Vad en dator inte kan

Schillerska kulturdag 2021-12-13

Fredrik Engström
Docent i Logik
Institutionen för filosofi, lingvistik och vetenskapsteori

Göteborgs Universitet

Vad en dator inte kan (och vad den kan dåligt)

Schillerska kulturdag 2021-12-13

Fredrik Engström
Docent i Logik
Institutionen för filosofi, lingvistik och vetenskapsteori

Göteborgs Universitet

- Att kunna beräkna snabbt blir allt viktigare.

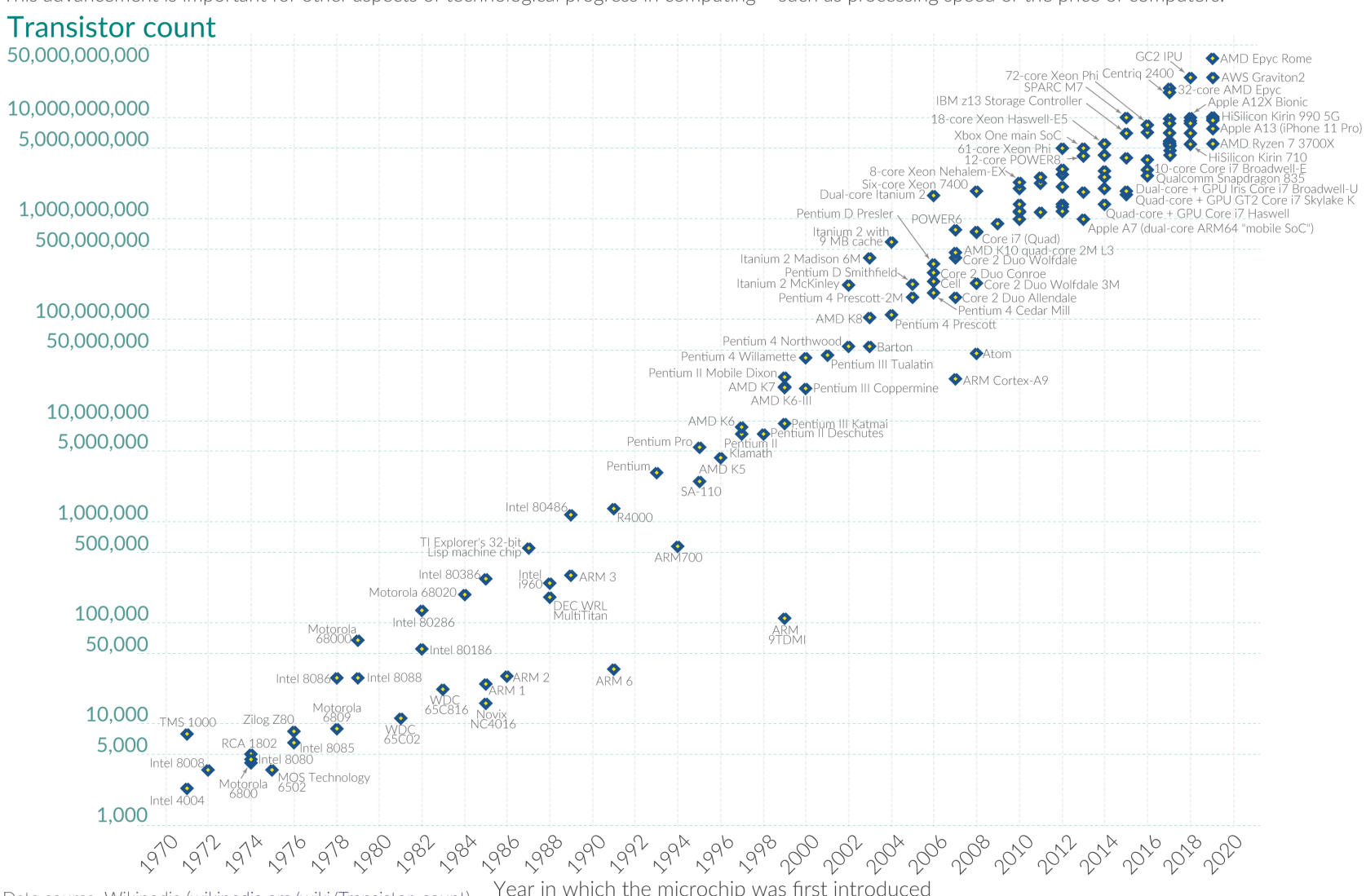
- Att kunna beräkna snabbt blir allt viktigare.
- Men är snabba beräkningar alltid tillräckliga?

