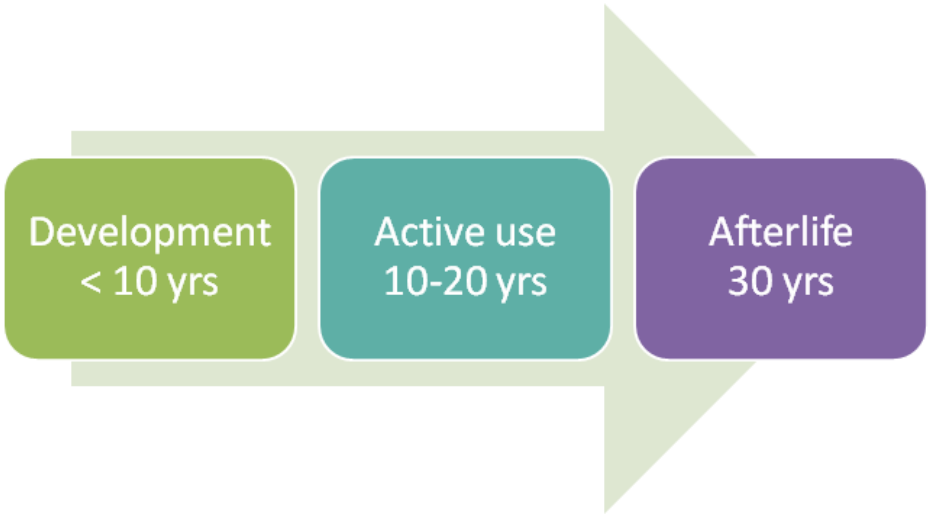
Kvanteresistent kryptografi

Thomas Gregersen

Dagens meny

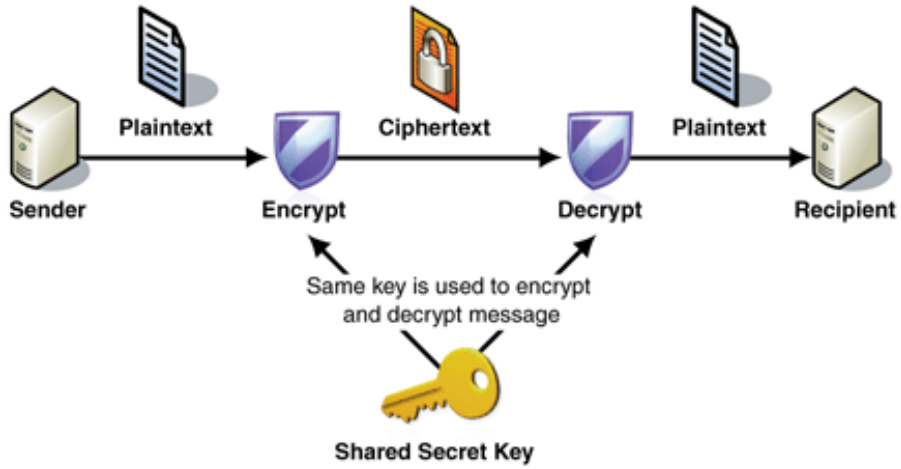
- Kvantedatamaskiner kan bli et problem for fundamentet bak dagens protokoller.
- Selv om vi ikke kjenner de nøyaktige forholdene må vi forberede oss.



  - For mange parter må informasjon beskyttes lenge.
  - Introduksjonen av nye algoritmer tar vanligvis lang tid.
  - Med et langsiktig perspektiv må vi altså starte tidlig for å kunne klare omstillingen i tide.

Dagens meny

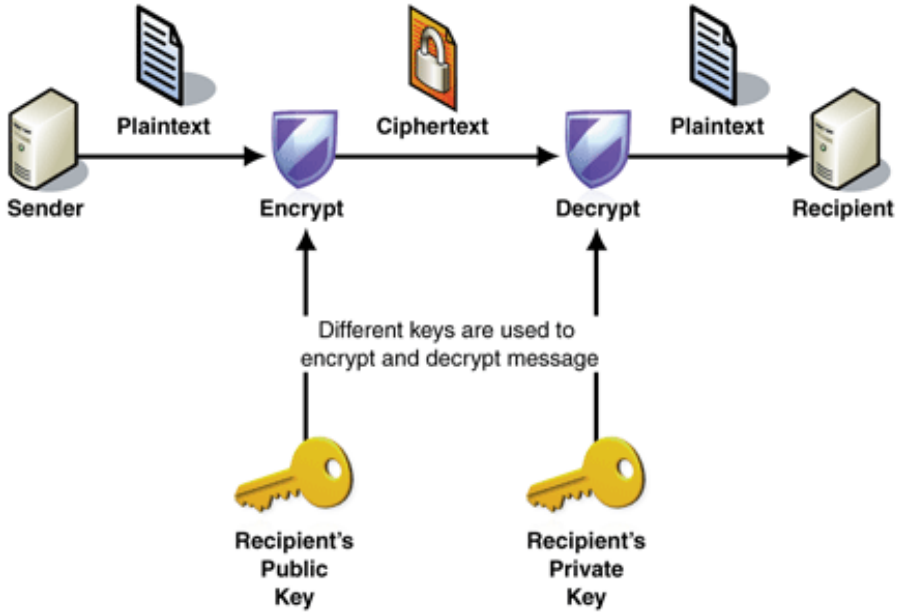
- Utgangspunktet for å etablere konfidensialitet:



