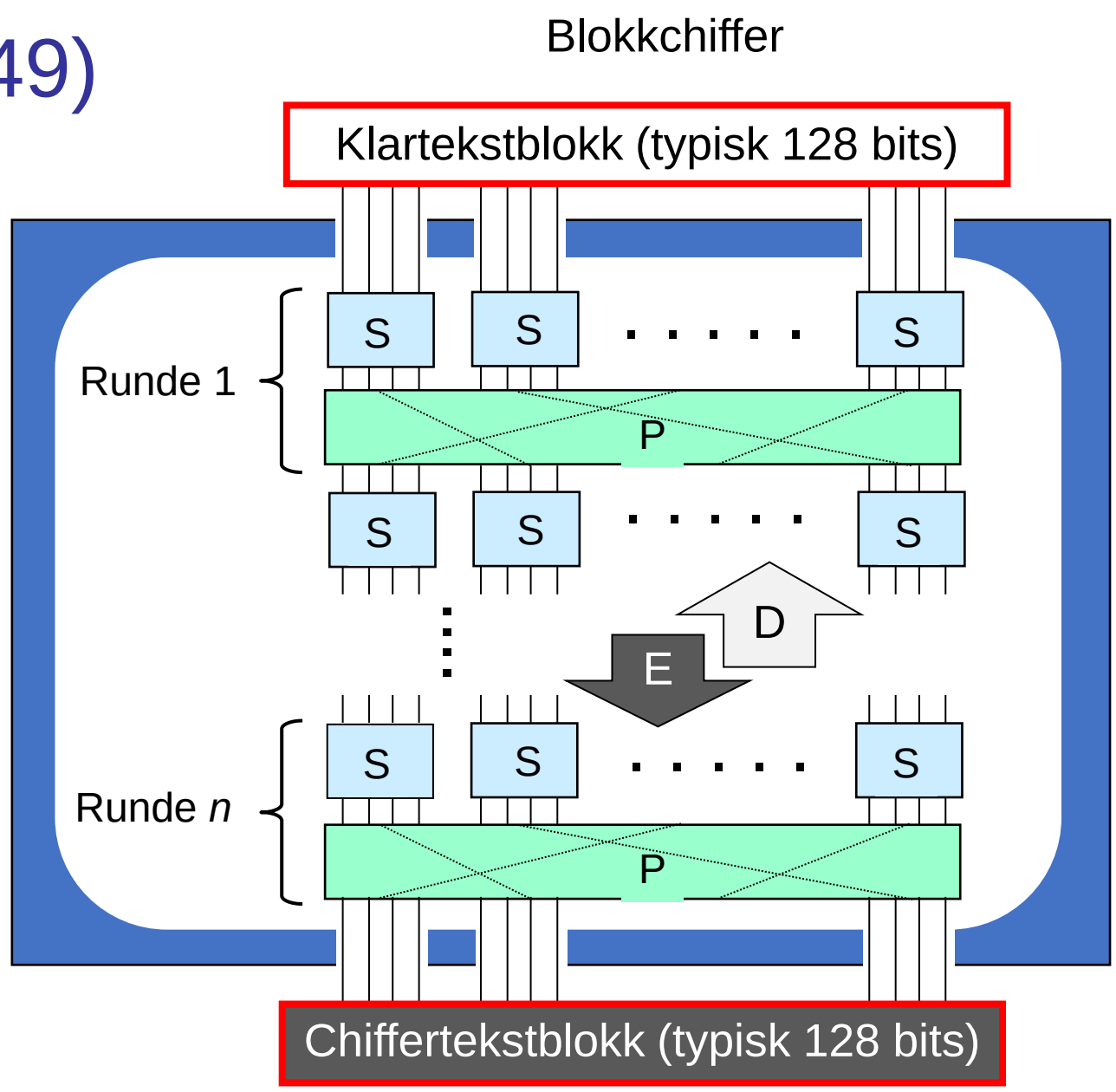
- Informasjonsteori
  - Definerte "binært siffer" (bit) som minste informasjonsenhet
  - Definert informasjonsentropi som mål på mengden informasjon
- Kryptografi
  - Modell for hemmelighetssikre systemer
  - Definert perfekt hemmelighetssikkerhet (Kommer senere)
  - Prinsipp for kryptering med SP-nettverk (substitusjon og permutasjon) for å viske ut statistiske ujevnheter



Shannon's SP-nettverk (1949)

Visker ut statistiske ujevnheter

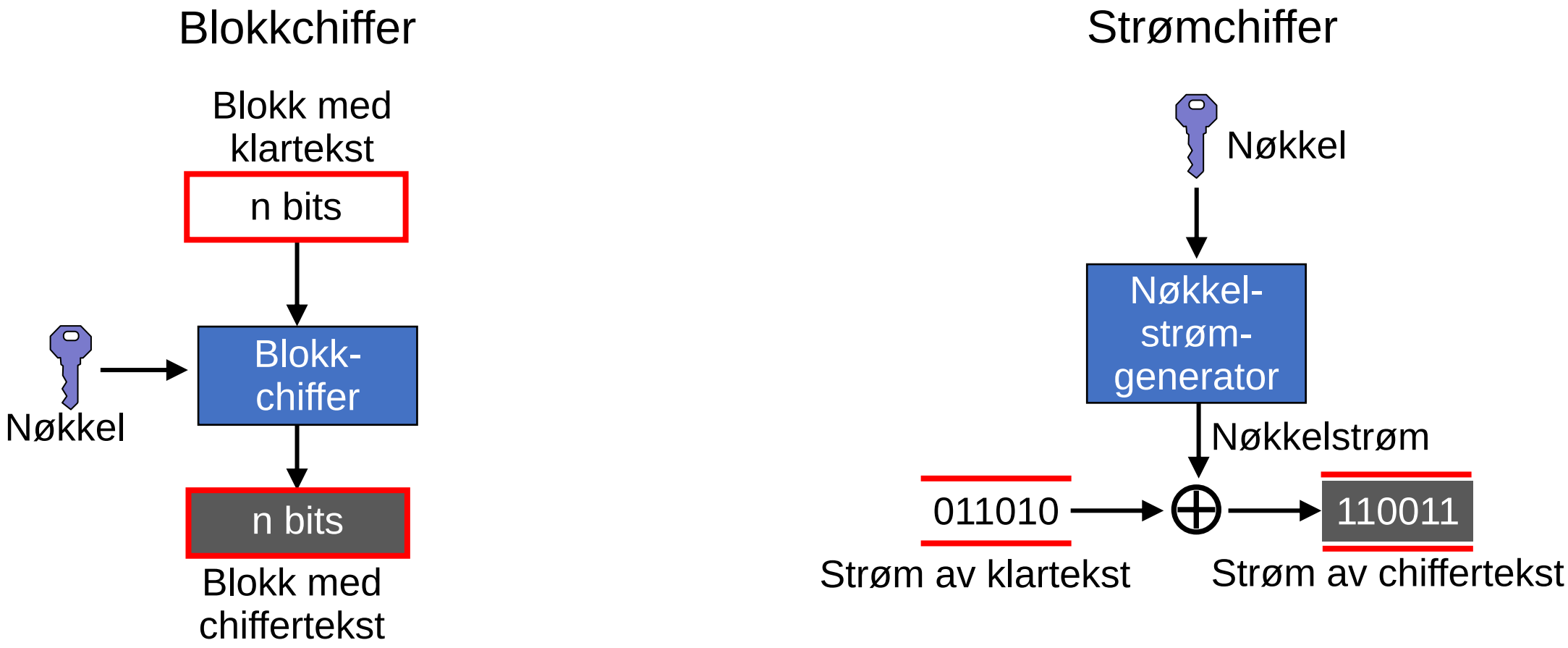
- Repeter substitusjon og permutasjon et tilstrekkelig antall ganger, typisk 10-20 runder.
- Substitusjon
  - Klartektsblokk deles opp i sub-blokker
  - Substitusjon av bits i hver sub-blokk f.eks. 0001 substitueres med 0110
  - Gir «confusion» dvs. skjuler sammenheng mellom klartekstblokk og chiffertekstblokk.
- Permutasjon
  - Sub-blokker flyttes rundt omkring i blokken.
  - Gir "diffusion", dvs. at endring av en enkelt klartekstbit (eller nøkkelbit) forårsaker endring av mange chiffertekstbits.
- Nøkkelen inngår i S eller P eller i separat funksjon



AES - Advanced Encryption Standard

- DES (Data Encryption Standard) fra 1977 hadde en 56-bits nøkkel og en 64-bits blokk. På midten av 1990-tallet kunne DES bli knekket med fullstendig nøkkelsøk.
- I 1997 kunngjorde NIST en åpen konkurranse om å designe et nytt blokkchiffer for å erstatte DES.
- Det beste forslaget kalt "Rijndael" (designet av Vincent Rijmen og Joan Daemen fra Belgia) ble ansett som best, og nominert til å bli AES (Advanced Encryption Standard) i 2001.
- AES har nøkkelstørrelser på 128, 192 eller 256 bit og blokkstørrelse på 128 bit.
- AES følger Shannons prinsipp med SP-nettverk.

