

---

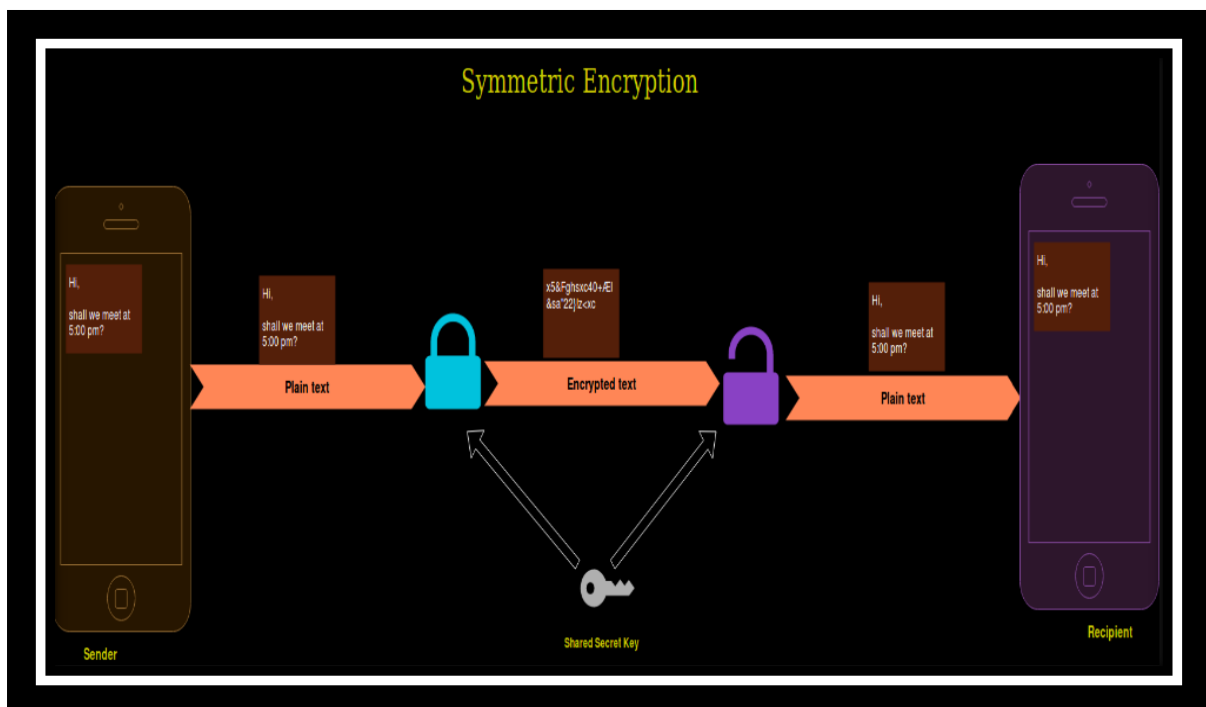
## *Symmetric and asymmetric encryption*

### *(Modern cryptographic algorithm)*

---

#### **SYMMETRISK KRYPTOGRAFI**

Her brukes en hemmelig nøkkel for begge krypteringsoperasjoner, kryptering og dekryptering.



#### **DES, 3DES (Digital Encryption Standard)**

En av de første moderne symmetriske krypteringsalgoritmene var DES hvor nøkkelstørrelsen var på 56 bits. Det ble senere oppdaget at det ikke er trygt mot brute force angrep. I 1993 ble dermed utviklet Triple-Des (3DES) som brukte DES algoritmen tre ganger med en nøkkel lengde på 56 bits. Hvis to nøkler brukes tilsvarer det nøkkellengde på 112 bits og hvis tre = 168 bits.

#### **AES (Advanced Encryption Standard)**

Ble utviklet for å bytte ut DES etter at NIST annonserte en konkurranse for offentlig symmetrisk kryptering.

Blokk og nøkkellengder er 128, 192 og 256 bits.

De mest moderne symmetriske krypteringsalgoritmene er:

- Block cipher: tar inn data i blokker av fikset størrelse
- Implementert i runder: utfører liknende operasjoner igjen og igjen

AES bruker 10, 12 eller 14 runder for nøkler henholdsvis 128, 192 og 256 bits. Hver runde i AES algoritmen består av 4 steg

([https://www.youtube.com/watch?v=gP4PqVGudtg&ab\\_channel=AppliedGo](https://www.youtube.com/watch?v=gP4PqVGudtg&ab_channel=AppliedGo)):

1. **AddRoundKey (Legg til rundenøkkel):** Hver byte i tabellen kombineres med en "rundenøkkel", som er avledet fra hovednøkkelen.
2. **SubBytes (Erstatt bytes):** Hver byte i tabellen erstattes med en annen basert på en oppslagstabell.
3. **ShiftRows (Flytt rader):** Hver rad i tabellen blir flyttet sirkulært med et visst antall steg.
4. **MixColumns (Miks kolonner):** En inverterbar lineær transformasjon utføres på hver kolonne i tabellen.