- Dette krever en felles nøkkel som må etableres på forhånd.

Dagens meny

- For å slippe å møtes, kan de to partene etablere en felles nøkkel ved hjelp av asymmetriske algoritmer:



- Det er vanlig å kombinere dette i en hybridisert løsning: Den symmetriske algoritmen tar seg av bulktransport siden den er mye raskere.

- Det finnes flere varianter vi kan benytte til nøkkeletablering:
  - RSA
  - DH
  - ECDH
- I hver av dem antar vi at vi kan redusere det å finne nøkler eller klartekst til et beregningstungt problem (faktorisering, finne logaritmer).
- Kvantealgoritmer angriper disse underliggende problemene og utgjør en alvorlig trussel.

Dagens meny

- Det finnes igjen flere varianter vi kan benytte:
  - RSA
  - DSA
  - ECDSA
- De underliggende problemene blir igjen angrepet av kvantealgoritmer.

Dagens meny

- Algoritmen finner i elementer som avbilder til et oppgitt element gjennom en funksjon f.
- Kryptografisk relevant fordi vi f.eks. kan la f være et chiffer eller en kryptografisk hash-funksjon hvor vi ønsker å finne nøkler eller kollisjoner.
- Kompleksiteten for denne beregningen er en forbedring sammenlignet med klassiske varianter:

$$\mathcal{O}(\sqrt{N}) \text{ versus } \mathcal{O}(N), (N = |dom(f)|).$$

Dagens meny

- Shors algoritme tar et naturlig tall N som input og finner en ikke-triviell divisor a.
- I RSA kan vi dermed faktorisere den offentlige modulusen vi bygger på og finne den private nøkkelen.
- Algoritmens kjerne finner perioden til en funksjon, og dette bruker Fourier-transformen som kan implementeres effektivt i en kvantekrets. Oppsettet kan modifiseres til å finne logaritmer i enkelte typer grupper og dermed utfordre DH/ECDH.
- Med andre ord kan en ikke forlate én enkelt algoritme for å unngå dette angrepet.

Dagens meny

Introduksjon

Kryptografi i dag

Symmetrisk kryptografi

Asymmetrisk nøkkeletablering

Digitale signaturer

Kvantetrusselen

Grovers algoritme

Shors algoritme

Kryptografiske konsekvenser

Veien mot kvanteresistent kryptografi

Kvanteresistent kryptografi

Kodebaserte kryptosystemer

Lattice-baserte kryptosystemer

Hash-baserte signaturer

Kvanteberegninger kommer, eller?

Oppsummering

- Tar vi et pessimistisk utgangspunkt blir vi nødt til å doble antall bits i symmetriske nøkler, men dette er mest sannsynlig ikke det endelige estimatet.
- For de asymmetriske algoritmene vi vanligvis bruker ser det verre ut: Det kan bli nødvendig å utvide til størrelser det er helt urealistisk å implementere.
- For å finne kandidater vi kan bytte til er det satt opp flere løp som organiserer veien fremover:
  - National Institute of Standards and Technology (NIST) er allerede i gang med runde 2 for å komme til mulige erstattere[1].
  - PQCRYPTO (EU) er et annet inititiativ som skal levere kandidater og praksis[2].

[1] https://csrc.nist.gov/Projects/Post-Quantum-Cryptography
[2] https://pqcrypto.eu.org/

Dagens meny

- Hva ønsker vi oss av nye algoritmer?
  - Nøkler/signaturer/chiffertekst som ikke tar for mye plass.
  - Kryptering/dekryptering/signering/autentisering som ikke tar for lang tid.
  - Sikkerhet basert på reduksjon til beregningsproblemer som vi vet er vanskelige.
- Det er på ingen måte lett å kombinere alt dette samtidig, men det finnes kandidater.

