Introduksjon til kryptering

Ismail Hassan

2019

Cryptography (Kryptografi)

Cryptography er et gresk ord som betyr skjult skriving:

- Crypto = Sjult
- Graphy = Skriving

Cryptology (Kryptologi)

Fagfeltet kryptologi kan deles i 2:

- **Kryptografi:**
  - Kunsten å konstruere matematiske kryptoalgoritmer
- ~~**Kryptoanalyse:**~~
  - Kunsten å knekke/bryte kryptoalgoritmer

Hva er kryptering (Encryption)?

" kryptering (Encryption): er en matematisk prosess hvor informasjonen blir kodet på en slik måte at kun den som kjenner koden kan lese informasjonen."

Begreper

- Klartekst (Plaintext), **P**
  - teksten som er leselig for alle
- Kryptert tekst (Ciphertext), **C**
  - teksten som har blitt kryptert
    - Teksten er uleselig for alle bortsett fra den som har nøkkelen som kan dekryptere den
- Krypteringsnøkkel (Encryption Key), **K**
  - Nøkkelen som benyttes for å krpytere teksten
    - Samme krypteringsnøkkel(**K**) kan benyttes for kryptering og dekryptering (Dette kales **Symmetrisk kryptering**)
    - Forskjelig krypteringsnøkkel(**K**) kan benyttes for kryptering og dekrytering (Dette kalles **Asymmetrisk kryptering**)
- Krypteringsfunksjonen, **E**
  - Funksjonen som **P** (Plaintext) sendes gjennom for å få **C** (Ciphertext). $E_K(\mathbf{P}) = \mathbf{C}$.
- Dekrypteringsfunksjonen, **D**
  - Funksjonen som **C** (Ciphertext) sendes gjennom for å få **P** (Plaintext), $D_K(\mathbf{C}) = \mathbf{P}$

Kryptering

- Kryptert tekst (Cipher) og algoritmer (algorithms) er to typer operasjoner som brukes i kryptografi
- Hvis vi har en ukryptert verdi **P** (**plaintext**) og kjører det gjennom en kryteringsalgoritme **Encryption**, da får vi en uleslig tekst **C** (**ciphertext**)

- Problemet med dette er sikkerheten i den uleselige teksten (Ciphertext) er avhengig av hemligheten/oppskriften til algoritmen. Dersom noen får kjennskap til oppskriften eller gjør "reverse engineering" på den, vil det føre til at alle meldinger lett kan dekrypteres

Assume your secrets are not safe:

Det er ekstremt farlig å stole på hemmeligholdelse(Security through obscurity) av oppskriften/algoritmen!

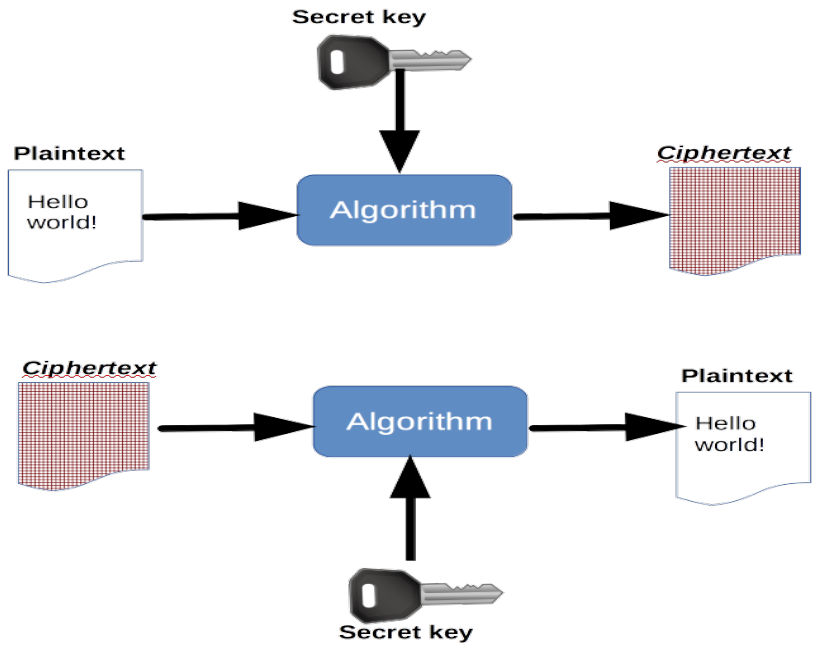
> *" Kerckhoffs's principle:*
> *A cryptographic system should be secure even if everything about the system, except the key, is public knowledge."*

Figur: Auguste Kerckhoffs

Source: Kerckhoffs's principle

Kryptosystem

- En mer alminnelig løsning er der oppskriften/algoritmen er godt kjent, men at den tar 2 input
  - En input er klarteksten(plaintext), og den andre inputen en hemlighet som er kunn kjent av de som er involvert i kommunikasjonen
  - Denne hemligheten kalles en nøkkel(key), i dette tilfellet den hemmlige nøkkelen



- Vi trenger ikke å hemmligholde oppskriften, vi må derimot holde nøkkelen strengt hemmelig.

Hvorfor bruke kryptering?

Services of Cryptosystems

Kryptosystemer kan tilby følgende tjenester:

- **Konfidensialitet (Confidentiality):**
  - Hindre at uvedkommende får tilgang til informasjon de ikke burde ha tilgang til. Gjør informasjonen uforståelig/uleselig bortsett fra autorisert enheter.
- **Integritet (Integrity):**
  - Sørge for at data ikke er endret på en uautorisert måte siden den ble opprettet, overført eller lagret. Bare de som har lov til å endre informasjonen, får endret den.
- **Autentisitet (Authenticity):**
  - Sikre opphavet til informasjonen. At vi kan bekrefte hvem som sendte informasjonen
- **Ikke-fornekting (Accountability/Non-repudiation):**
  - At en digital handling ikke skal kunne benektes i etterkant.

Klassisk kryptografi (Classical Cryptography)

Krypteringsteknikker

De to grunnleggende byggesteinene i alle krypteringsteknikker er substitution og transposition.