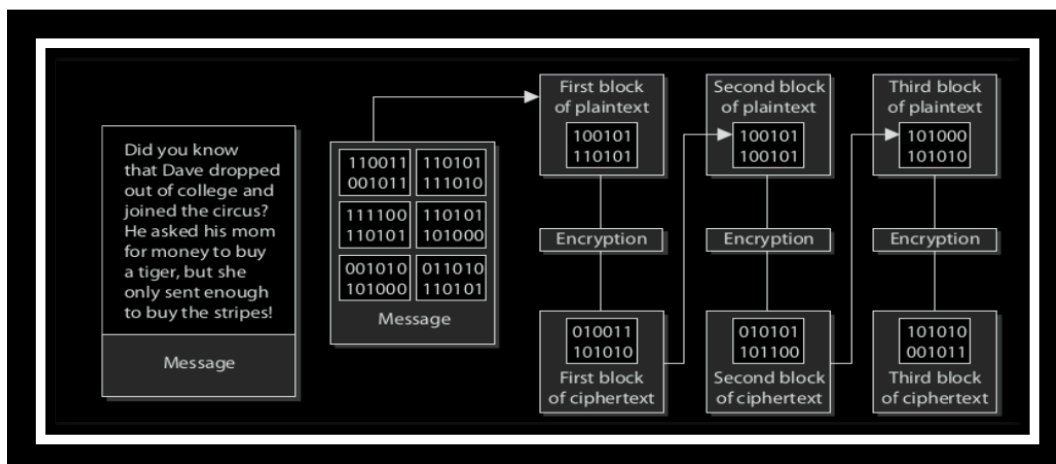
## BLOCK CIPHERS

Krypterer blokker med data på en fiksert lengde/størrelse.

Deler data inn i blokker som er X mengde biter og krypterer dem til blokker som også har X mengde biter. Blokk størrelsene varierer avhengig på algoritmen (fra 56 biter til 256).

Symmetrisk kryptering som bruker blokk cipher metoden er:

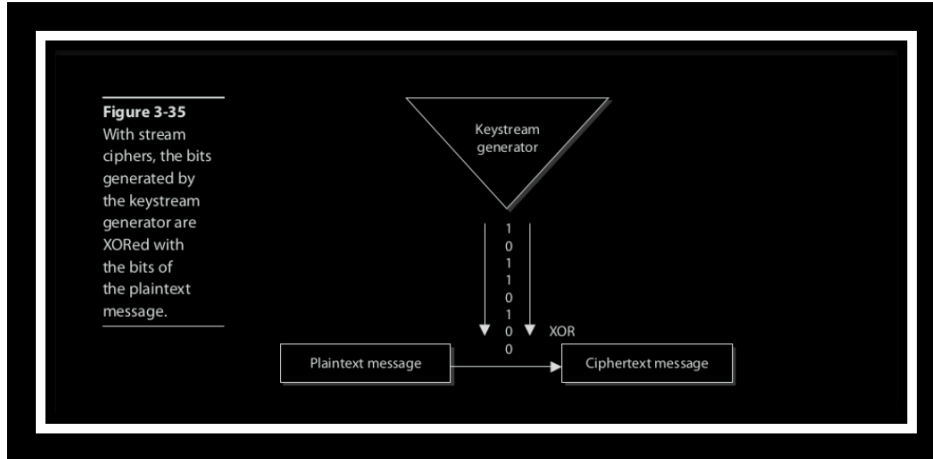
DES, 3DES, AES, RC6, Blowfish, Twofish, Serpent og IDEA



## STREAM CIPHER

Krypterer data en bit eller byte av gangen (ASCII OG UTF). Symmetrisk kryptering ved bruk av strøm cipher metoden er:

Salsa20, ChaCha, SEAL, TWOPRIME, RC4, A5(GSM)