Blokkchiffer og strømchiffer



Blokkchiffer: Operasjonsmoduser

- Et blokkchiffer krypterer en blokk på (typisk) 128 bits, som bare er omtrent 16 bokstaver.
- For kryptering av mer enn en blokk benyttes en spesifikk modus.
- Krypteringsmodusene har ulike egenskaper.
- Vanlige moduser er:
  - **C**oun**T**e**R** Mode (CTR)
  - **C**ipher **B**lock **C**haining (CBC)
  - **O**utput **F**eed**B**ack (OFB)
  - **C**ipher **F**eed**B**ack (CFB)

sikker

  - **E**lectronic **C**ode **B**ook (ECB)

usikker

Electronic Code Book (ECB)

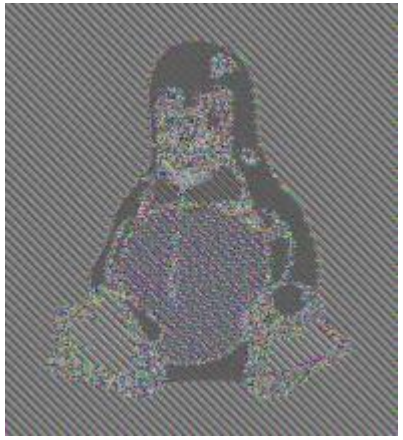
- Enkleste krypteringsmodus
- Klarteksten deles opp i blokker $M_1, M_2, \ldots, M_n$
- Hver blokk krypteres separat.
  - Notasjon kryptering: $C_1 = \mathrm{E}(M_1, K)$
  - Notasjon dekryptering: $M_1 = \mathrm{D}(C_1, K)$
  - Like klartekstblokker gir like chiffertekstblokker, dette er problemet !

Kryptering

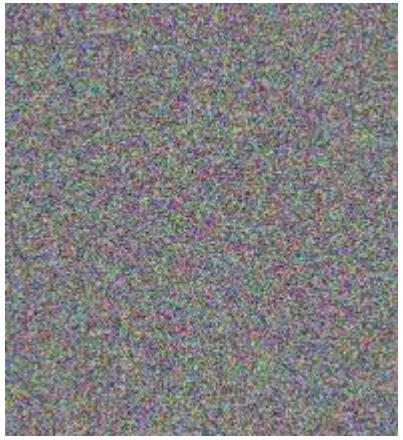
Dekryptering

Sårbarhet ved å bruke ECB-modus

Klartekst



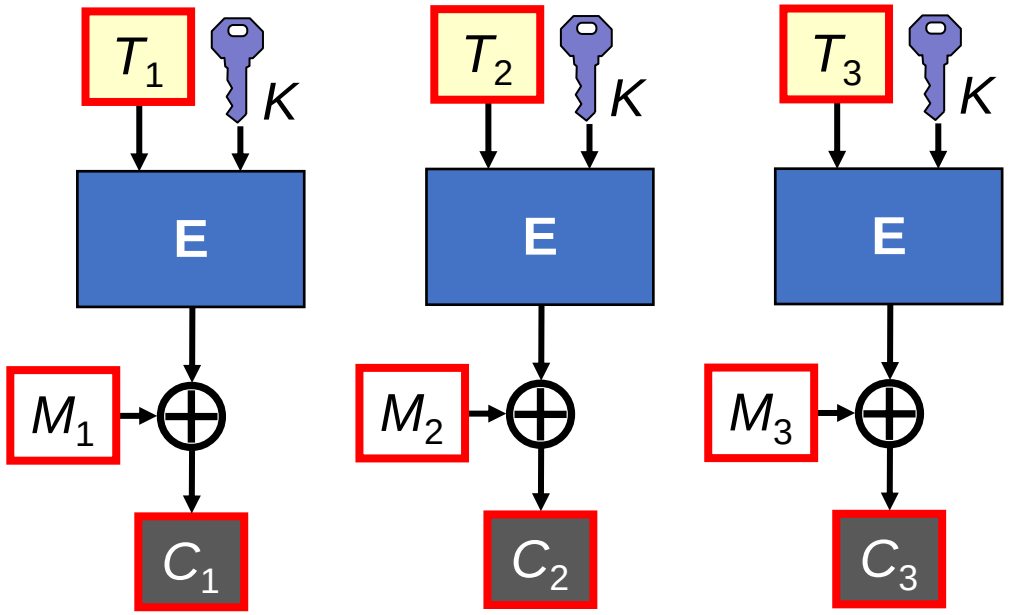
Chiffertekst med ECB-modus



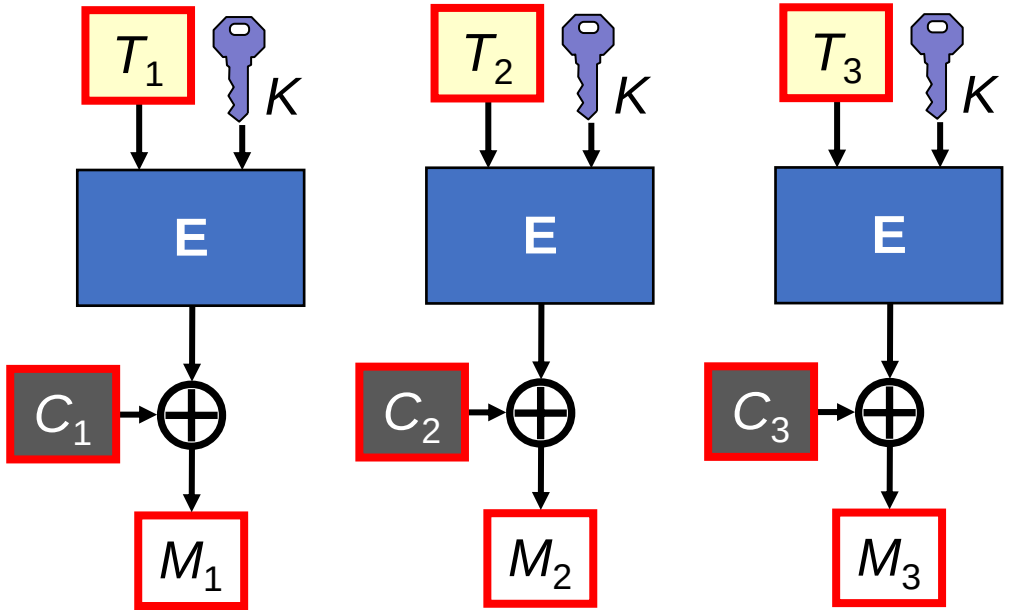
Chiffertekst med sikker modus

Tellermodus - Counter Mode (CTR)

- Klarteksten deles opp i blokker $M_1, M_2, \ldots, M_n$
- En inkrementerende tellerverdi T krypteres
- Hver krypterte telleverdi adderes til klartekstblokken med binær XOR $\oplus$
  - Like klartekstblokker gir forskjellige chiffertekstblokker, dette gir sikkerhet !



Kryptering



Dekryptering

CTR-kryptering og binæraddisjon med XOR

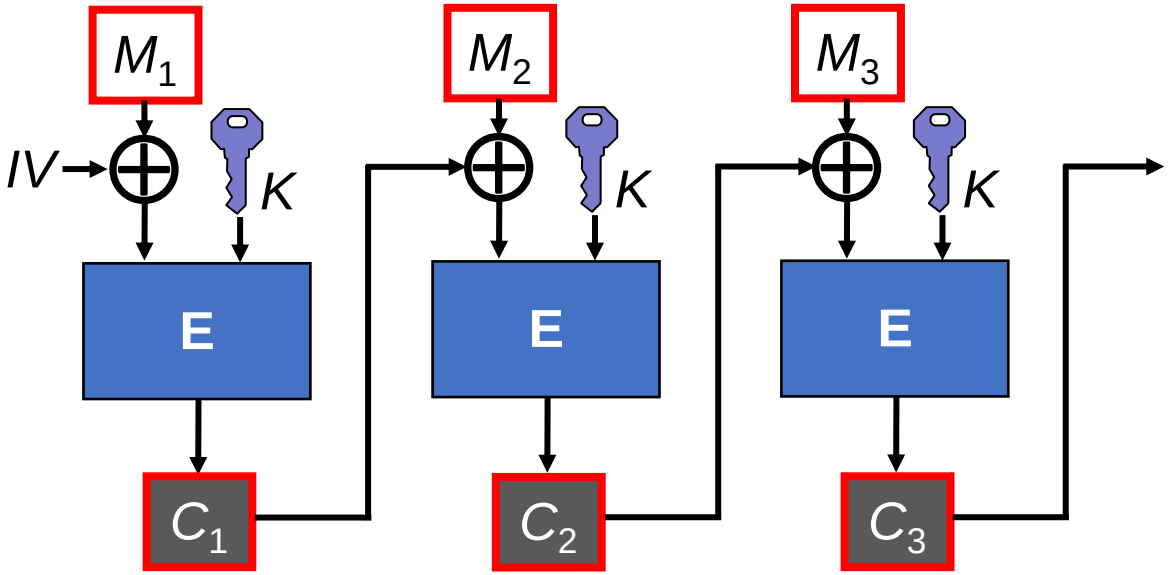
- Klarteksten deles opp i blokker: $M_1, M_2, \ldots, M_n$
- Inkrementerende tellerverdier med samme blokkstørrelse: $T_1, T_2, \ldots, T_n$
- Tellerverdiene krypteres og adderes til klartekstblokkene:
  - Notasjon kryptering: $C_1 = \mathrm{E}(T_1, K) \oplus M_1$
  - Notasjon dekryptering: $M_1 = \mathrm{E}(T_1, K) \oplus C_1 = \mathrm{E}(T_1, K) \oplus \mathrm{E}(T_1, K) \oplus M_1$
  - Krypteringsfunksjonen $\mathbf{E}$ benyttes til både kryptering og dekryptering
- Binæraddisjon med XOR $\oplus$
  - Addisjon av bit med seg selv gir alltid null