Moores lag

Moore's Law: The number of transistors on microchips doubles every two years

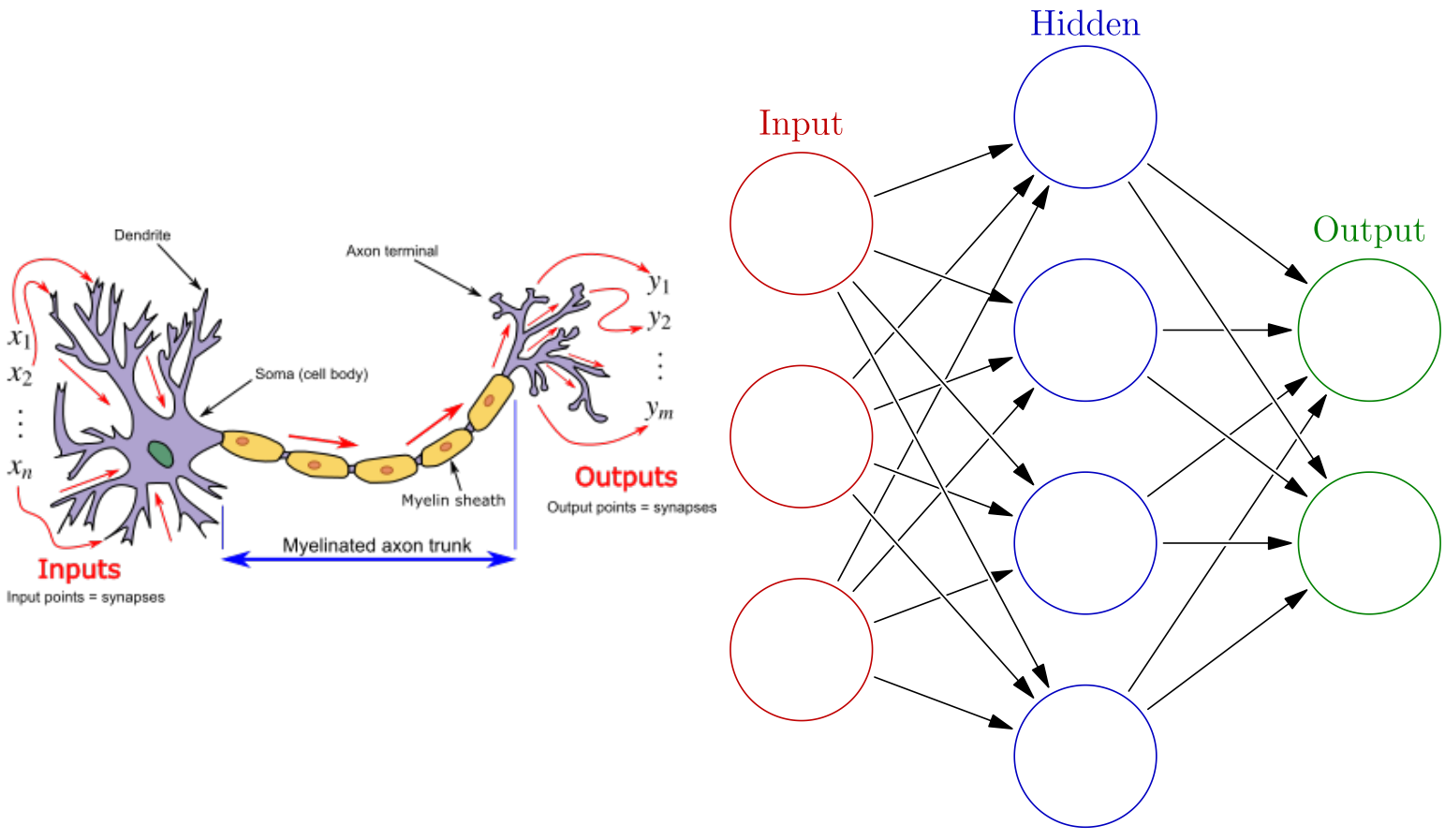
Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)

Nya algoritmer



Egm4313.s12 and Glosser.ca at English Wikipedia, CC BY-SA 3.0, via Wikimedia Commons

Nya typer av datorer

IBM Research, CC BY 2.0, via Wikimedia Commons

Vad är en dator?