- **Substitution(utbytting):**
- **Transposition (omstokking):**

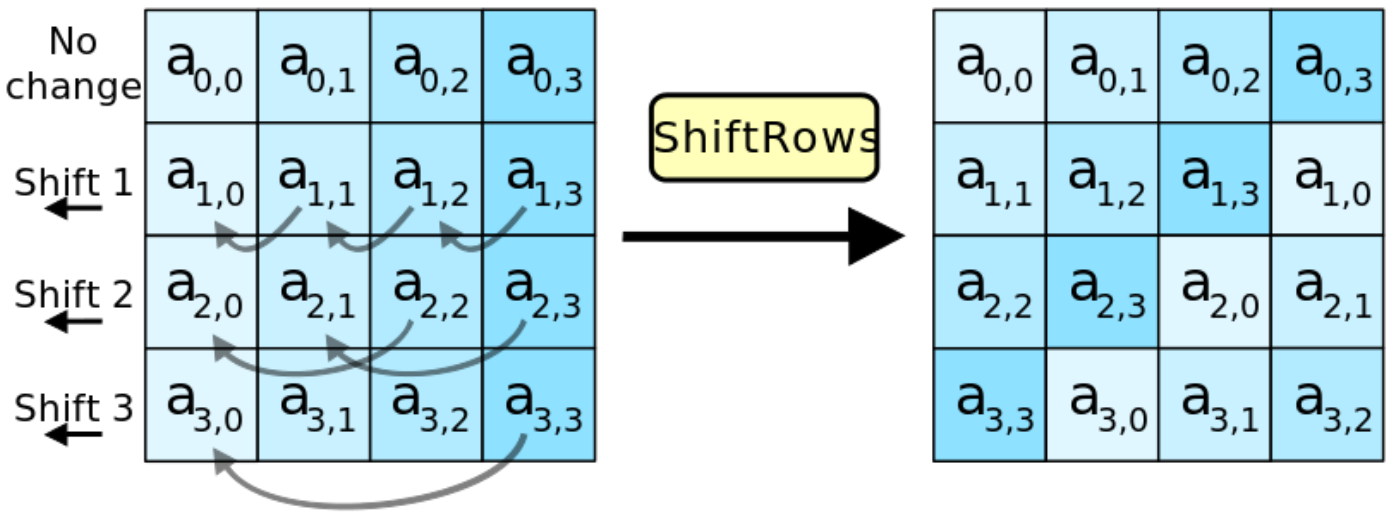
Substitution(utbytting):

- Substitusjonsmetoden erstatter hver bokstav/tegn eller blokker med forskjellige bokstav/tegn. Med andre ord en bokstav mappes direkte til en annen bokstav
  - Et eksampel på dette er Caesar Cipher som er en Monoalfabetisk substitusjon(Monoalphabetic substitution) der hver bokstav er erstattet med en annen bokstav i en fast avstand i alfabetet.

Mer om Caesar Cipher i de neste slidene

Transposition (omstokking):

- Transponeringsmetoden erstatter ikke den opprinnelige teksten med annen tekst, men heller flytter/omrokkerer de opprinnelige bokstavene/tegnene rundt for å skjule den opprinnelige betydningen.
  - Advanced Encryption Standard (AES) inneholder et transponeringstrinn som flytter/omstokker bytes i et 16-byte-array.

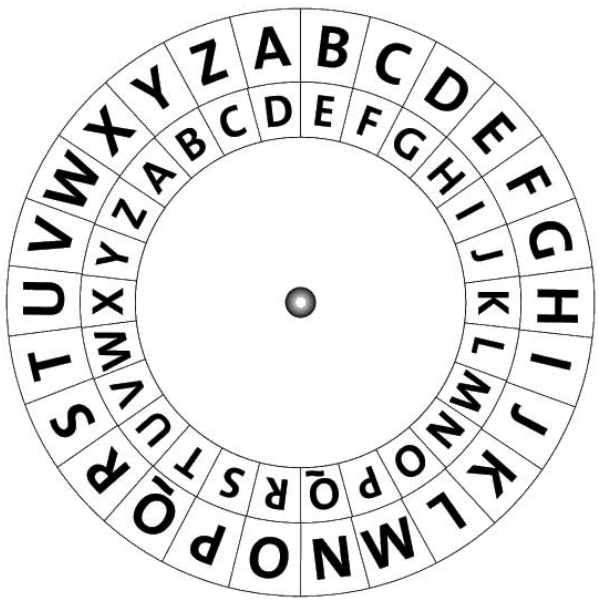


Mer om Advanced Encryption Standard (AES) i de neste slidene

Caesar cipher!

Klassisk kryptografi (Caesar cipher)

- Kryptering har vært benyttet i mer enn tusen år med stadig mer avanserte matematiske beregningsmetoder
  - Den tidligste kjente bruk av substitution cipher var Caesar cipher.
  - **Caesar cipher** innebærer å bytte hver bokstav i alfabetet med en bokstav som står 3 plasser lenger ned på alfabetet:



Monoalfabetisk substitusjon (Caesar cipher)

Direkte to-veis mapping mellom bokstaver

- Eksempel: Caesar Cipher (bokstavskift)
  - **C = E(p) = p + 3**
    - A —> D, B —> E, C —> F, D —>G,....., X —> A, Y —> B, Z —> C
  - **C = E(p) = p + 7**
    - A —> H, B —> I, C —> J, D —> K, ..., T—> A, U —> B, V —> C
- Svakhet:
  - Enkel å knekke gjennom Brute-force angrep
  - Enkel å knekke basert på frekvensanalyse(bokstavfordeling)

Example Caesar cipher with K=3

K=0	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
K=3	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Plaintext	D	A	T	A	S	I	K	K	E	R	H	E	T
Ciphertext	G	D	W	D	V	L	N	N	H	U	K	H	W

Example Caesar cipher with K=7

K=0	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
K=7	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G

Plaintext	D	A	T	A	S	I	K	K	E	R	H	E	T
Ciphertext	K	H	A	H	Z	P	R	R	L	Y	O	L	A

Demo av Caeser Cipher Brute Force Attack

Moderne kryptografi (Modern Cryptography)

Moderne kryptografi

- Moderne kryptografi benytter avanserte matematiske funksjoner for å forvrenge innhold i dokumenter og meldinger
- Det finnes to hovedkategorier av moderne kryptografi
  - Symmetrisk kryptografi (Symmetric Cryptography)
    - Klassisk kryptografi var basert utelukkende på symmetrisk kryptografi
  - Asymmetrisk kryptografi (Asymmetric Cryptography)