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

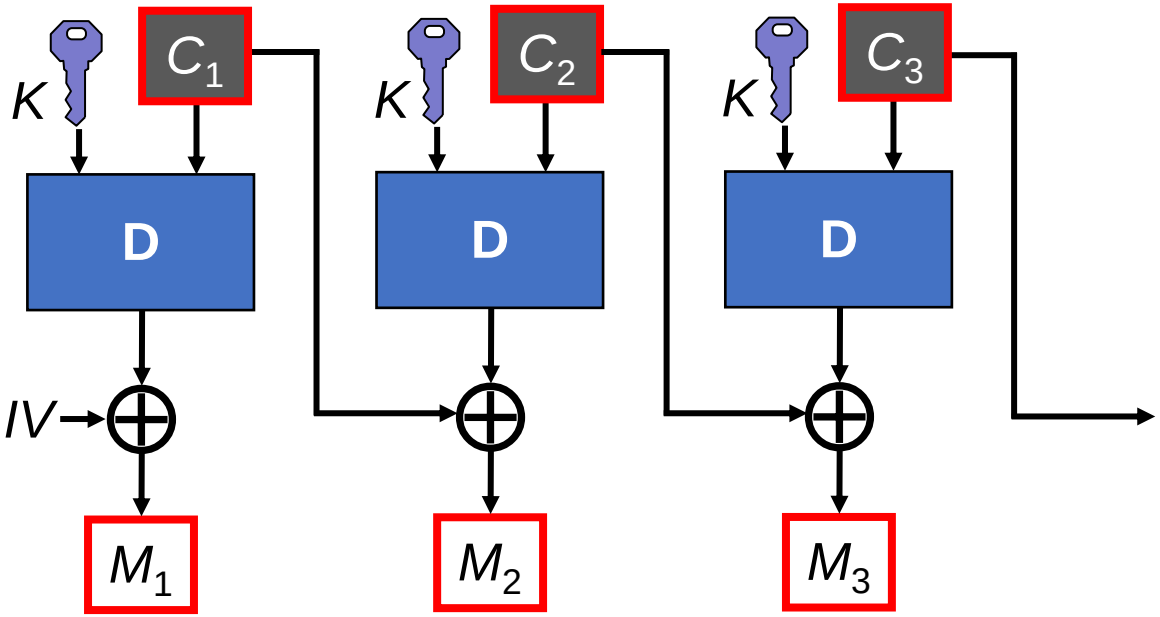
- Eksempel kryptering og dekryptering: $M_1 = 1111$ $\quad$ $\mathrm{E}(T_1, K) = 1001$
  - Kryptering: $C_1 = 1001 \oplus 1111 = 0110$
  - Dekryptering: $M_1 = 1001 \oplus 0110 = 1111$

Cipher Block Chaining (CBC)

- Klarteksten deles opp i blokker $M_1, M_2, \ldots, M_n$
- Hver chifferblokk adderes til neste klartekstblokken med binær XOR ⊕ før kryptering
  - Starter med en tilfeldig *IV* (initialiseringsvektor) som ikke trenger å være hemmelig
  - Like klartekstblokker krypteres til forskjellige chiffertekstblokker, dette gir god sikkerhet !



Kryptering



Dekryptering

Notasjon for CBC

- Klarteksten deles opp i blokker: $M_1, M_2, \ldots, M_n$
- Hver chifferblokk adderes til neste klartekstblokk med binær XOR før kryptering

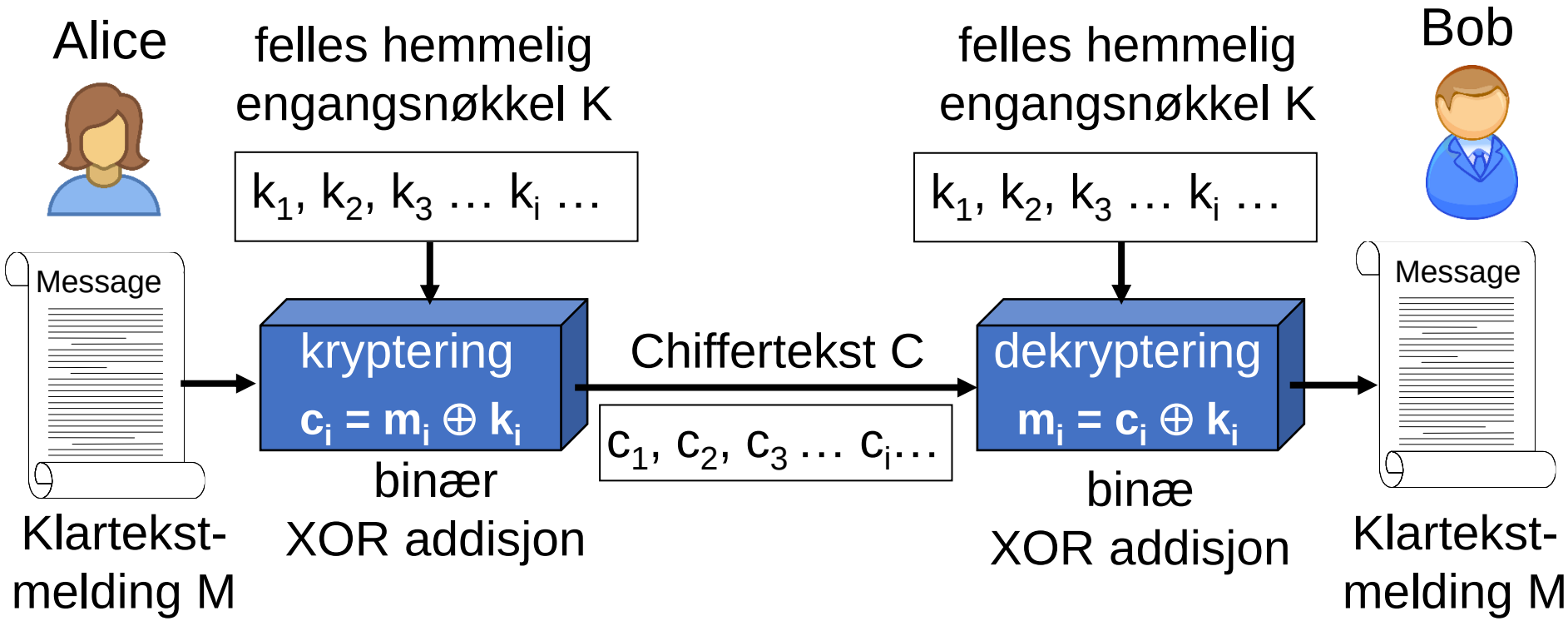
  – Notasjon kryptering:

$$C_1 = \mathrm{E}((IV \oplus M_1), K)$$
$$C_2 = \mathrm{E}((C_1 \oplus M_2), K)$$
$$C_n = \mathrm{E}((C_{n\text{-}1} \oplus M_n), K)$$

  – Notasjon dekryptering:

$$M_1 = \mathrm{D}(C_1, K) \oplus IV$$
$$M_2 = \mathrm{D}(C_2, K) \oplus C_1$$
$$M_n = \mathrm{D}(C_n, K) \oplus C_{n\text{-}1}$$

Engangsnøkkel: One-Time Pad (Gilbert Vernam, 1917)

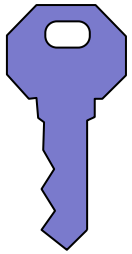


- Bit for bit binær XOR addisjon: $c_i = m_i \oplus k_i$
  $m_i = c_i \oplus k_i = m_i \oplus k_i \oplus k_i = m_i \oplus 0 = m_i$
- OTP gir perfekt sikkerhet ved å anta at OTP-nøkkelen er helt tilfeldig, av samme lengde som meldingen, og brukes bare en gang.