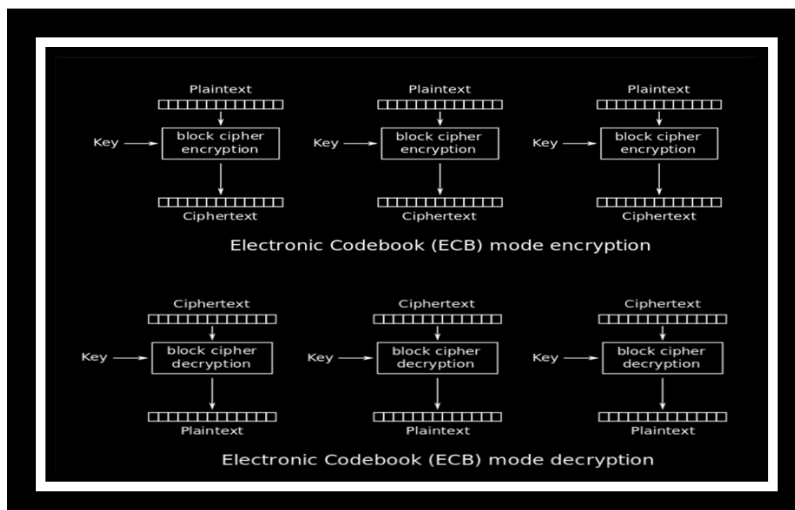


## HVORDAN BRUKE BLOKK CIPHER?

Blokk cipher: krypterer blokker med data av fikserte lengder:

- DES bruker 56 bit blokker mens AES bruker 128 bit blokker, 192 bit og 256 bit blokker.
- Vi trenger en måte å kryptere en meldinger for en hvilken som helst lengde
  - o For eks: melding med størrelse 1 MB

National institute of standard technology (NIST) definerer flere måter å gjøre det på som de kaller «modes of operation». Den letteste måten å bruke symmetrisk kryptering på er ved å kryptere hver blokk med data (Plaintext) med den samme nøkkelen. Denne krypteringsmetoden kalles Electronic Code Book (ECB).



## ELECTRONIC CODE BOOK

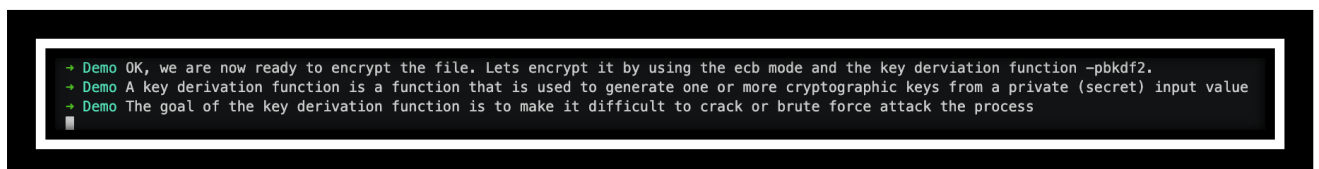
Skjuler ikke data mønstre, passer ikke for lange meldinger eller store filer.



## SVAKHETEN TIL ECB

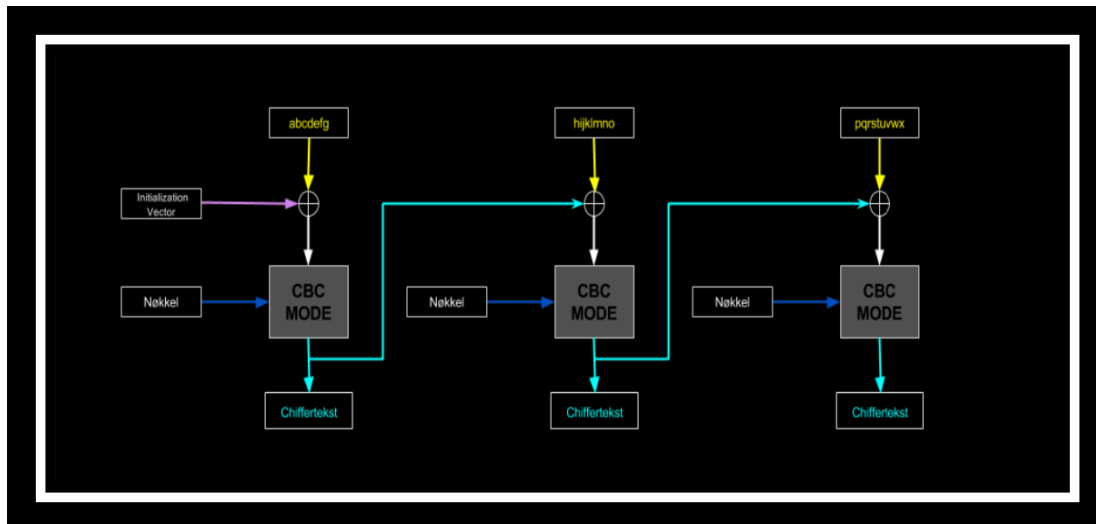
Electronic Codebook (ECB) er en blokk-krypteringsmodus der hver blokk med inngangsdata krypteres uavhengig, og den samme inngangsblokken vil alltid produsere den samme krypterte utdataen. Dette kan føre til sikkerhetsproblemer og er derfor sjelden brukt alene i moderne kryptografi, spesielt når det er gjentatte blokker i dataene. Andre moduser som **Cipher Block Chaining** (CBC) brukes ofte for bedre sikkerhet og beskyttelse mot mønsterangrep.