Symmetrisk kryptografi

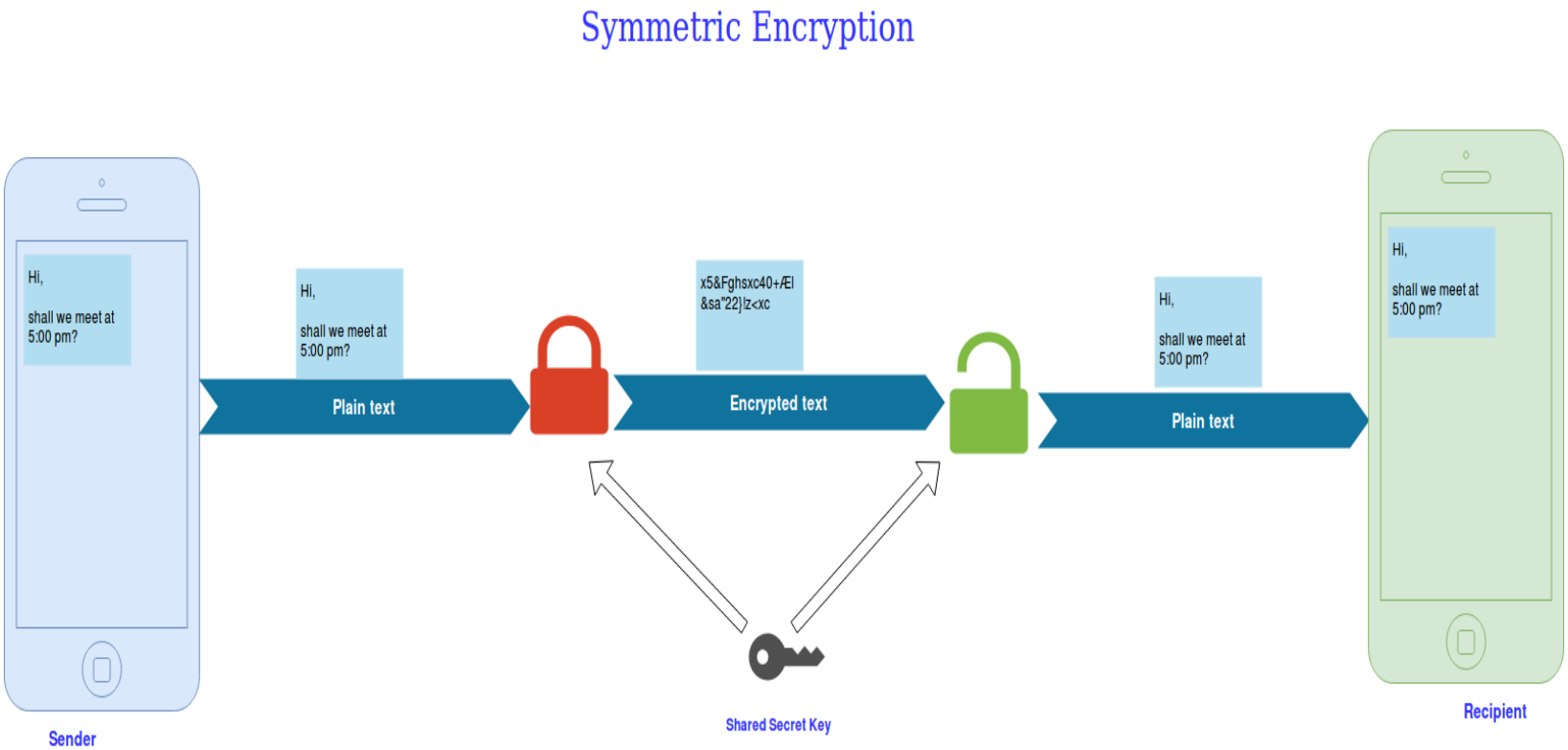
What is Symmetric Cryptography?

> *"is a branch of cryptography in which the algorithms use the same key for both of two counterpart cryptographic operations (e.g.,encryption and decryption)."*

Source: RFC 4949 - Internet Security Glossary, Version 2

Symmetrisk kryptografi

- En hemmelig nøkkel benyttes til kryptering og dekryptering
  - Problem: Utveksling av nøkler



Symmetrisk krypteringsalgoritmer

DES, 3DES, and AES

- En av de første moderne symmetrisk krypteringsalgoritme som ble benyttet var **DES**, the **Digital Encryption Standard**
- Den ble utviklet av **IBM** på 70 tallet og deretter modifisert av **NSA**
- Utviklingen i datamaskinprosessor i de senere årene førte til at **DES** ikke lenger var sikker mot **brute force attack**
- **Triple-DES (3DES)** er en videreutvikling av krypteringsalgoritmen **DES**. Den bruker **DES** algoritmen tre ganger
- **Advanced Encryption Standard (AES)** også kjent som Rijndael ble en standard i 2002 for å erstatte **DES** og **3DES**.

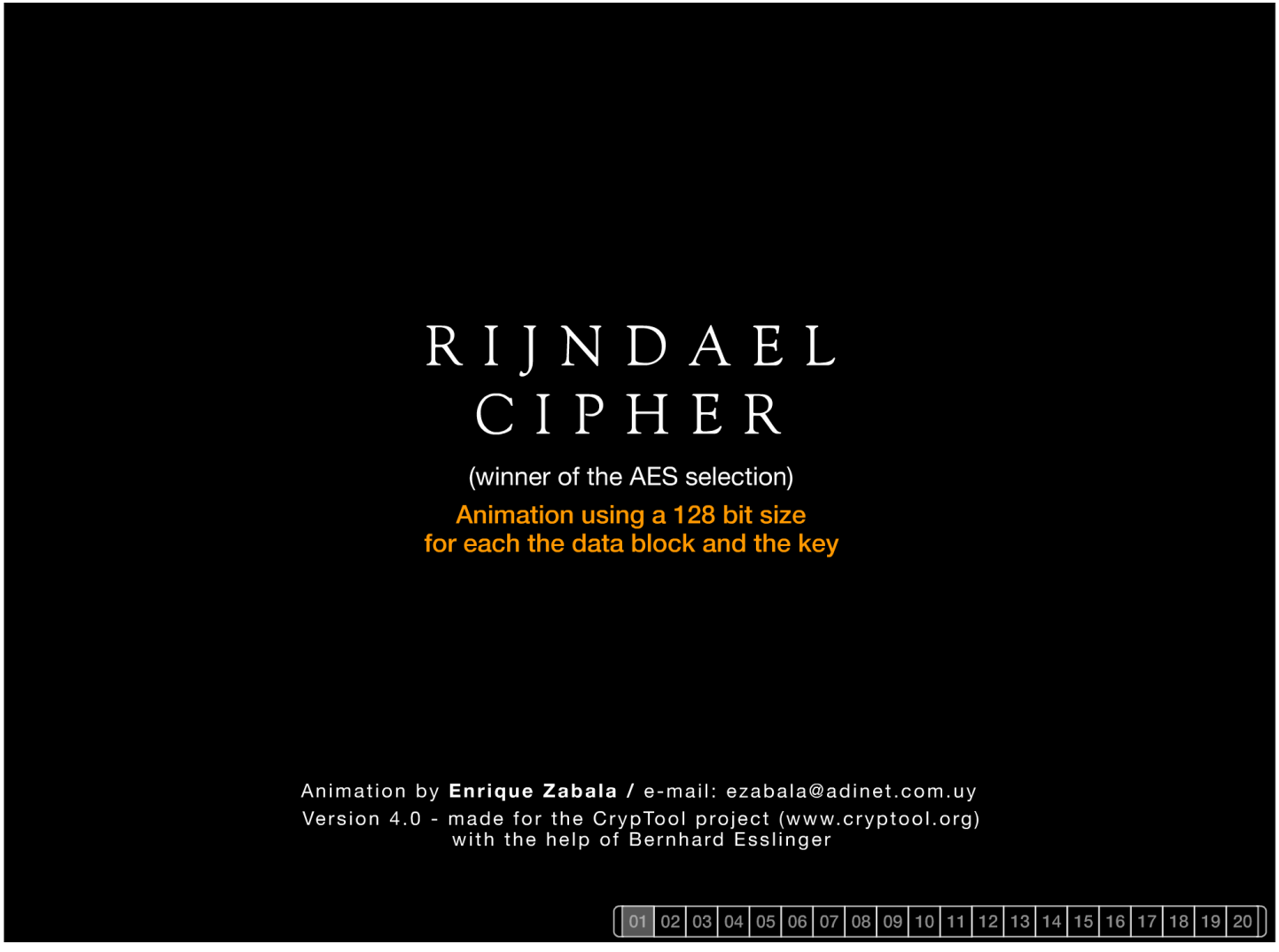
Advanced Encryption Standard (AES)