Den perfekte chiffermaskinen: One-Time-Pad

- Teleks med OTP på hullbånd, produsert av STK på Økern
- Moderne versjoner kan bruke DVDer med Gbyte med tilfeldige data

HASHFUNKSJONER OG MELDINGSAUTENTISERING

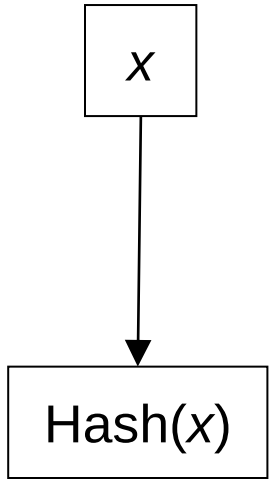


Hashfunksjoner

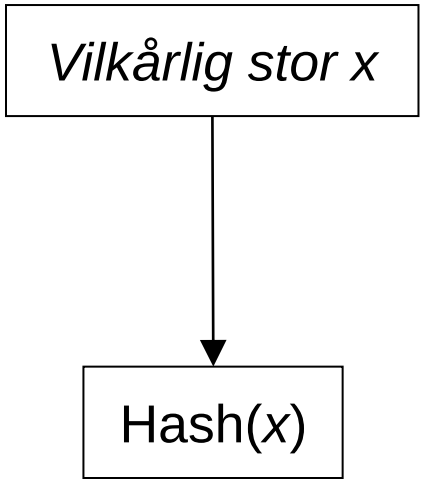
Krav til en hashfunksjon Hash:

1. Lett å beregne: Gitt inputdata *x*, det skal være lett å beregne Hash(*x*).
2. Komprimering: Komprimerer vilkårlig stor *x* til en hash-verdi Hash(*x*) med fast størrelse *n* (typisk 256 bits eller 512 bits).
3. Enveis: Gitt hash-verdi *y*, det skal være praktisk umulig å finne inputdata *x* slik at Hash(*x*) = *y*.
4. Kollisjonsresistens (svak): Gitt inputdata *x* og tilhørende hash-verdi Hash(*x*), det skal være praktisk umulig å finne et annet datasett *x'* slik at Hash(*x*) = Hash(*x'*) (svak kollisjonsresistens).
5. Kollisjonsresistens (sterk): Det skal være praktisk umulig å finne to ulike datasett *x* og *x'* slik at Hash(*x*) = Hash(*x'*) (sterk kollisjonsresistens).

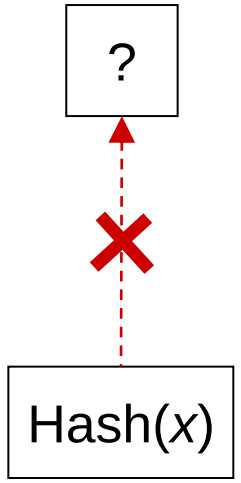
Egenskaper ved hashfunksjoner



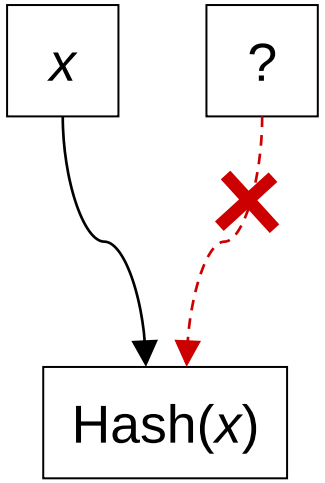
Lett å beregne



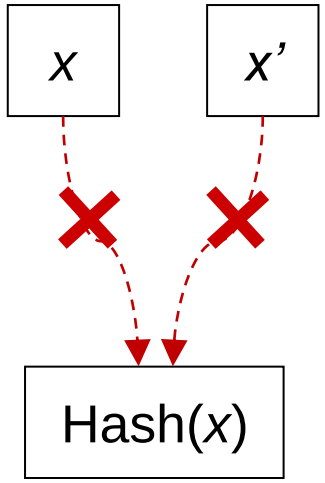
Komprimering til fast størrelse



Enveis-funksjon



Svak kollisjons-resistens



Sterk Kollisjons-resistens

Velkjente hashfunksjoner

- **MD5** (1991): 128 bits hash-verdi. Lett å finne kollisjoner, på grunn av liten hash-størrelse og dårlig design. Skal ikke lenger brukes.

- **SHA-1** (Secure Hash Algorithm):160 bits hash-verdi. Designet av NSA i 1995. Relativt lett å finne kollisjoner. Skal ikke lenger brukes, men forekommer i fremdeles eldre applikasjoner.

- **SHA-2** designet av NSA i 2001. Kan generere 256, 384 og 512 bits hashverdi. Betraktes som sikker. Erstatning for SHA-1.

- **SHA-3:** designet av Joan Daemen + andre i 2010. Standardisert i 2015. Kan generere: 256, 384, og 512 bits hashverdi. SHA-3 har liten bruk, fordi SHA-2 fremdeles regnes som sikker.

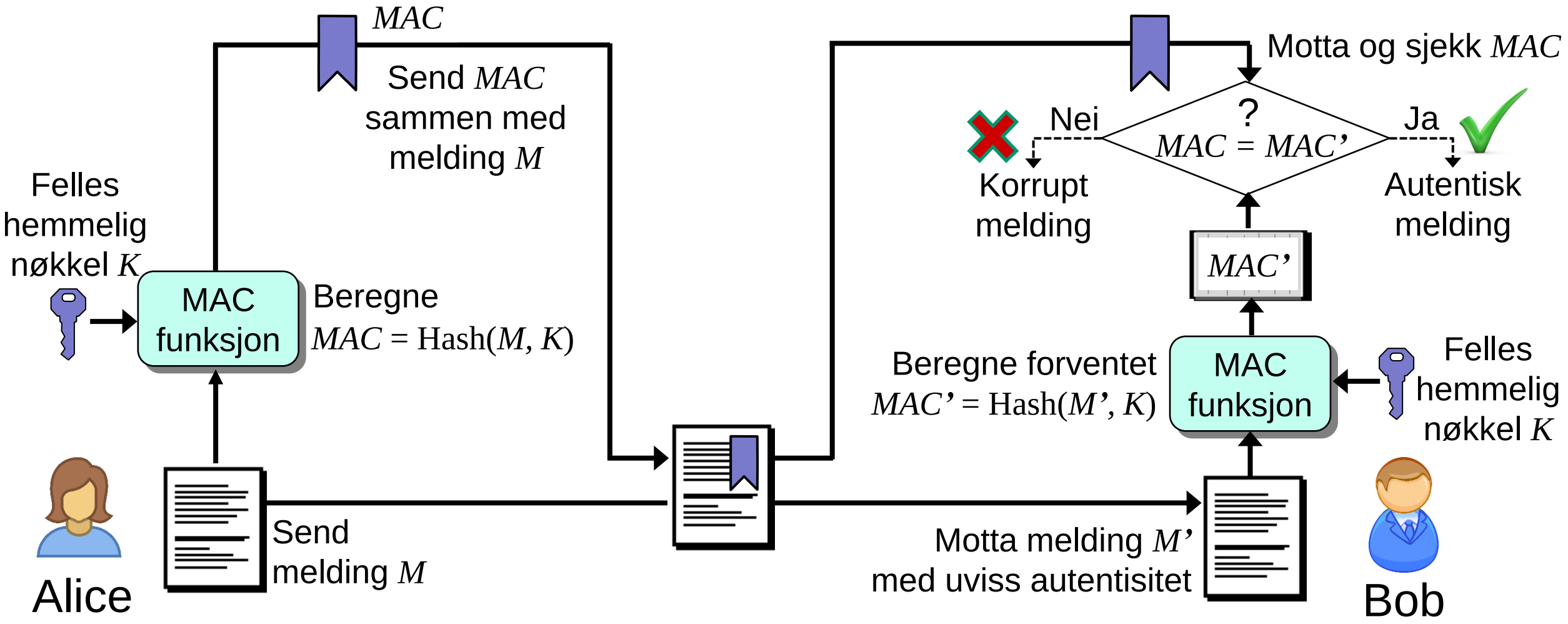
Anvendelser av hashfunksjoner

- Sammenligning av filer
- Beskyttelse av passord
- Integritetssjekk, f.eks. av SW-distribusjoner
- Generering av meldingsautentiseringskoder (MAC)
- Digitale signaturer
- Bitcoin og kryptovaluta
- Generering av pseudotilfeldige tall
- Generering av kryptonøkler

Meldingsautentiseringskode – MAC (Message Authentication Code)

- En melding *M* med en enkel hash-verdi Hash(*M*) kan lett endres av angriper.
- For å hindre angrep er det nødvendig å bruke en autentisert hash-verdi.
- MAC (meldingsautentiseringskode) inkluderer en hemmelig nøkkel *k* for beregning av hashfunksjon, som gir en autentisert hash-verdi *MAC* = Hash(*M, k*) .
- For å validere og autentisere en melding må mottakeren ha den samme hemmelige nøkkel *k* som ble brukt av avsender til å beregne *MAC*.
- En tredjepart som ikke kjenner nøkkelen kan ikke validere *MAC* -verdien.