- NIST startet med 69 algoritmer for nøkkeletablering og signaturer som nå har blitt til 17 og 9.
- Til nøkkeletablering er kandidatene basert på kodebasert, lattice-basert eller isogeni-basert kryptografi:
  - BIKE/Classic McEliece/HQC/LedaCrypt/NTS-KEM/ ROLLO/RQC.
  - CRYSTALS-KYBER/FrodoKEM/LAC/NewHope/NTRU/ NTRU Prime/Round5/SABER/Three Bears.
  - SIKE.
- Signaturalgoritmene er basert på lattice, multivariate polynomsystemer, Zero Knowledge Proof-system og hash-baserte signaturer:
  - CRYSTALS-DILITHIUM/FALCON/qTesla.
  - GeMSS/LUOV/MQDSS/Rainbow.
  - Picnic.
  - SPHINCS+.

- McEliece/Niederreiter-system (1978/1986) basert på feilkorrigerende koder.
- En feilkorrigerende kode $\mathcal{C}$ er en metode for å legge til redundans til informasjon slik at feil kan rettes etter sending.
- Tilhørende finnes en dekodingsalgoritme $\mathcal{D}_{\mathcal{C}}$ som retter de ev. feil som har oppstått.

- En *lineær* feilkorrigerende kode gjør dette ved å behandle informasjon som vektorer i et omkringliggende vektorrom.
- Dermed kan $\mathcal{C}$ en $[n, k]$-kode spesifiseres ved:
  - Rekkerommet til en generatormatrise $G \in F_2^{k \times n}$

$$\mathcal{C} = \{mG | m \in F_2^k\}$$

  - Nullrommet til en paritetsjekkmatrise $H \in F_2^{(n-k) \times n}$

$$\mathcal{C} = \{c | Hc^\intercal = 0, c \in F_2^n\}$$

- Systemparametre: $n, t \in \mathbb{N}, t \ll n$.
- Nøkkelgenerering:
  - $G : k \times n$ generatormatrise for en lineær kode (binær, irredusibel Goppa) $\mathcal{C}$ som kan korrigere opp til t feil.
  - $S : k \times k$ tilfeldig binær invertibel matrise.
  - $P : n \times n$ tilfeldig permutasjonsmatrise.

  Beregn så $G' = SGP$.
- Public Key: (G', t).
- Private Key: $(S, P, \mathcal{D}_{\mathcal{C}})$.

- Kryptering: $m \in F_2^k$ sendes til

$$c = mG' + e$$

hvor $e \in F_2^n, wt(e) = t$.

- Dekryptering:
  - Beregn

$$cP^{-1} = (mS)G' + eP^{-1}$$

og får kodeordet

$$mS = \mathcal{D}_{\mathcal{C}}(cP^{-1}).$$

  - Beregn til slutt

$$m = mSS^{-1}.$$

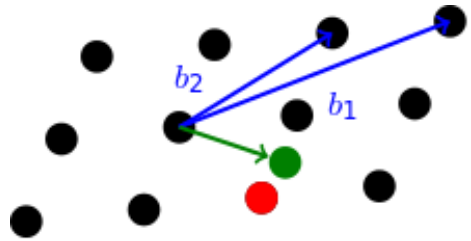
- Å dekode en generell lineær kode er et NP-hardt problem. Derfor er den private nøkkelen forkledd ved matrisene S og P, og den er vanskelig å skille fra en generell lineær kode.
- De raskeste angrepsalgoritmene viser seg å være informasjonssettdekoding(ISD)-angrep som dekoder meldinger fra en generell kode, men for binære Goppa-koder er dette fortsatt langt fra effektivt.

- Skolebokvariantene av McEliece/Niederreiter blir for tynne i seg selv: Det er muligheter for å bruke strukturen i kryptosystemet for å gjøre analysen lettere:
  - Delvis kjent klartekst:
    Dette medfører at vi kan redusere raskeste dekodingsangrep (ISD-varianter) til de komplementære bitene.
  - Kjente relasjoner mellom meldinger:
    Disse forplanter seg til chifferteksten og kan brukes til å redusere antall feilvektorer vi trenger å teste i ISD-analysen.
- Dette løser man ved å bruke en CCA2-sikret variant som medfører overhead: Ekstra bits for en valgt sikkerhetstoleranse som avhenger av input til hashfunksjonen som involveres.

- Matrisene vi trenger er store og kan gi problemer når det er lite plass.
- Vi kan velge mellom flere mulige koder, men mange viser seg for svake til kryptografisk anvendelse.