- Utviklet for å erstatte **DES** etter at National Institute of Standards and Technology (NIST) utlyste en konkuranse om en offentlig symmetrisk kryptering. (Link to competitions)
- Utviklet av to belgiere: Joan Daemon og Vincent Rijmen under navnet Rijndael.
- Blokk og nøkkellengde på henholdvis **128, 192** eller **256** bits
- Motstandsdyktig mot alle kjente kryptoanalysemetoder
- Enkel, rask, lett å implementere i maskin- og/eller programvare

Advanced Encryption Standard (AES)

- De fleste moderne symmetrisk krypteringsalgoritmer er:
  - Block cipher: tar inn data i blokker i fast størrelse;
  - Implementert i runder: utfører liknende operasjoner gjentatte ganger.
- **AES** bruker 10, 12 eller 14 runder for henholdsvis nøklene til 128, 192 og 256 bits.
- Hver runde i AES algoritmen består av fire trinn.
  - **AddRoundKey:** Hver byte i tabellen blir kombinert med en round key", som er utledet fra nøkkelen
  - **SubBytes:** Hver byte i tabellen blir erstattet med en annen basert på en oppslagstabell.
  - **ShiftRows:** Hver rad i tabellen forskyves syklisk med et bestemt antall steg.
  - **MixColumns:** På hver kolonne i tabellen utføres en inverterbar lineær transformasjon.

Demo og oversikt over AES



Source: AES Rijndael Cipher - Visualization

Praktisk bruk av symmetrisk kryptering(1)

- Alice og Bob ønsker å kommunisere med hverandre på en sikker måte ved å bruke **symmetrisk kryptering:**
  - Dette kan de gjøre ved å bruke for eksampel openssl

Alice kan kryptere meldingen med en felles hemmelig nøkkel(01234567)

```
openssl enc -aes-256-cbc -in klartekst.txt -out kryptert.enc -K 01234567 -iv 0000000000
```

```
Openssl opsjoner:
    enc                 Forteller openssl at vi ønsker å benytter symmetrisk kryptering
    -aes-256-cbc        Her ønsker vi å bruke AES med 256 bits nøkkel i CBC mode
    -in klartekst.txt   Filen som innholder klarteksten og som skal krypteres
    -out kryptert.enc   Ønsket navn på den kryterte filen
    -K                  Den hemmelige nøkkelen
    -iv                 initialization vector skal være random generert for å øke sikkerheten
```

Praktisk bruk av symmetrisk kryptering(2)

- Alice og Bob ønsker å kommunisere med hverandre på en sikker måte ved å bruke **symmetrisk kryptering:**
  - Dette kan de gjøre ved å bruke for eksampel openssl

Bob kan dekryptere meldingen med samme hemmelig nøkkel(01234567) og iv

```
openssl enc -d -aes-256-cbc -in kryptert.enc -out ukryptert.txt -K 01234567 -iv 0000000000
```

```
Openssl opsjoner:
    enc                 Forteller openssl at vi ønsker å benytter symmetrisk kryptering
    -d                  Her sier vi at vi vil bruke dekryptering funksjonen
    -aes-256-cbc        Her ønsker vi å bruke AES med 256 bits nøkkel i CBC mode
    -in kryptert.enc    Filen som skal dekrypteres
    -out ukryptert.txt  Navnet på den dekryterte filen. Vi kan kalle den hva vi vil.
    -K                  Den hemmelige nøkkelen
    -iv                 initialization vector skal være random generert for å øke sikkerheten
```

AES DEMO!.

Asymmetrisk Kryptering

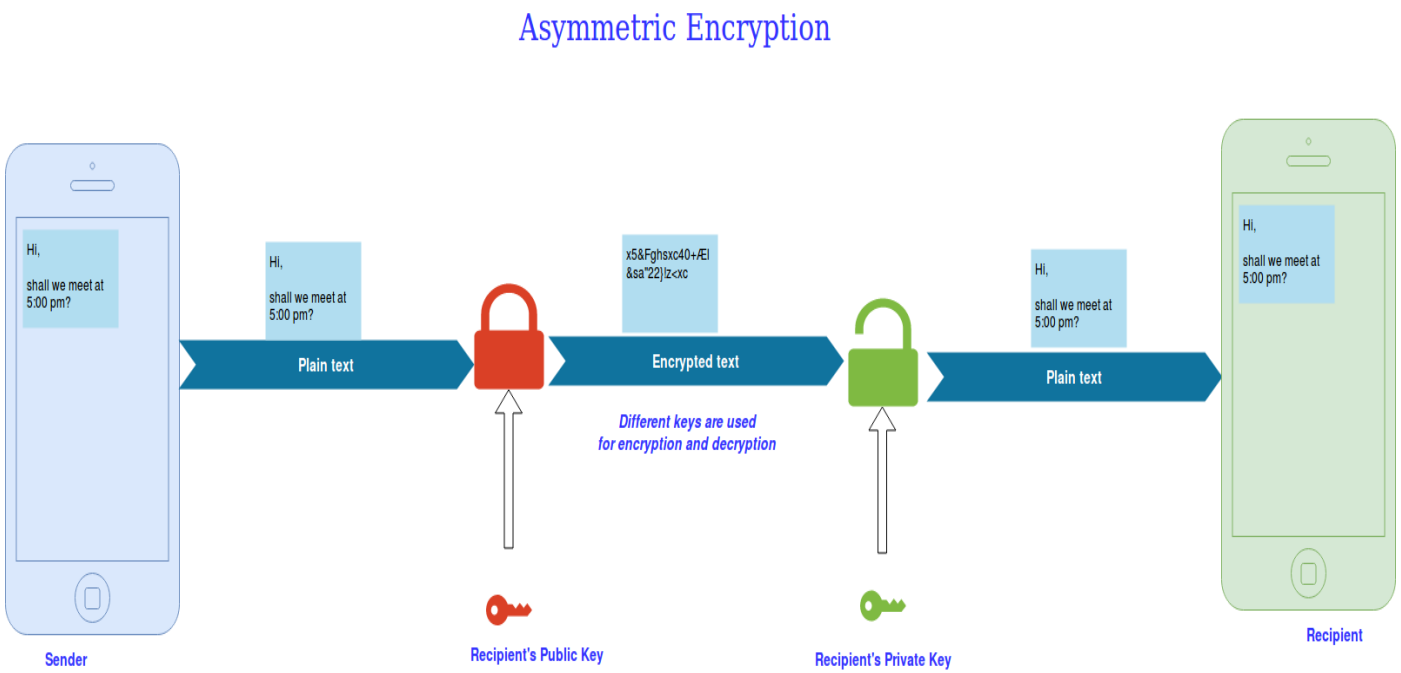
What is Asymmetric Cryptography?