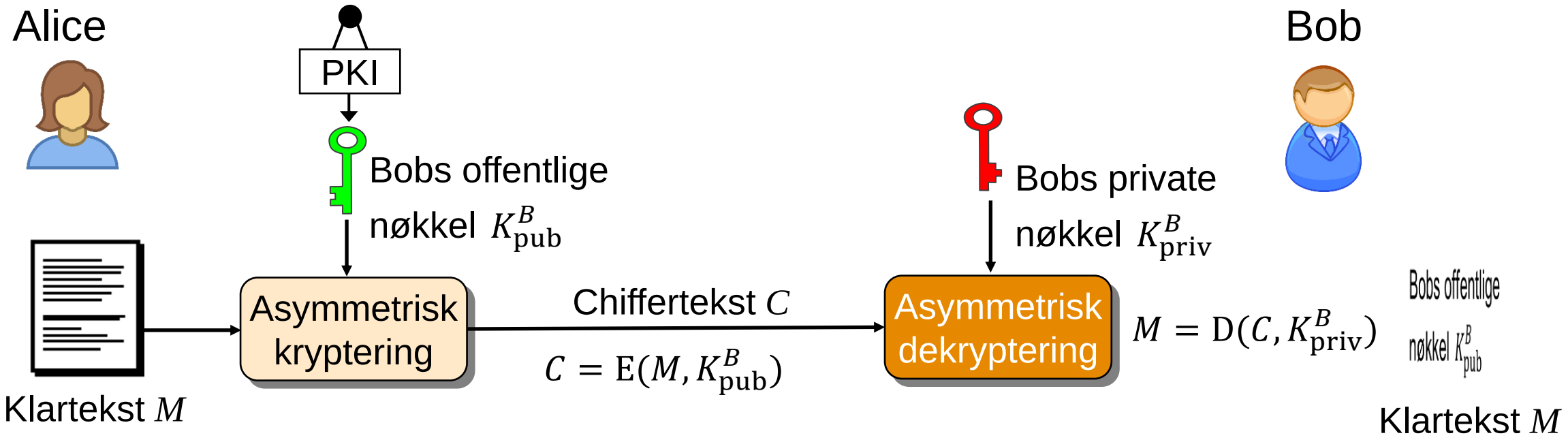
Meldingsautentisering med MAC



ASYMMETRISK KRYPTOGRAFI

- Asymmetrisk kryptering
- Diffie-Hellman nøkkelutveksling
- Digital signatur

Asymmetrisk kryptering – grunnleggende prinsipp



- Asymmetrisk kryptering og dekryptering krever tung beregning, og brukes ikke til direkte kryptering slik som vist over.
- I praksis brukes hybrid kryptering som kombinerer både en asymmetrisk og en symmetrisk algoritme.

Tradisjonelle asymemtriske krypteringsalgoritmer

- RSA: mest kjent asymmetrisk algoritme.
  - RSA = Rivest, Shamir og Adleman (utgitt 1977)
  - Historie: Den britiske kryptografen Clifford Cocks oppfant den samme algoritmen i 1973, men publiserte den ikke fordi det ble hemmeligstemplet.
  - Krever etterhvert store nøkler (typisk 2048 bits) for å opprettholde sikkerhet

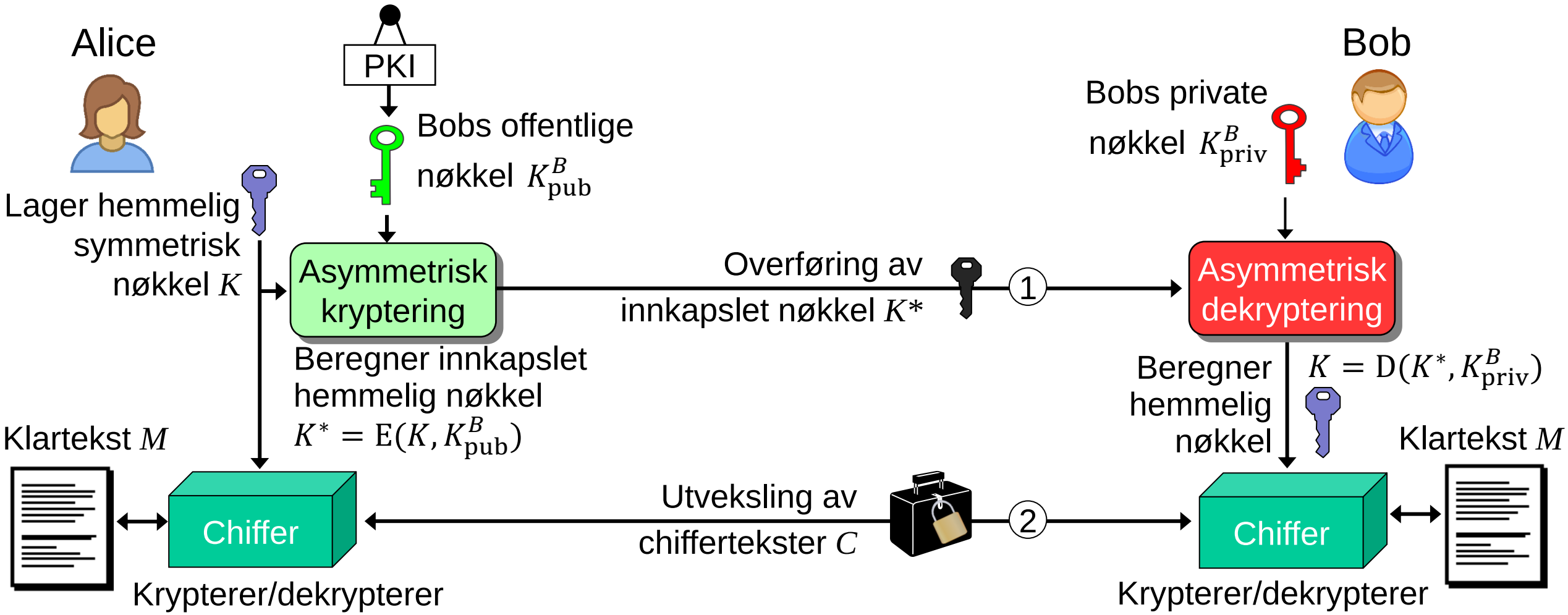
- Elliptisk kurvekryptografi
  - Basert på vanskeligheten med å løse EC diskrete logaritme.
  - Nøkler har mindre størrelse (typisk 256 bits) enn RSA.

Hybrid kryptering

- Symmetriske chiffer er mye raskere enn asymmetriske chiffer (fordi symmetriske chiffre benytter relativt enkle matematiske beregninger), men ...
- Asymmetriske chiffer støtter effektiv nøkkeldistribusjon, derfor ...
- Praktisk å benytte en kombinasjon av både symmetriske og asymmetriske chiffer - et hybridsystem:
  - Det asymmetriske chifferet brukes til å distribuere en hemmelig symmetrisk nøkkel.
  - Den hemmelige symmetriske nøkkelen med det symmetriske chifferet brukes for rask kryptering av datasett og meldinger.

- Svakheten ved hybrid kryptering er at det **ikke** gir perfekt fremoverhemmelighold (perfect forward secrecy).
  - Derimot kan perfekt fremoverhemmelighold oppnås ved å etablere øktnøkkel basert på Diffie-Hellman protokollen (som i TLS 1.3).

Hybrid kryptering (gir ikke fremoverhemmelighold)



Diffie-Hellman nøkkelutveksling

Alice velger privat delnøkkel a

Bob velger privat delnøkkel b

Alice beregner offentlig $g^a \pmod p$

Bob beregner offentlig $g^b \pmod p$

Alice sender til Bob → $g^a \pmod p$

$g^b \pmod p$ ← Bob sender til Alice

Alice beregner hemmelig $g^{ab} = (g^b)^a \pmod p$

Bob beregner hemmelig $g^{ab} = (g^a)^b \pmod p$

Alice og Bob har utvekslet anonym hemmelig nøkkel g^{ab}

Angripere kan ikke finne de hemmelige delnøklene *a* og *b* fordi beregning av diskret logaritme av store heltall er vanskelig. Dermed kan angripere ikke beregne den hemmelige nøkkelen = g^{ab} mod *p*.