KEY DERIVATION MED ECB MODE I DEMO VIDEO TIL ISMAIL!

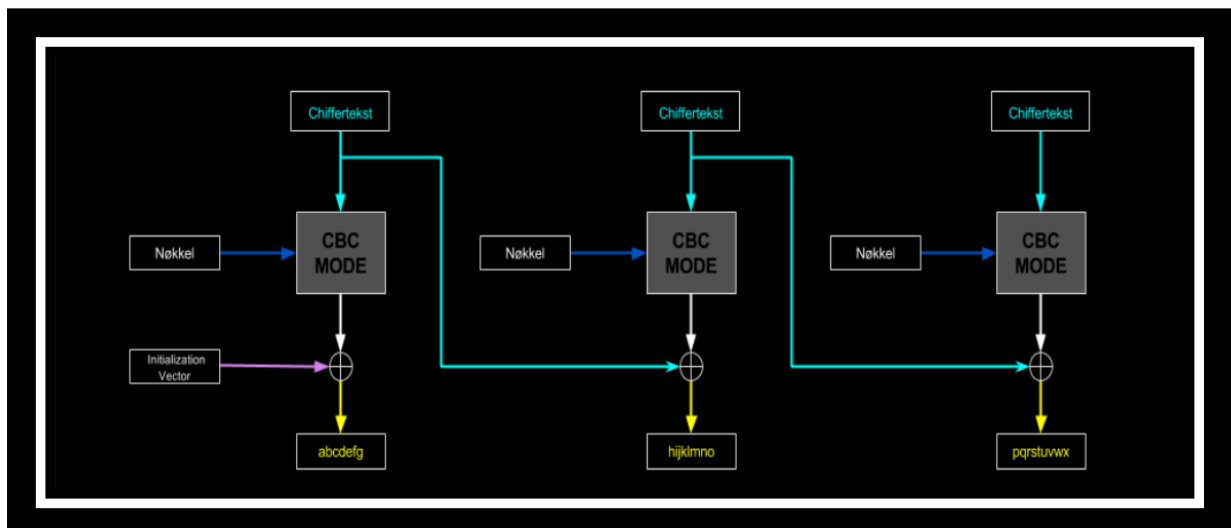


## CIPHER BLOCK CHAINING

En løsning på problemet med ECB fordi CBC tilfeldig blander blokkene før de krypteres. CBC utfører en XOR operasjon på hvert etterfølgende planitext blokk med den forrige digit blokken også utfører krypteringen. For å gjøre hver melding unik, må en initialiseringsvektor (IV) brukes i den første blokken. En initialiseringsvektor (IV) trenger ikke å være hemmelig, men det er viktig at den er tilfeldig og at den aldri blir brukt igjen med den samme nøkkelen.



Over er det vist hvordan man vil kunne kryptere med cipher block chaining og under er det vist hvordan man kan dekryptere med cipher block chaining:



Under er det en tabell med kjente «modes of operations»:

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	•Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	•General-purpose block-oriented transmission •Authentication
Cipher Feedback (CFB)	Input is processed $s$ bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	•General-purpose stream-oriented transmission •Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.	•Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements

## MESSAGE PADDING

Symmetriske algoritmer som 3DES og AES opererer med blokker av ren tekstdata. For at dette skal fungere, må lengden på ren tekst være nøyaktig lik blokk lengden eller et heltall multiplum av blokk lengden som brukes av algoritmen.

I virkeligheten kan ren tekst være lengre eller kortere enn en hel blokk (64, 128, 192 eller 256 bits).