"A modern branch of cryptography (popularly known as public-key cryptography") in which the algorithms use a pair of keys (a public key and a private key) and use a different component of the pair for each of two counterpart cryptographic operations (e.g., encryption and decryption, or signature creation and signature verification)."

Source: RFC 4949 - Internet Security Glossary, Version 2

Asymmetrisk kryptografi

- Bruker et nøkkelpar, en en offentlig, K_{Public} og en hemmelig, $K_{Private}$
  - K_{Public} krypterer eller verifiserer
  - $K_{Private}$ dekrypterer eller singerer
- **Mer om Asymmetrisk kryptering neste uke**



Asymmetrisk kryptering

- I 1976, foreslo **Whitfield Diffie** og **Martin Hellman** offentlig nøkkel kryptering (asymmetrisk kryptering) der ulike nøkler brukes for kryptering og dekryptering.
- Grunnlaget for asymmetrisk kryptering er identifikasjonen av en enveis-funksjon som er lett å beregne, men vanskelig å invertere uten ytterligere informasjon

Rivest-Shamir-Adelman (RSA) algorithmen

- Rivest-Shamir-Adleman (RSA) er en asymmetrisk krypteringsalgoritmen
- RSA er den mest brukte asymmetrisk krypteringsalgoritme i verden. Algoritmen bygger på og er avhengig av vanskelighetsgraden av å faktorisere store tall(primtall).

Modular Math(Modulær matematikk)

- Både Deffie-Hellman og RSA bruke Modulær matematikk/aritmetikk (Modular Math)
  - Modulo-operasjonen regner ut resten når to heltall deles på hverandre.

Eksempel

- $7 \bmod 2 = 1$
- $13 \bmod 5 = 3$

Modulær matematikk/aritmetikk har visse egenskaper som er interessant for Asymmetrisk kryptering

- Nemlig, flere tall kan gi oss samme resttall
  - $\mathbf{3 \bmod 10 = 3}$
  - $\mathbf{13 \bmod 10 = 3}$
  - $\mathbf{23 \bmod 10 = 3}$
  - $\mathbf{33 \bmod 10 = 3}$

Rivest-Shamir-Adelman (RSA) algorithmen

RSA algoritmen

1. Generate two large random primes, **p** and **q**, of approximately equal size such that their product **n = p*q** is of the required bit length, e.g. 1024 bits.
2. Compute **n = p*q** and ϕ **= (p - 1)(q - 1)**
3. Choose an integer **e**, where **1** $<$ **e** $<$ ϕ(n), such that **gcd(e,** ϕ**(n)) = 1**
4. Compute the secret exponent **d**, 1 $<$ **d** $<$ **phi**, such that **e*d** $\equiv$ **1 (mod** ϕ**(n))**
5. The public key is **(n, e)** and the private key **(d, p, q)**. Keep all the values **d, p, q and** ϕ secret.

- **n** is known as the modulus
- **e** is known as the public exponent or encryption exponent or just the exponent
- **d** is known as the secret exponent or decryption exponent.

[Video:] **RSA**

RSA Example

Eksempel

- generate two primes, p and q. In our example 3 and 11
  - p = 3 and q = 11
- The product of p and q is n, in our case:
  - n = 3 * 11 = 33
- ϕ(n) is calculated below:
  - ϕ(n) = (p - 1) * (q - 1) = 2 * 10 = 20
- Now we have to chose an exponent, e that is relatively prime to ϕ(n) = (p-1) * (q-1).
  - This means that e and ϕ(n) must not share factors other than 1($\gcd(e,\phi(n)) = 1$)
  - Let's choose 7 as the value for e (e = 7)
- The pair (e, n) is our public key that is used to encrypt messages.
- Calculate a value for d, which must be chosen so that it is the inverse of e, modulo n
  - e * d $\equiv$ 1 mod ϕ(n)
  - One solution is d = 3 [(3 * 7) mod 20 = 1]
- Public key is (e, n) => (7, 33) and Private key is (d, n) => (3, 33)

The encryption of m = 2 is $c = 2^7 \bmod 33 = 29$

The decryption of c = 29 is $m = 29^3 \bmod 33 = 2$

Praktisk bruk av asymmetrisk kryptering(1)

Alice generer sitt nøkkelpar ved å for eksempel bruke openssl

openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -out privkey-Alice.pem

```
Openssl opsjoner:
    genpkey                          Forteller openssl at vi ønsker å benytter Asymmetrisk kryptering
    -algorithm RSA                   Her sier vi at vi vil bruke RSA kryperingsalgoritmen
    -pkeyopt rsa_keygen_bits:2048    Her ønsker vi å bruke 2048 bits nøkkel av typen RSA
    -out privkey-Alice.pem           Ønsket navn på den private nøkkelen
```

Deretter trekker Alice ut den offentlige delen av nøkkelparet

openssl pkey -in privkey-Alice.pem -out pubkey-Alice.pem -pubout

```
Openssl opsjoner:
    pkey                          Forteller openssl at vi ønsker å benytter Asymmetrisk kryptering
    -in privkey-Alice.pem         Her sier vi at vi vil bruke RSA kryperingsalgoritmen
    -out pubkey-Alice.pem         Her ønsker vi å bruke 2048 bits nøkkel av typen RSA
```

Filen pubkey-Alice.pem inneholder Alice sin offentlige nøkkel og kan disbrueres til venner eller publiseres på nett.