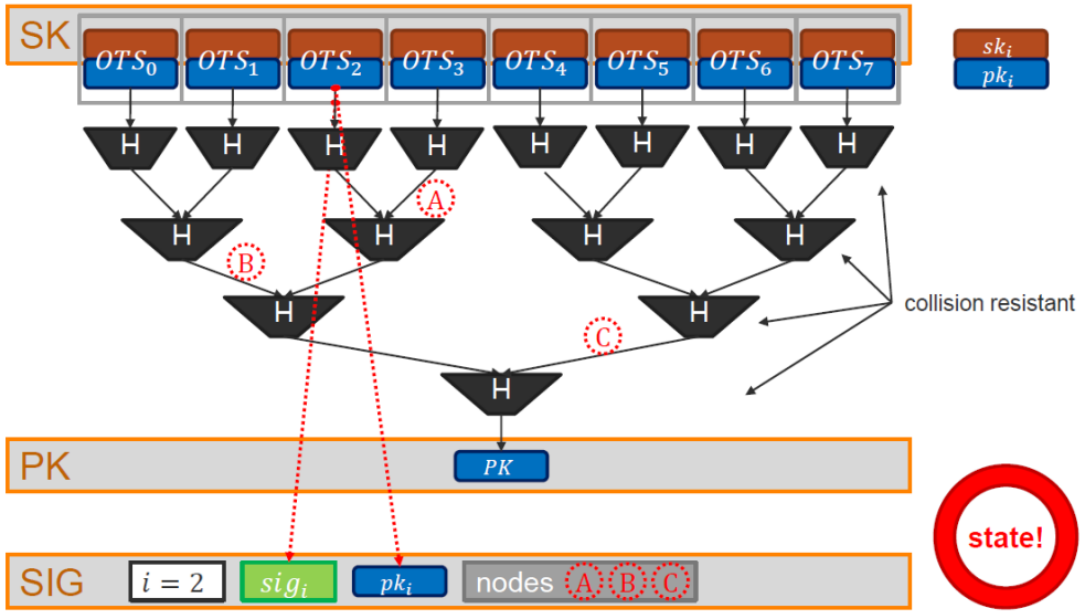
  De klassiske binære Goppa-kodene holder enda, men kanskje er det flere (LDPC/MDPC/Rank-Metric codes).

- Kan baseres på flere strukturer som hviler på lattice-teori: LWE, RLWE, MLWE, NTRU.
- Raske fakta:
  - Kryptering/dekryptering innebærer å kode informasjon som heltallskombinasjoner av vektorer (et lattice).



  - Sikkerhet ved reduksjon til geometriske problem i lattice som er velkjente, men ikke så godt studert som McEliece/Niederreiter.
  - Mer effektive enn McEliece/Niederreiter og det er foreslått varianter hvor vektorene tar form av polynomer som kompaktifiserer. Dette kan likevel gå utover sikkerheten hvis vi innfører for mye struktur.

- Merkle signaturer[3]:



- Raske fakta:
  - Tar engangssignaturer som utgangspunkt (Lamport, Winternitz,..).
  - Kan i utgangspunktet signere et endelig antall ganger for en gitt offentlig nøkkel, noe som kan løses med dynamiske trær.
  - Store signaturer hvis treet er stort.
  - Sikkerhet hviler på styrken til den interne hash-funksjonen H.

[3] https://postscryptum.lip6.fr/slides_aline.pdf

- Noen observasjoner:
  - Nøkler/signaturer/chiffertekst er i noen tilfeller veldig store (tall i bytes for Classic McEliece[4] og SPHINCS+[5]):

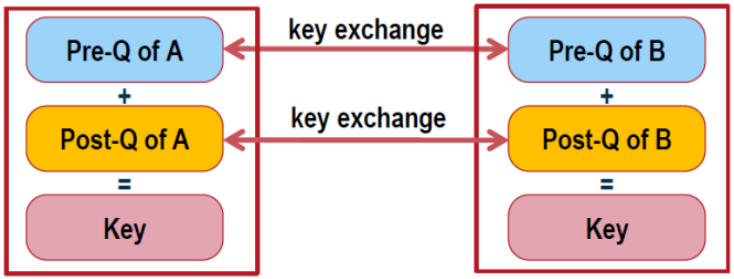
	Public key	Private key	Ciphertext	Session key
mceliece348864	261120	6452	128	32
mceliece460896	524160	13568	188	32
mceliece6688128	1044992	13892	240	32
mceliece6960119	1047319	13908	226	32
mceliece8192128	1357824	14080	240	32

	public key size	secret key size	signature size
SPHINCS$^+$-128s	32	64	8 080
SPHINCS$^+$-128f	32	64	16 976
SPHINCS$^+$-192s	48	96	17 064
SPHINCS$^+$-192f	48	96	35 664
SPHINCS$^+$-256s	64	128	29 792
SPHINCS$^+$-256f	64	128	49 216

  - Regnetid man som regel kan leve med.
  - Sikkerhet er i noen tilfeller godt fundert, andre ganger er det mindre kryptoanalyse å hvile på.