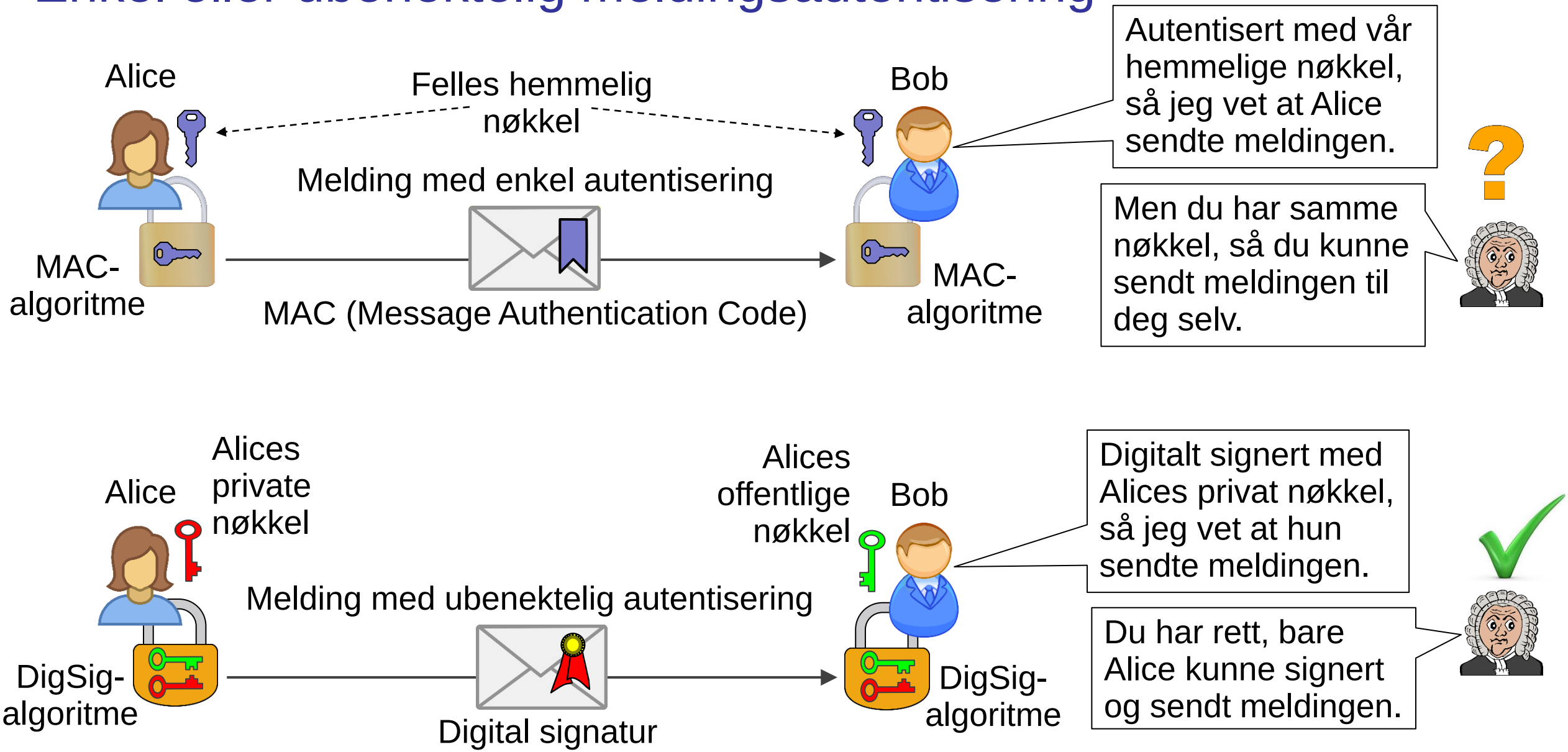
Diffie-Hellman nøkkelutveksling

- Problem:
  - Gir ingen autentisering
  - Alice og Bob kan ikke vite hvem de kommuniserer med
  - Mann-i-midten angrep mulig
- Løsning
  - Kombinasjon med digital signatur gir autentisert nøkkelutveksling
- Applikasjoner:
  - TLS (Transport Layer Security) og https
  - IKE (Internet Key Exchange) og IPSec (IP Security)

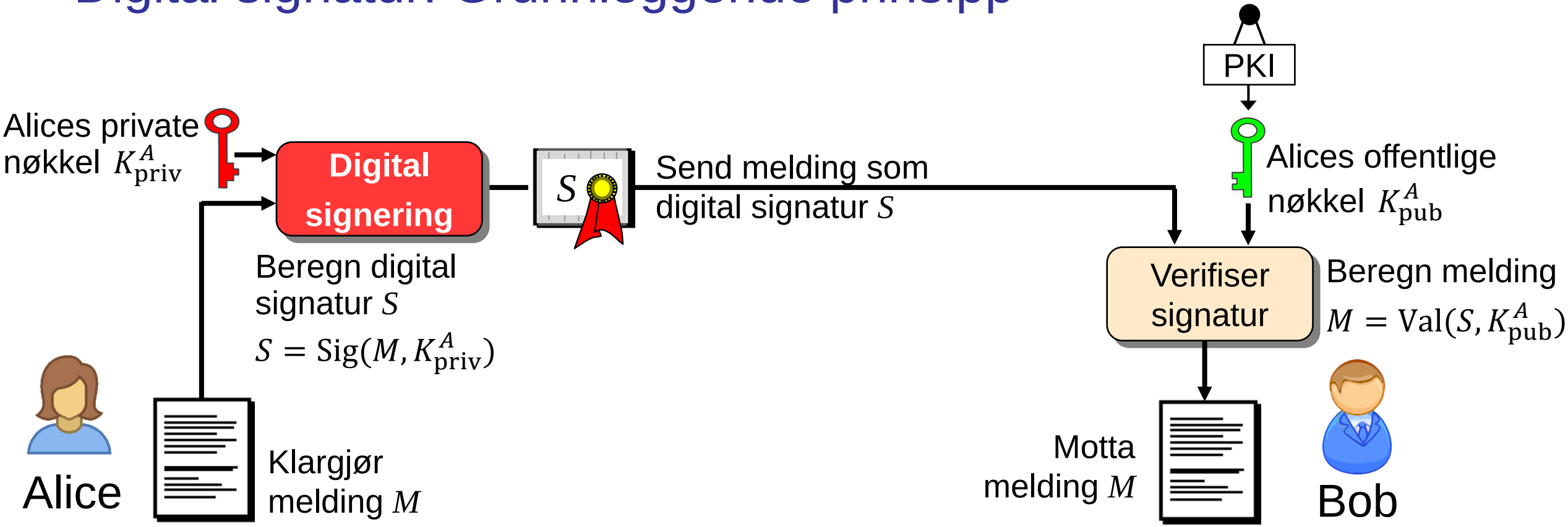
Behov for digital signatur

- En MAC kan ikke brukes som bevis for data-autentisitet som skal verifiseres av en tredjepart
- Digitale signaturer kan valideres av tredjepart
  - Gir ubenektelig (sterk) data-autentisering,
- Funksjoner for digitale signatur:
  - Signering (bruker privat nøkkel)
  - Verifisering (bruker offentlig nøkkel)