Praktisk bruk av asymmetrisk kryptering(2)

- Bob ønsker å sende en kryptert fil til Alice. Han laster ned hennes offentlige nøkkel

Deretter lager Bob en kryptert melding ved å bruke openssl:

openssl pkeyutl -encrypt -in melding.txt -pubin -inkey pubkey-Alice.pem -out hemmelig.enc

```
Openssl opsjoner:
    pkeyutl                          Forteller openssl at vi ønsker å benytter Asymmetrisk kryptering
    -encrypt                         Her sier vi at vi vil bruke kryptering
    -in melding.txt                  Filen som skal krypteres
    -pubin -inkey pubkey-Alice.pem   Spesifisere hvilken offentlig nøkkel vi ønsker å kryptere filen me
    -out hemmelig.enc                Ønsket navn på den krypterte filen
```

Bob kan nå sende den krypterte filen hemmelig.enc til Alice

Praktisk bruk av asymmetrisk kryptering(3)

- Alice får den krypterte filen og starter prosessen med å dekryptere den

Alice dekryptere filen med openssl ved å bruke sin egen privat nøkkel

```
openssl pkeyutl -decrypt -in hemmelig.enc -inkey privkey-Alice.pem -out melding.txt
```

```
Openssl opsjoner:
    pkeyutl                     Forteller openssl at vi ønsker å benytter Asymmetrisk kryptering
    -decrypt                    Her sier vi at vi ønsker å dekryptere
    -in hemmelig.enc            Filen som skal dekrypteres
    -inkey pubkey-Alice.pem     Spesifisere hvilken privat nøkkel vi ønsker å dekryptere filen me
    -out melding.txt            Ønsket navn på den ukrypterte filen
```

Hvis alt er gjort riktig, skal Alice kunne få se den dekrypterte meldingen fra Bob

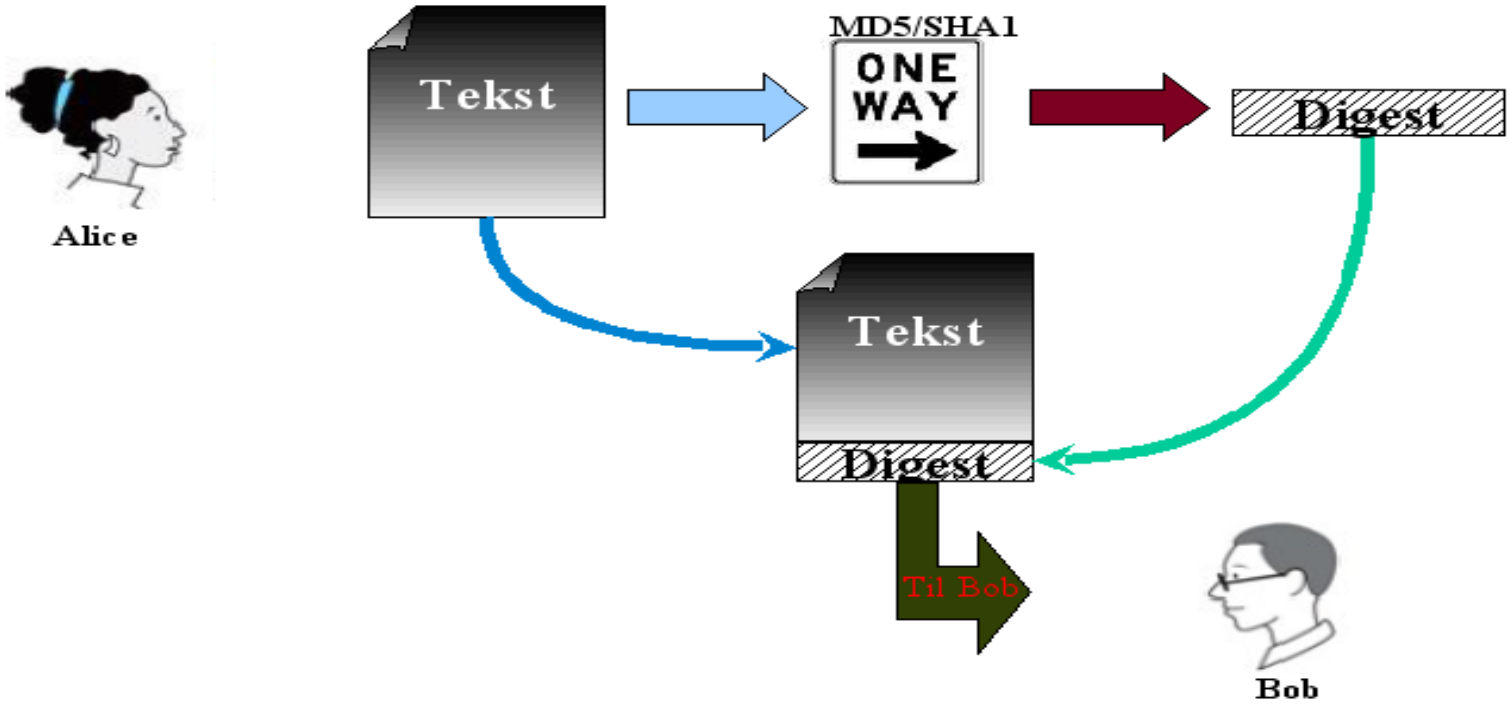
RSA DEMO!.

Digital Signering

- Anta at **Alice** sender en melding **M** til **Bob** med signatur **f(Alice, M)**: Vi vil gjerne at signaturen skal ha visse egenskaper:
  - **Unforgeable(Kan ikke forfalskes):** Det bør være vanskelig for andre enn **Alice** å produsere signaturen **(Alice, M)**
  - **Authentic(Autentisk):** Bob kan bekrefte at Alice Signerte meldingen/dokumentet **M**
  - **Non-repudiation(Ikke-fornektelse):** at **Alice** ikke kan nekte å ha produsert signaturen
  - **Integrity(Integritet):** etter å ha blitt sendt/overført, kan **M** ikke endres
  - **Not reusable:(Kan ikke gjenbrukes:** signaturen kan ikke tas av eller adskilles og brukes på nytt for en annen melding.
- Både Symmetrisk og Asymmetrisk kryptering er velegnet for digitale signaturer