For å få en hel blokk, brukes en metode som kalles "padding":

Padding går ut på å legge til noen ekstra biter på slutten av data-/teksten for å oppnå en blokk på 64 eller 128 biter. Under legges de 5 vanlige padding metodene til:

- ▶ Pad with bytes all of the same value as the number of padding bytes ( )
- ▶ Pad with 0x80 followed by zero bytes
- ▶ Pad with zeroes except make the last byte equal to the number of padding bytes
- ▶ Pad with zero (null) characters
- ▶ Pad with space characters

<http://www.di-mgt.com.au/cryptopad.html>

## STYRKER OG SVAKHETER VED SYMMETRISK KRYPTOGRAFI

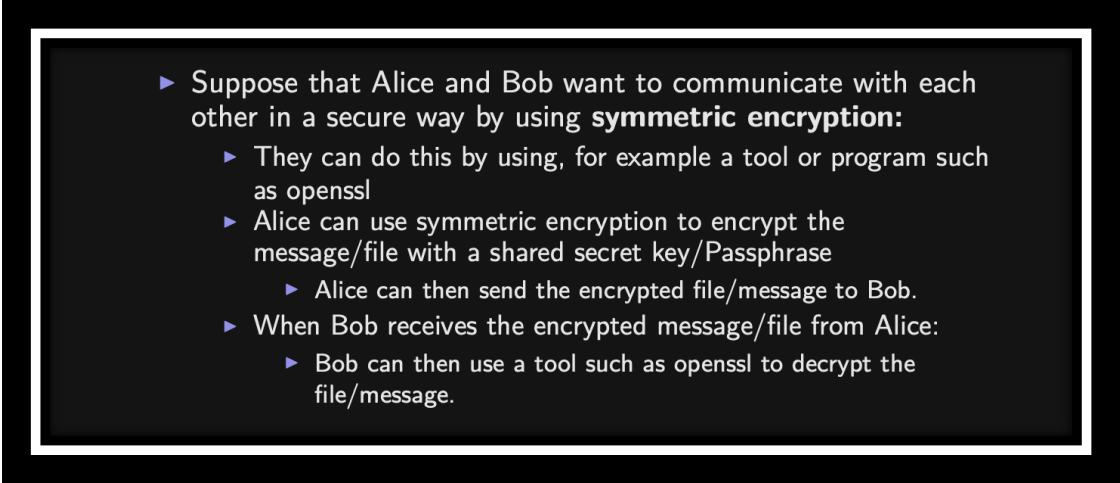
Styrker:

- Mye raskere (mindre data intensivt) enn asymmetriske systemer
- Vanskelig å knekke dersom man bruker store nøkkel størrelser

Svakheter:

- Krever en sikker mekanisme for å levere nøkler ordentlig
- Hvert par av brukere trenger en unik nøkkel, så når antall individuelle øker, så gjør antall mengder av nøkler, som muliggjør å lage nøkkel håndterings overveldende
- Tilbyr konfidensialitet, men ikke autensitet eller non-repudiation

## PRAKTISK BRUK AV SYMMETRISK KRYPTERING

- 
- ▶ Suppose that Alice and Bob want to communicate with each other in a secure way by using **symmetric encryption**:
    - ▶ They can do this by using, for example a tool or program such as openssl
    - ▶ Alice can use symmetric encryption to encrypt the message/file with a shared secret key/Passphrase
      - ▶ Alice can then send the encrypted file/message to Bob.
    - ▶ When Bob receives the encrypted message/file from Alice:
      - ▶ Bob can then use a tool such as openssl to decrypt the file/message.

## ASYMMETRISK KRYPTOGRAFI

I 1976 introduserte Diffie og Hellman offentlig nøkkel kryptografi (asymmetrisk kryptografi) ved å bruke ulike nøkler for kryptering og dekryptering. Basisen for asymmetrisk kryptografi er identifiseringen av en enveiskrypteringsfunksjon som er enkel å beregne, men vanskelig å reversere uten ytterligere informasjon.

## RSA ALGORITMEN

Rivest-Shamir-Adleman (RSA) er en asymmetrisk krypterings algoritme. Algoritmen er basert på og avhenger av en vanskelighetsgrad av å faktorisere store tall (primtall).

Både Diffie-Hellman og RSA bruker modulro matte/aritmetikk. Modulo operasjonen regner rest når to heltall deles.

## Eksempel

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

Modular mathematics/arithmetic has certain properties that are interesting for Asymmetric encryption

- ▶ Namely, several numbers can give us the same residual number
  - ▶  $3 \bmod 10 = 3$
  - ▶  $13 \bmod 10 = 3$
  - ▶  $23 \bmod 10 = 3$
  - ▶  $33 \bmod 10 = 3$