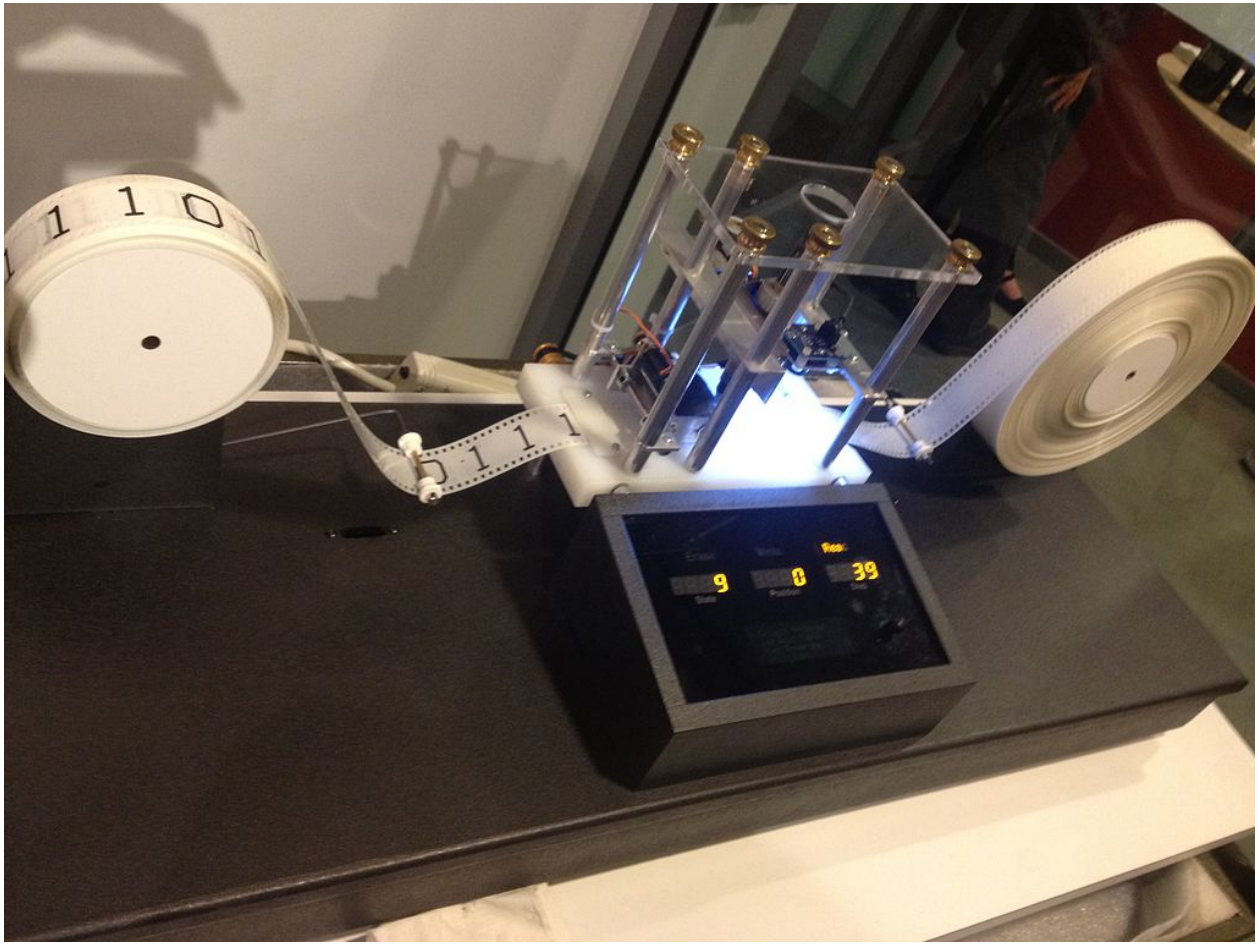
- Grundläggande principen är dock den samma.

Vad är en dator?