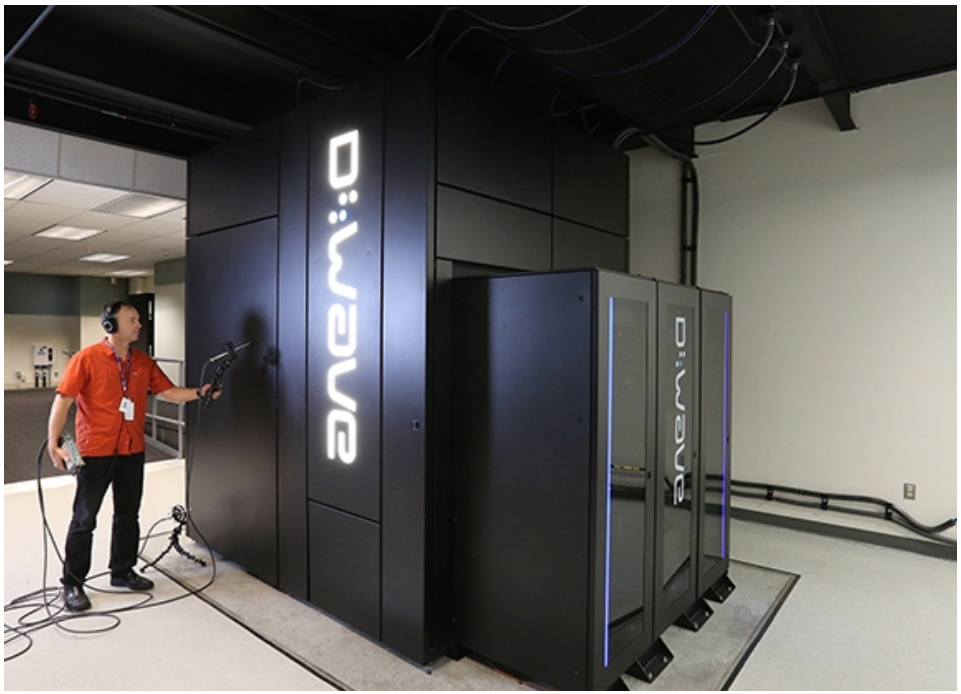
[4] https://classic.mceliece.org/nist/mceliece-20190331.pdf
[5] https://sphincs.org/data/sphincs+-round2-specification.pdf

- I denne tidlige fasen er vi usikre på hvor inngripende kvantealgoritmer faktisk blir. Det er foreslått å bruke hybridløsninger:



- Protokollene vi bruker kan i så fall få større kompleksitet og vi blir nødt til å analysere hvordan dette påvirker sikkerhet og effektivitet.
- Det kan bli standardisert mange muligheter avhengig av brukercase, altså ikke én variant til alle formål.

- Masser av forskning peker mot det å realisere kvantekretser, men på hvilket nivå?
- Et lite modent eksempel[6]:



[6] https://www.dwavesys.com

- Med så mange veier til mål for å realisere dem, virker det dumt å satse på at ingen forsøk på å realisere dem vil lykkes.
- Det gjenstår en god del arbeid før vi har en stor og stabil nok kvantekrets hvor en vilkårlig kvantealgoritme kan kjøres (logiske versus fysiske qubits).
- Det er satt av store ressurser til forskning og utvikling, og mange aktører vil bidra (Google, IBM, Lockheed Martin..).

- Det er ikke lett å avgjøre *når* vi må være klare for å håndtere en kvantedatamaskin/kvantekrets.
- I denne fasen er det tatt utgangspunkt i at dette kan være realistisk rundt 2030.
- For oss blir det viktigste å kunne håndtere nye primitiver/algoritmer, altså ha plass nok til å kunne integrere dem.

Oppsummering

- Kvantealgoritmer tvinger oss til å finne nye byggesteiner i kryptografien.
- Vi vet fortsatt ikke når vi må ha dem på plass, men så snart som mulig.
- For oss som jobber med planlegging er det en god idé å følge standardprosessen tett og sørge for at det blir plass til de nye primitivene der hvor vi må bruke dem.