Under her ser vi måten Alice lager privat og offentlig nøkkel på. DEMO RSA.

```
+ Demo This Demo is created by Ismail Hassan for the course ITPE3100 to demonstrate The concept of RSA asymmetric encryption/decryption using Openssl
+ Demo suppose that Alice and Bob wants to communicate with each other securely using RSA Asymmetric algorithm.
+ Demo RSA has different key sizes, but in this Demo we will use the key size 2048 bits
+ Demo There are plenty of tools one can use for that purpose, but in this Demo we will use Openssl
+ Demo Alice must first generate her private key using openssl

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

+ Demo The command will generate the private key and create the file privkey-Alice.pem
+ Demo Alice can then derive the public from the generated private key thus making them unique and mathematically connected.

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

+ Demo The command will create the public key pubkey-Alice.pem
+ Demo Alice can now share her public key with Bob or any other person that Alice wants them sending her encrypted messages
+ Demo since Alice is the only person that has her private key privkey-Alice.pem, she will be the only person capable of decrypting messages encrypted with her public key pubkey-Alice.pem
■
```

Under ser vi også hvordan bob lager en fil og krypterer den med en offentlig nøkkel. Den kan

```
+ Demo suppose that Bob receives Alice's public key. Bob will then be able communicate confidentially with Alice by encrypting messages or files to Alice with her public key
+ Demo OK, let Bob create a file containing a secret messages

+ Demo echo 'Top Top Top secret' > top-secret.txt

+ Demo Lets check the content of the file with the command cat
+ Demo cat top-secret.txt
Top Top Top secret
+ Demo OK, Bob is now ready to encrypt the file. Bob then encrypts the file with Alice's public key pubkey-Alice.pem

+ Demo openssl pkeyutl -encrypt -in top-secret.txt -pubin -inkey pubkey-Alice.pem -out top-secret.enc

+ Demo Bob, can the send the encrypted file to Alice
+ Demo Alice, after receiving the encrypted file from Bob, can decrypt it with her private key privkey-Alice.pem

+ Demo openssl pkeyutl -decrypt -in top-secret.enc -inkey privkey-Alice.pem -out file-from-Bob.txt

+ Demo The command will decrypt the file and create the file file-from-Bob.txt. Alice can then read the content using any editor, but in this Demo we will use the cat command
+ Demo cat file-from-Bob.txt
Top Top Top secret
```

kun da åpnes med personen som har den private nøkkelen som i dette tilfellet er Alice.



## RSA ALGORITME I DYBDEN

### RSA algorithm

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, 2048 or 4096 bits.
  2. Compute **n = p\*q** and **φ = (p - 1)(q - 1)**
  3. Choose an integer **e**, where  $1 < e < \phi(n)$ , such that  $\text{gcd}(e, \phi(n)) = 1$
  4. Compute the secret exponent **d**,  $1 < d < \phi(n)$ , such that  $e*d \equiv 1 \pmod{\phi(n)}$
  5. The public key is (**e**, **n**) and the private key (**d**, **n**). Keep all the values **d**, **p**, **q** and **φ** secret. One can actually discard/throw away **p** and **q** once **n** is calculated
- ▶ **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.

La oss si at vi nå har våre offentlige og private nøkler. Hvordan fungerer dette egentlig?

#### Kryptering i RSA-algoritmen:

Hvis du trenger å kryptere melding M med din offentlige nøkkel (e, n), blir følgende formel

brukt:  $C = M^e \bmod n$

#### Dekryptering i RSA-algoritmen:

Deretter må du dekryptere meldingen med din private nøkkel (d), så blir følgende formel

brukt:  $M = C^d \bmod n$

## ASYMETRISK KRYPTERING I ACTION

### ALICE LAGER NØKKELPAR (MED OPENSSL):

Alice generates her key pair by using openssl, for example

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

```
Openssl opsjoner:
genpkey           Tells openssl that we want to use Asymmetric encryption
-algorithm RSA    Here we say that we want to use the RSA encryption algorithm
-pkeyopt rsa_keygen_bits:2048 Here we want to use 2048 bit key of the type RSA
-out privkey-Alice.pem Desired name of the private key
```

### ALICE TAR UT DEN OFFENTLIG NØKKELEN



```
Openssl opsjoner:
pkey             Tells openssl that we want to use Asymmetric encryption
-in privkey-Alice.pem Here we specify the private key to use
-out pubkey-Alice.pem Desired name of the public key
```

Fila som heter pubkey-Alice.pem inneholder Alice sin offentlige nøkkel og kan deles til venner eller publiseres offentlig.

### STYRKER OG SVAKHETER VED ASYMMETRISK KRYPTOGRAFI

STYRKER	SVAKHETER
<ul style="list-style-type: none"><li>- Bedre nøkkel distribusjon</li><li>- Bedre skalerbarhet enn symmetriske systemer</li><li>- Kan tilby bedre autentisering/autentisitet og ikke-avvisning</li></ul>	<ul style="list-style-type: none"><li>- Jobber så mye mer tregt enn symmetriske systemer</li><li>- Matematisk intensiv oppgave spesielt når man bruker store nøkler</li></ul>

### DIGITAL SIGNERING

Hva er en digital signering?