- Grundläggande principen är dock den samma.
- Allt ingår i **Alan Turings** definition från 1936: Turingmaskin.

Vad är en dator?

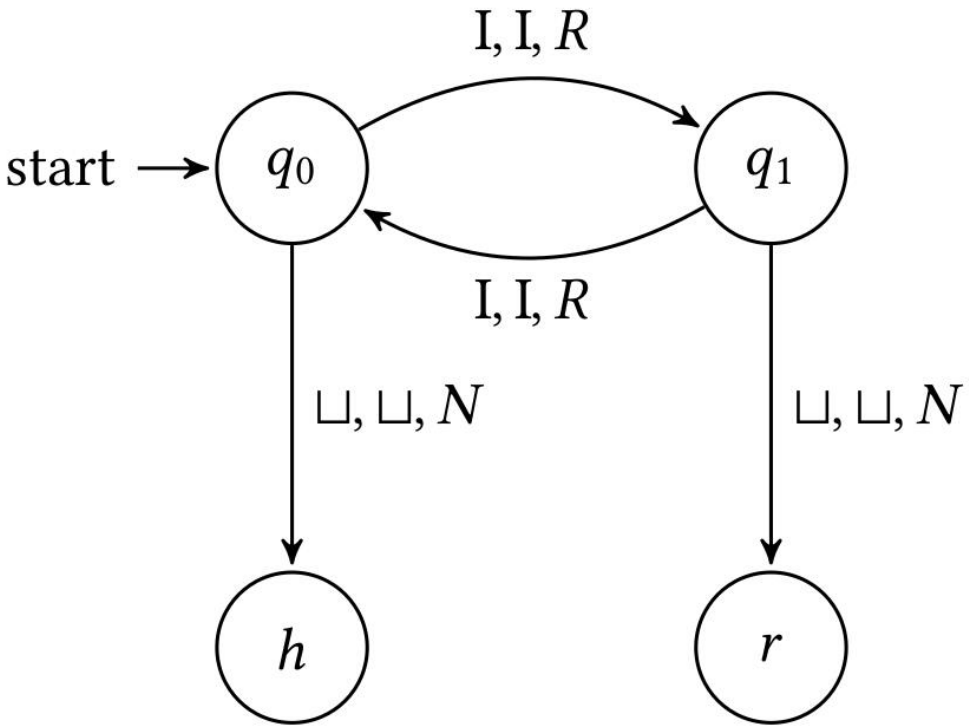
- Grundläggande principen är dock den samma.
- Allt ingår i **Alan Turings** definition från 1936: Turingmaskin.



GabrielF, CC BY-SA 3.0, via Wikimedia Commons

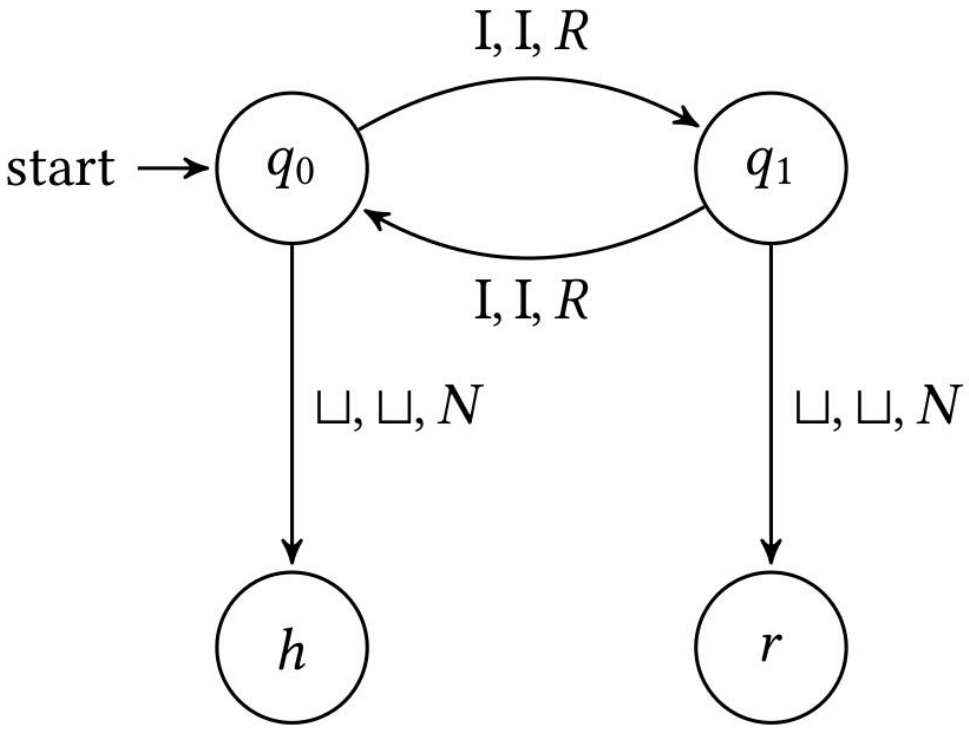
Vad är en dator?