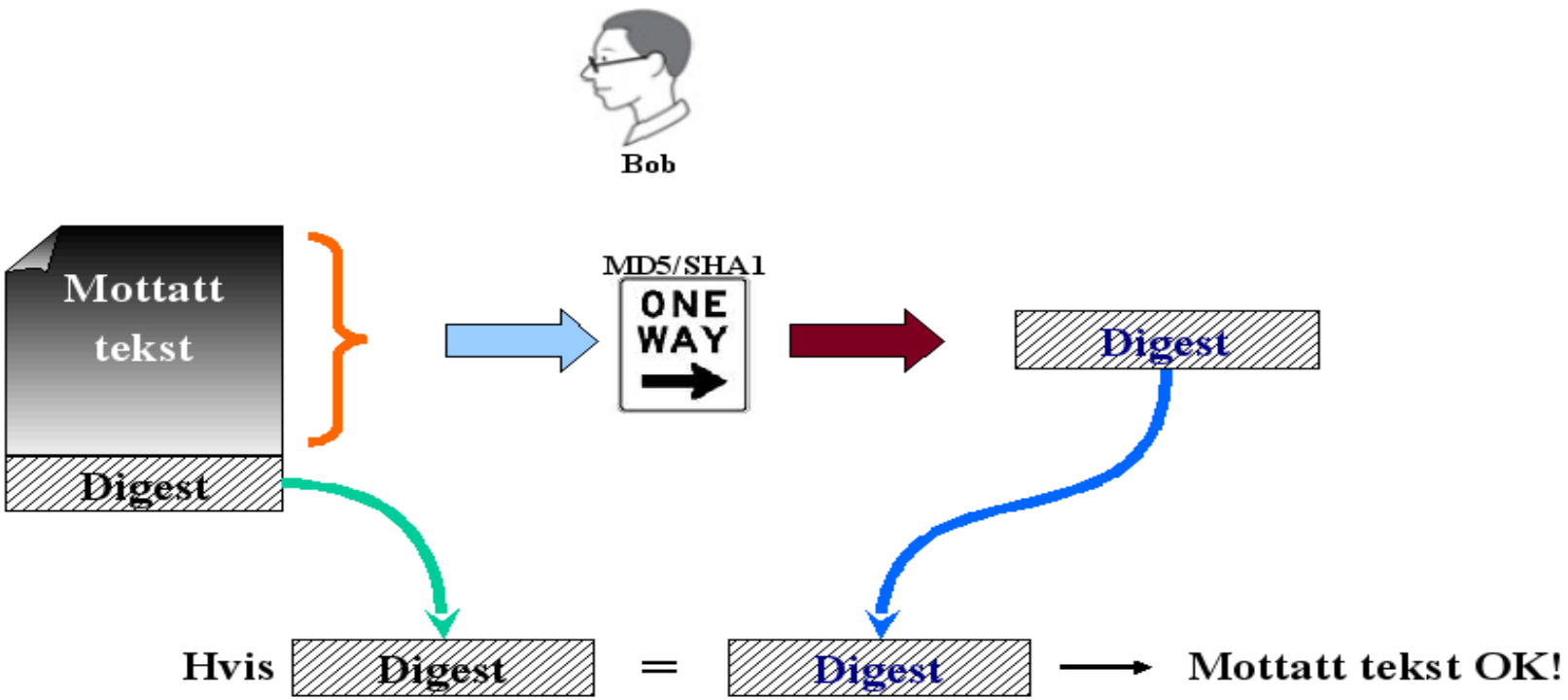
Digital Signering med Symmetrisk algoritmer(1)

- **Alice** og **Bob** må bli enige om en hemmelig nøkkel **K**
  - Nøkkelen må utveksles gjennom en sikker kanal!
  - **Alice** beregner "message Authenctication Code" (MAC), a, v.h.a funksjonen **h** og sender meldingen og MAC til **Bob**



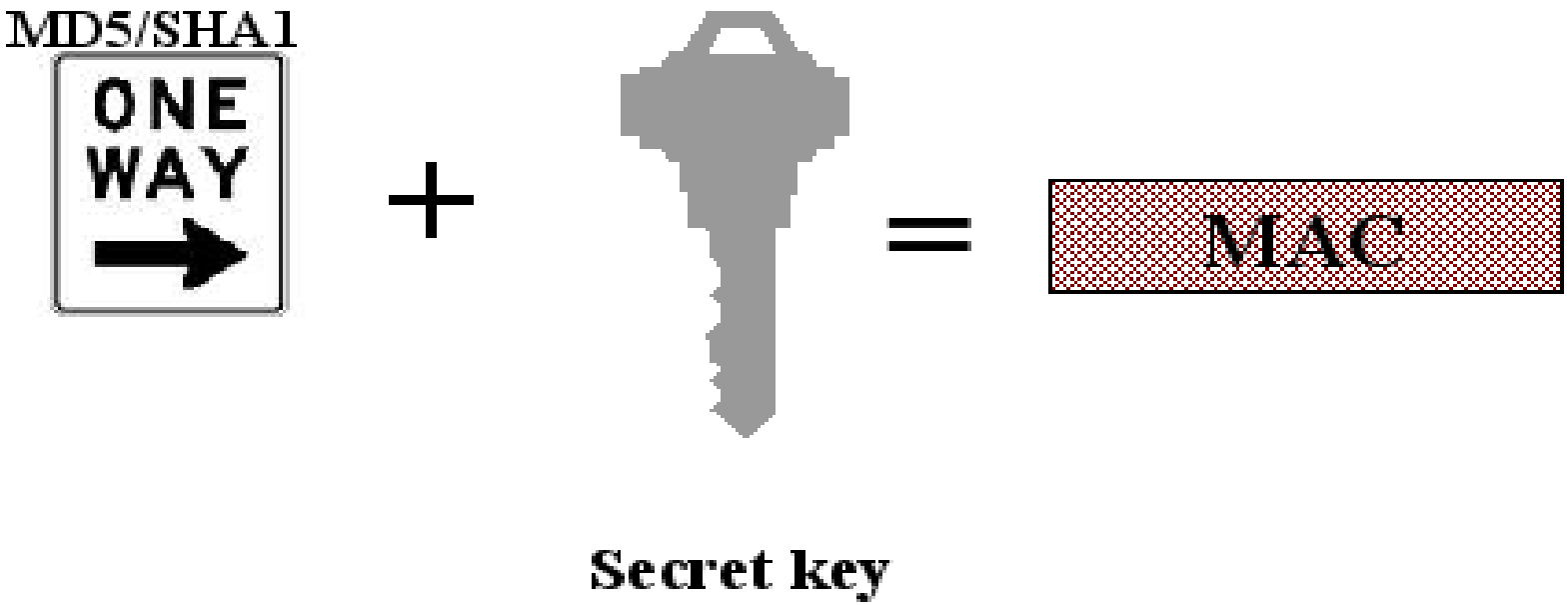
Digital Signering med Symmetrisk algoritmer(2)