Enkel eller ubenektelig meldingsautentisering

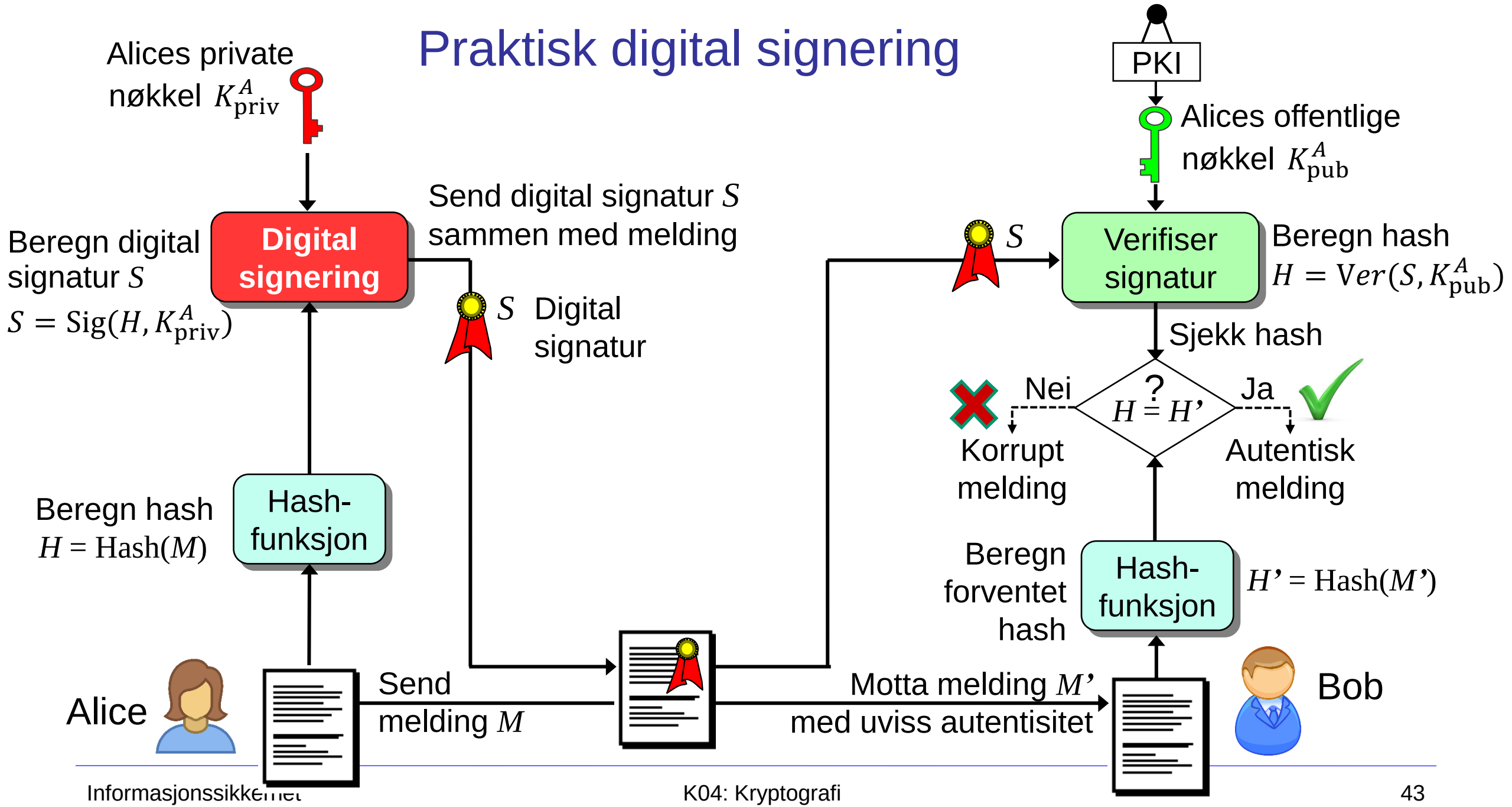


Digital signatur: Grunnleggende prinsipp

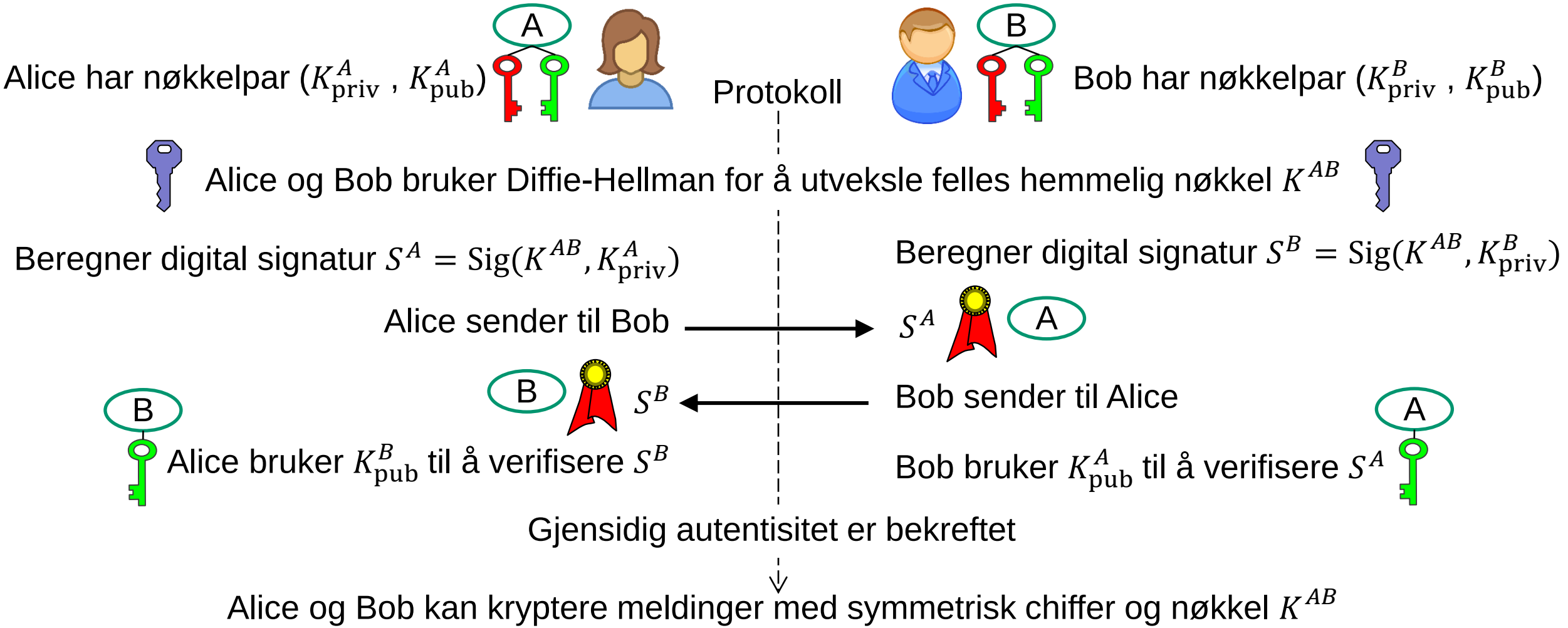


- Digital signering og validering krever tung beregning, og brukes ikke til direkte signering slik som vist over.
- I praksis brukes hybrid signering som kombinerer en hashfunksjon og digital signering.

Praktisk digital signering



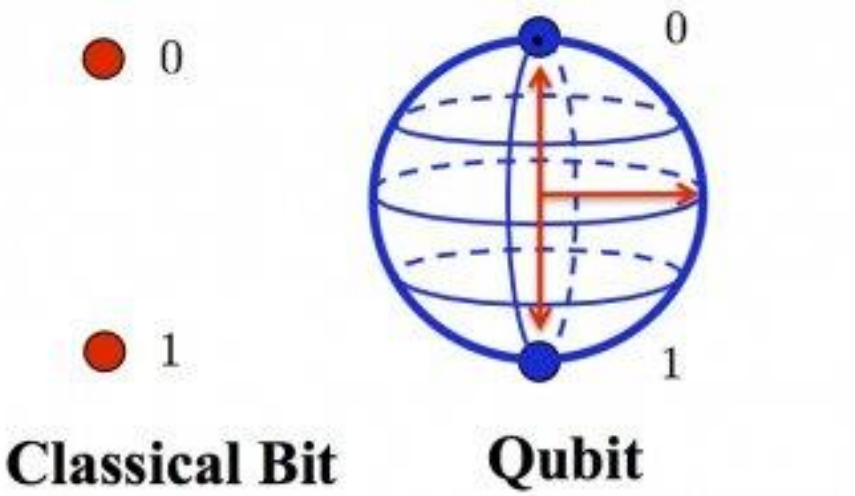
Autentisert nøkkelutveksling for fremoverhemmelighold



POSTKVANTEKRYPTOGRAFI

Kvantecomputere

- Kvanteberegning (Quantum Computing - QC) er basert på kvante «qubits» istedetfor binære bit



Eksperimentell kvantecomputer

- Kvantealgoritmer, dvs. algoritmer for kvantecomputere, kan potensielt knekke vanlige asymmetriske krypto-algoritmer, f.eks. RSA, DSA and Diffie-Hellmann.

Kryptografiske funksjoner

Symmetrisk kryptering

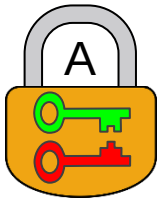
Konfidensialitet

Autentisitet / Integritet

Hash-funksjoner

Digital Signatur

PKI / Nøkkeldistribusjon



Asymmetrisk kryptering & Digital Signatur (Tradisjonell), f.eks. RSA, DSA, Diffie Hellman

Konfidensialitet

Kvantetrusselen

Kryptografiske funksjoner

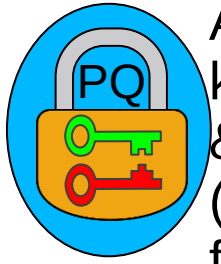
Symmetrisk kryptering

→ Konfidensialitet

→ Autentisitet / Integritet

Hash-funksjoner

→ Digital Signatur
PKI / nøkkeldistribusjon

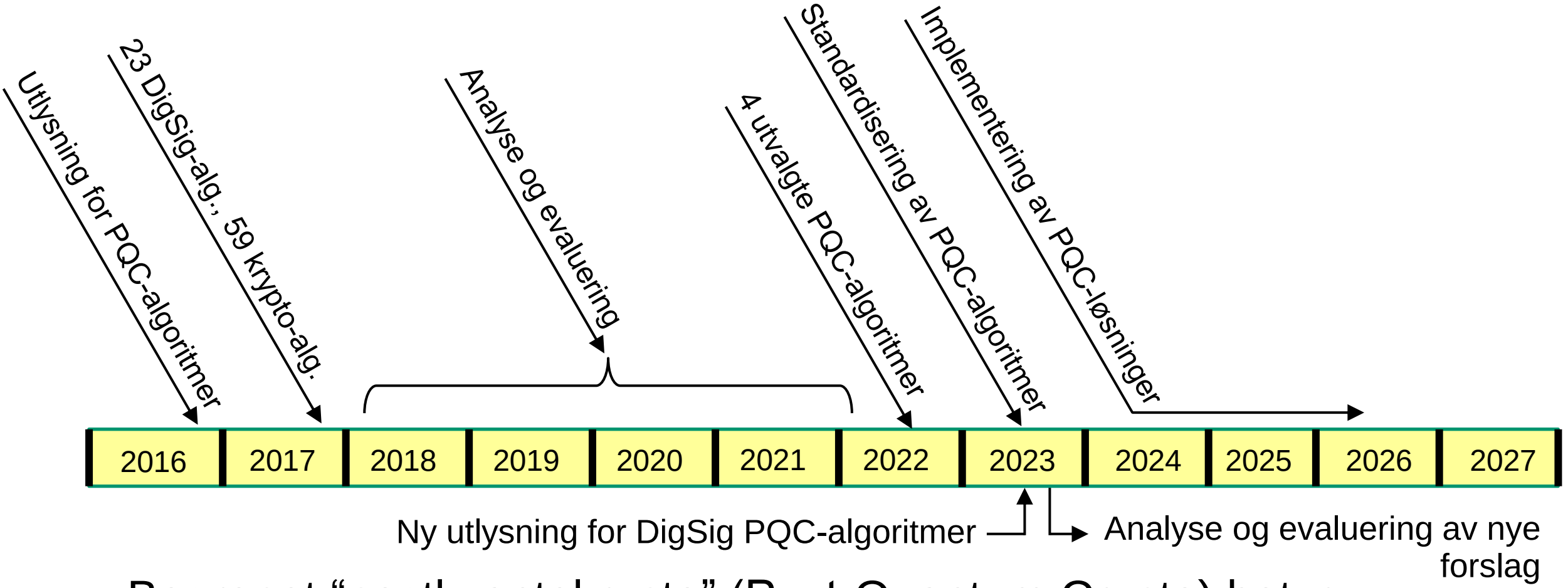


Asymmetrisk kryptering & Digital Signatur (Post-Quantum), f.eks. Lattice-based, Multivariate, Hash-based, Code-based, Elliptic curve isogeny