Digital signering er som et elektronisk stempel som viser at en melding eller et dokument ikke er endret og kommer fra den riktige avsenderen. Det hjelper mottakeren med å verifisere at informasjonen er ekte og ikke er blitt manipulert på vei.

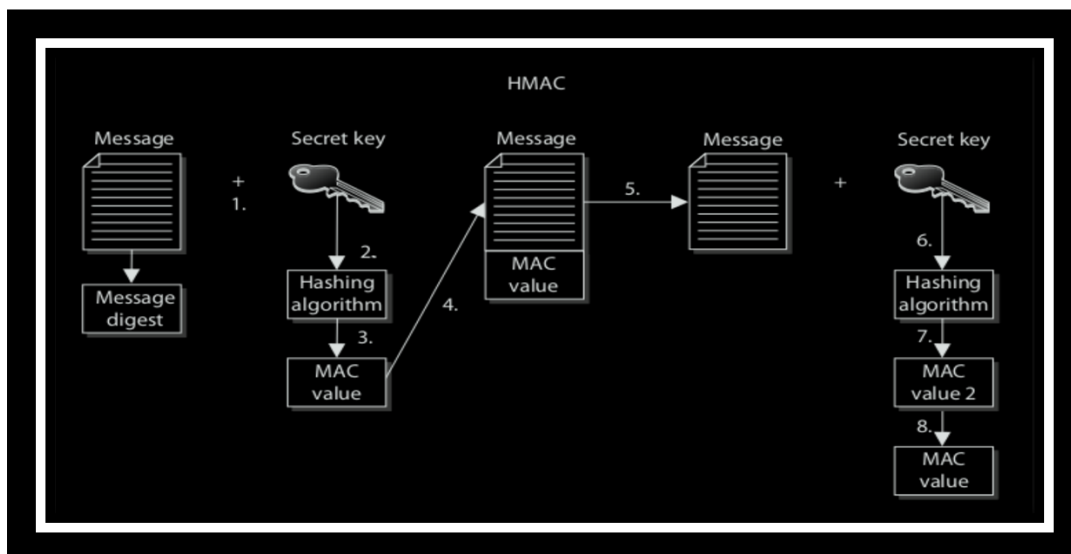
### ØNSKEDE ATTRIBUTTER MED EN DIGITAL SIGNATUR

Vi ønsker at en digital signatur skal ha følgende:

- Uglemmelig: det burde være vanskelig for hvem som helst andre enn senderen å produsere signaturen
- Autentisk: mottakeren kan bekrefte at senderen har signert meldingen eller dokumentet
- Ikke-avvisning (non-repudiation): den som har signert kan ikke nekte å ha produsert signaturen
- Integritet: etter å ha blitt sendt kan meldingen verken endres eller bli endret på
- Ikke gjenbrukbar: signaturen kan ikke fjernes eller gjenbrukes for en annen melding

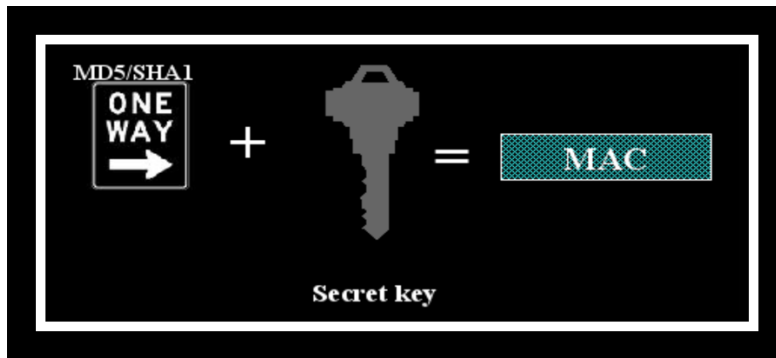
## BÅDE SYMMETRISK OG ASYMMETRISK KRYPTERING PASSER TIL DIGITAL SIGNERING.

Alice og Bob må dele en hemmelig nøkkel K og dette må gjøres via en sikker kanal. Alice kan regne «message Authentication Code» (MAC) ved å bruke funksjonen h og sende meldingen og MAC til Bob. Da regner bob MAC på meldingen han mottar og verifiserer at det er den samme som den mottatte MAC. Hvis ikke kaster han meldingen.