- **Bob** beregner MAC på meldingen han mottar og verifiserer at den er den samme som den mottatte MAC. Hvis ikke kaster han meldingen



Digital Signering med Symmetrisk algoritmer(1)

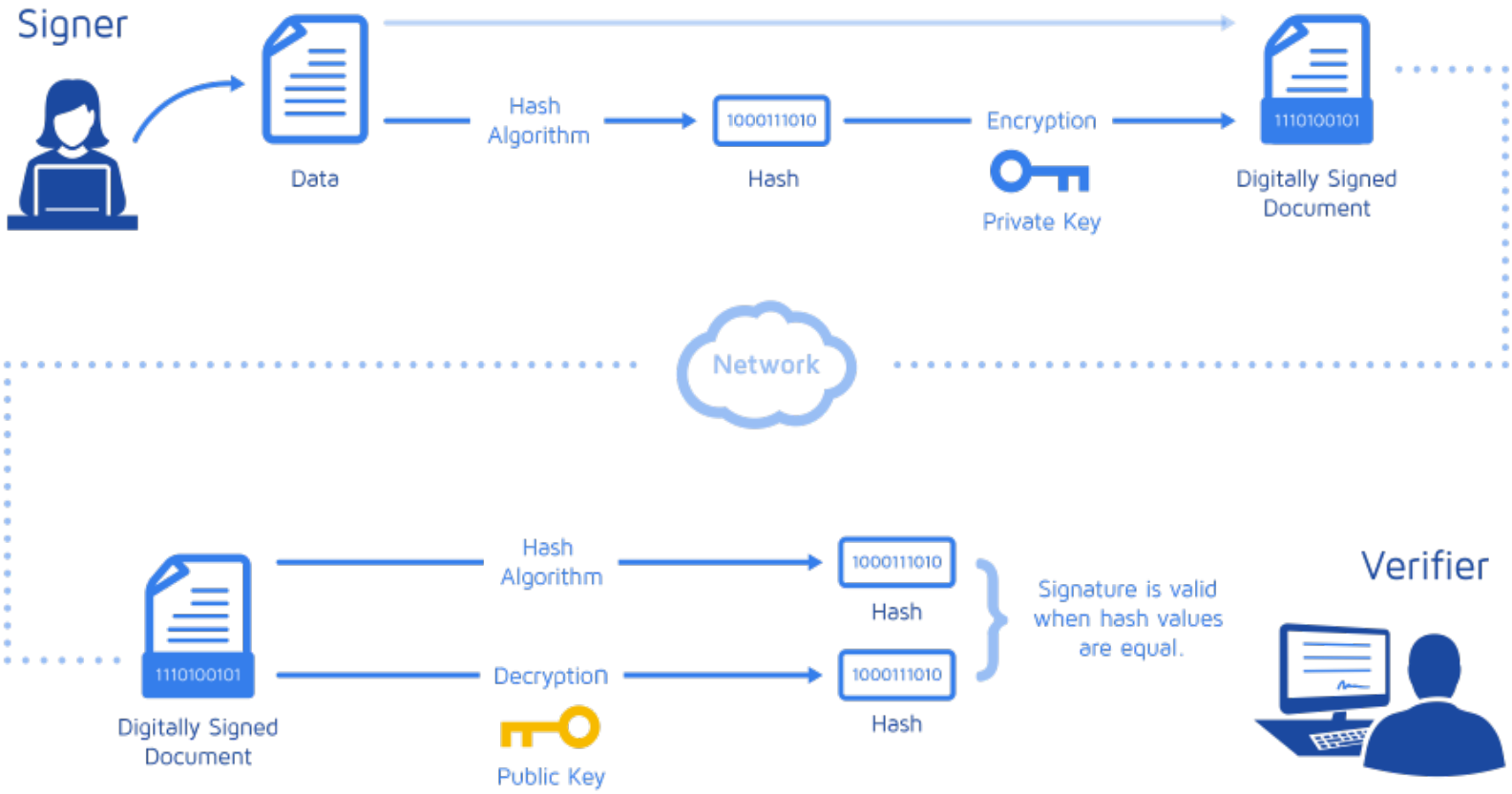
- Hvordan vet **Bob** at meldingen kommer fra **Alice** ?
  - Alice kombinerer hash funksjonen med symmetrisk kryptering
  - **(Hash + kryptering) = MAC**



Digital Signering med Symmetrisk algoritmer - DEMO!.

Digital Signering med Asymmetrisk algoritmer

- **Alice** Kan signere med sin private nøkkel (Private key)
- **Bob** kan verifisere med **Alice** sin offentlige nøkkel (Public key)



Digital Signering med Asymmetrisk algoritmer(1)

- **Alice** genererer et par med nøkler ($K_{privAlice}$,$K_{pubAlice}$)
  - $K_{privAlice}$ er den hemmelige (private) nøkkelen som må holdes hemmelig!
  - Alice publiserer sin offentlige nøkkel $K_{pubAlice}$
- **Alice** beregner en signatur **S** til meldingen **M** med den hemmelige nøkkelen $K_{privAlice}$ og en signeringsfunskjon **H**

Signering med openssl

```
openssl dgst -sha512 -sign privAlice.pem -out hashavfilen.bin Melding.txt
```

Digital Signering med Asymmetrisk algoritmer(2)

- **Bob** bruker en verifikasjonsalgoritme **V** som benytter Alice's offentlige nøkkel, $K_{pubAlice}$, til å verifisere signaturen

Verifisering med openssl

```
openssl dgst -sha512 -verify pubAlice.pem -signature hashverdi.bin Melding.txt
```

Digital Signering med Asymmetrisk algoritmer - DEMO!

Utfordringen med Digital Signering ved hjelp av Asymmetrisk algoritmer

- Hvordan kan **Bob** være sikker på at det faktisk er **Alice** sin offentlige nøkkel og ikke noen andre sin?
- Kanskje **Eva** lagde et nøkkelpar, publisert den offentlige nøkkelen og utgav seg for å være **Alice???**

Utfordringen med Digital Signering ved hjelp av Asymmetrisk algoritmer

- En løsning til problemet er å benytte en annen person/enhet som både **Alice** og **Bob** har tilitt til som mellomledd
- Dette kalles **Trusted Third Party (TTP)**
  - **TTP** kan sende en kryptert melding som kunn **Alice** og **Bob** kan lese for å bekrefte/bevise identiteten til hverandre
    - M.a.o: "Jeg, TTP går god for at den offentlige nøkkelen $K_{pubAlice}$ tilhører **Alice**"
  - **TTP** løsninger som brukes i dag er:
    - sertifikater som er signert av en **CA(Certificate Authority)**
    - **Pretty Good Privacy (PGP)** Web-of-Trust

?