→ Konfidensialitet

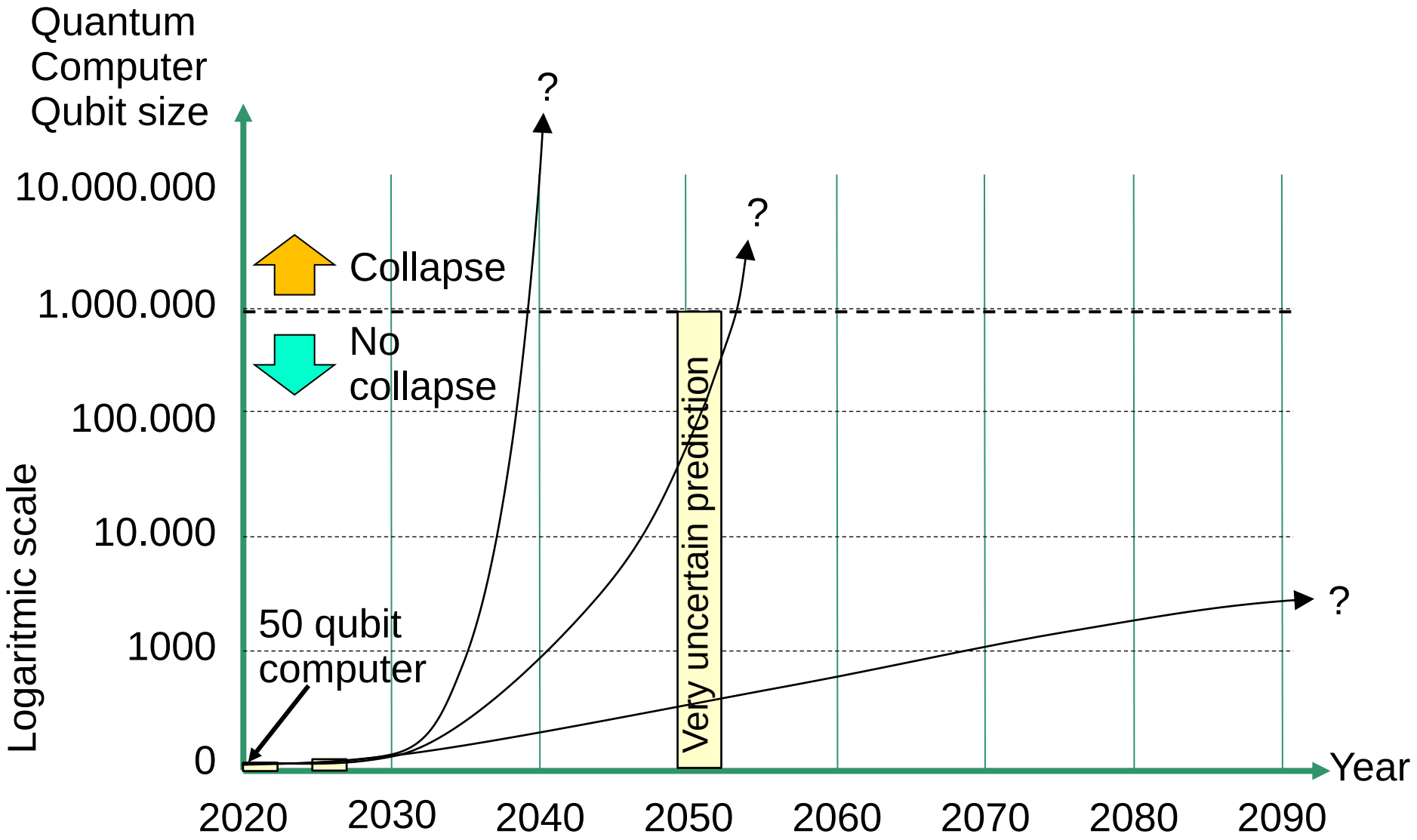
Med postkvantekrypto kan vi beholde DigSig og PKI selv med kvantecomputere som har 1 million qubit

Standardisering av postkvantekrypto



- Begrepet “postkvantekrypto” (Post-Quantum Crypto) betyr kryptografi som ikke kan knekkes av kvantecomputere, også kalt kvanteresistente algoritmer.

Sammenbrudd av trad. asymmetrisk crypto?



Hvorfor gå over til postkvantekryptografi?

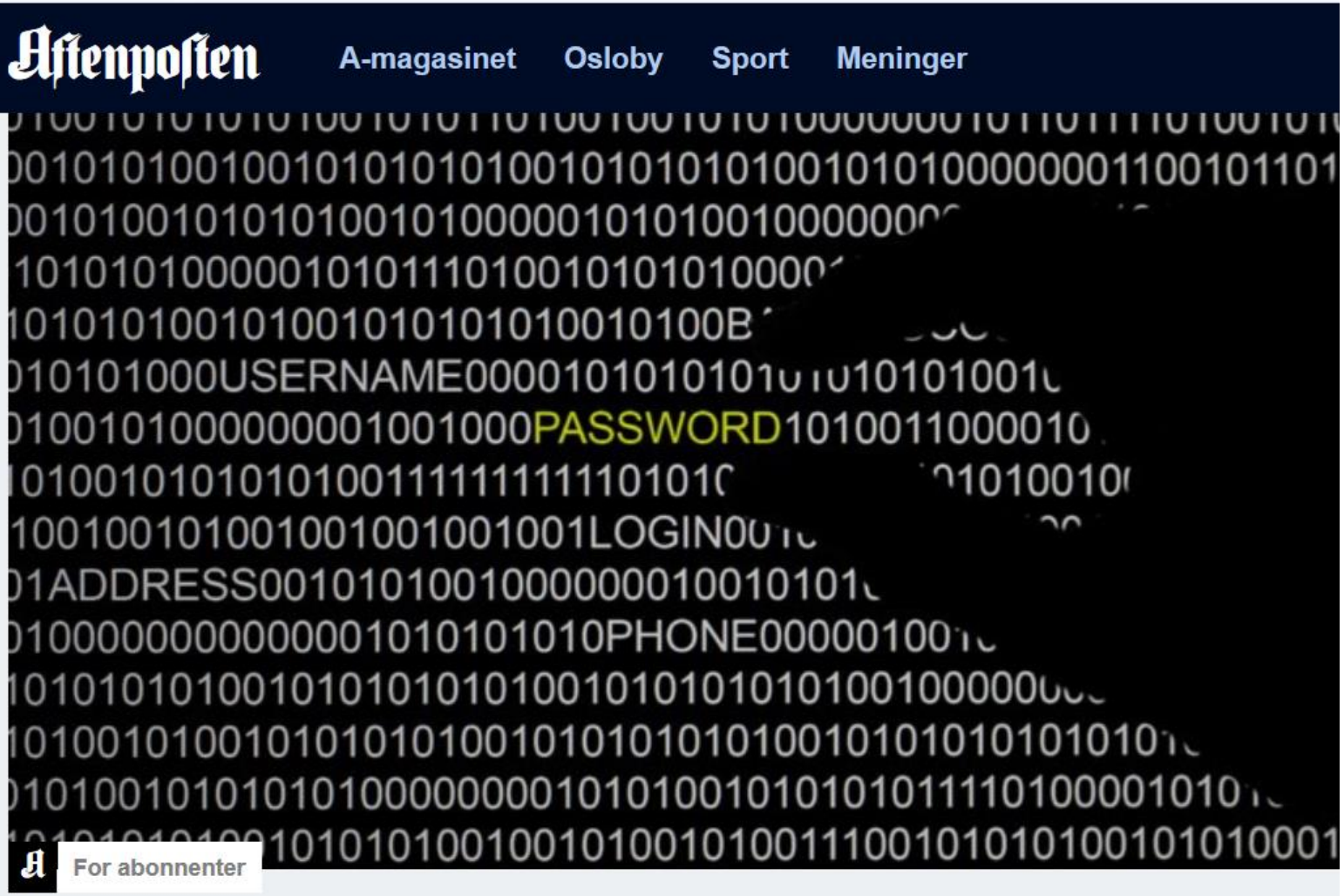
A. Bruk postkvantekrypto fordi kvantecomputere sannsynligvis vil knekke RSA, Diffi-Hellmann og DSA en gang i fremtiden.
B. Bruk postkvantekrypto fordi du ikke vil at organisasjonen din skal havne på forsiden i avisen, beskyldt for å være uansvarlig.

DN.no, 1. desember 2017

Aftenposten.no, 10. mai 2018

https://www.aftenposten.no

Aftenposten A-magasinet Osloby Sport Meninger



For abonnenter

Norges hemmeligheter skjules bak sterke krypteringer. Om få år vil kodene kanskje kunne knekkes.

SLUTT PÅ PRESENTASJONEN