Hvordan vet Bob at meldingen faktisk kommer fra Alice? Alice kombinerer hash funksjonen med symmetrisk kryptering. Vi kan si at:

$$(\text{Hash} + \text{encryption}) = \text{MAC}$$



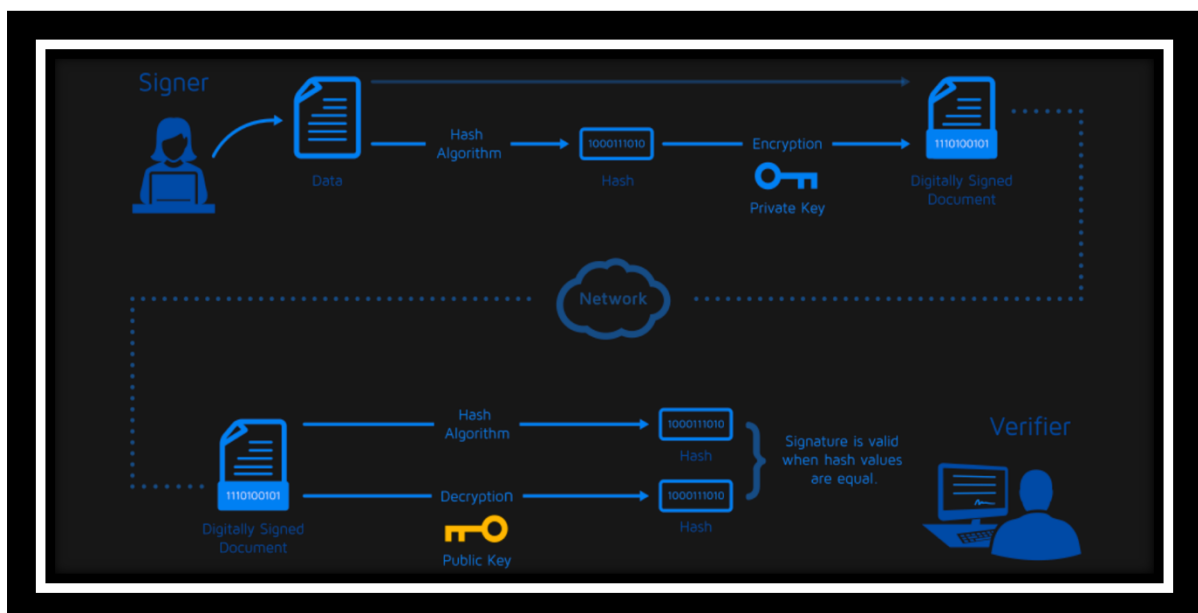
Under er bildet tatt fra demo til digital signering med symmetrisk algoritmer. Det er slik vi bruker det hemmelige delte nøkkelordet datasikkerhet med algoritmen sha512 til å kryptere fila msg.txt.

```

→ Demo we can now start the signing process:
→ Demo openssl dgst -sha512 -hmac datasikkerhet msg.txt
HMAC-SHA512(msg.txt)= 1876fde8a359d4e4b7d1b0934b76db4db113f8

```

## DIGITAL SIGNERING MED ASYMMETRISK ALGORITMER



Under regner Alice en signatur S til meldingen M med en hemmelig nøkkel  $K(\text{privAlice})$  og en signeringsfunksjon  $H$ :

```

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

```

Bob på den andre siden bruker en verifikasjonsalgoritme  $V$  som benytter Alice sin offentlige nøkkel,  $K(\text{publicAlice})$ , for å verifisere signaturen.

## UTFORDRINGER MED ASYMMETRISK ALGORITMER

Hvordan vet bob for sikkert at det er Alice sin offentlige nøkkel og ingen andres? Kanskje Eva lagde et nøkkelpar, publiserte den offentlige nøkkelen og later som om hun er Alice?

En løsning er å bruke en annen person/entitet som både Alice og Bob har stolt på som en mellomperson. Dette kalles Trusted Third Party (TTP). TTP kan sende en kryptert melding så Alice og Bob kan lese og bekrefte/bevise hverandres identiteter.

TTP løsninger i dag er:

- Sertifikater signert av CA (certificate authority)
- Pretty good privacy (PGP) Web-of-Trust

## LEGGER TIL EKSTRA PENSUM OM CIPHER FEEDBACK FORDI DET KOM SOM EN DEL AV QUIZEN (CFB):

Cipher feedback er en type blokk-kryptering som brukes i kryptografi for å muliggjøre strømkryptering av data. CFB-modus gir en måte å behandle data i mindre biter enn blokkstørrelsen til krypteringsalgoritmen.

Question 8 0 / 5 pts

What type Block cipher mode of operation does the figure below illustrates?

You Answered

Correct Answers

- CFB
- Cipher Feedback
- Cipher Feedback (CFB)