- Grundläggande principen är dock den samma.
- Allt ingår i **Alan Turings** definition från 1936: Turingmaskin.



Vad är en dator?

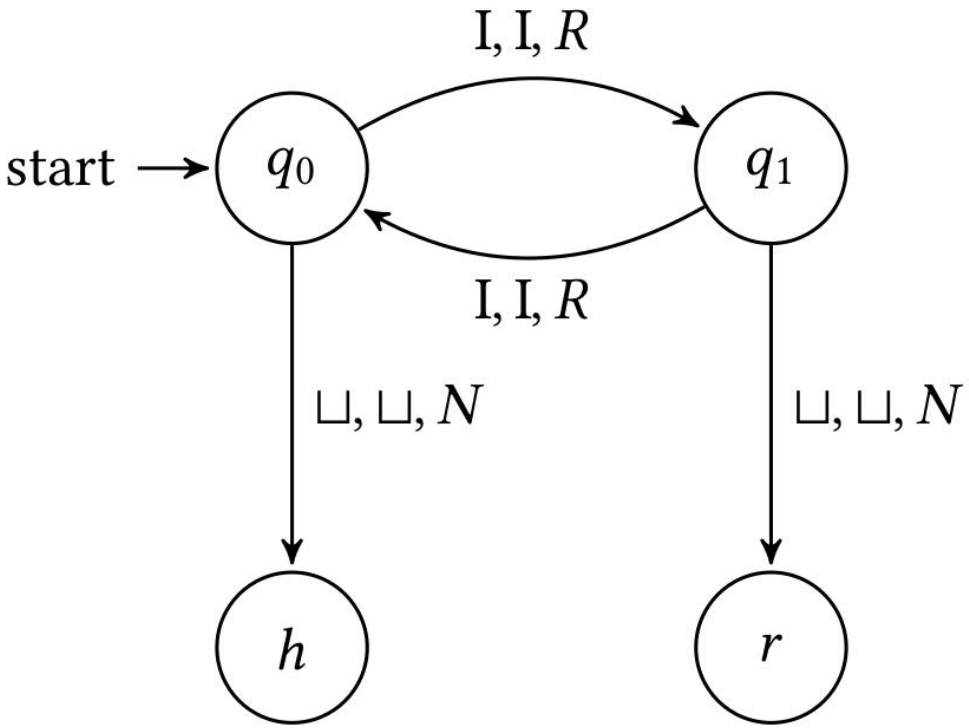
- Grundläggande principen är dock den samma.
- Allt ingår i **Alan Turings** definition från 1936: Turingmaskin.



- Allt som en Turingmaskin kan beräkna är beräkningsbart.

Vad är en dator?

- Grundläggande principen är dock den samma.
- Allt ingår i **Alan Turings** definition från 1936: Turingmaskin.



- Allt som en Turingmaskin kan beräkna är beräkningsbart.
- Allt som är beräkningsbart kan (i princip) beräknas med en Turingmaskin.

Haltproblemet

Halt-programmet

`Halts(P,W)` avgör om en körning av program `P` med indata `W` terminerar eller hamnar i en oändlig loop.

Halt-programmet

`Halts(P,W)` avgör om en körning av program `P` med indata `W` terminerar eller hamnar i en oändlig loop.

- Om program `P` med indata `W` terminerar så returnerar `Halts(P,W)` 1.

Halt-programmet

`Halts(P,W)` avgör om en körning av program `P` med indata `W` terminerar eller hamnar i en oändlig loop.

- Om program `P` med indata `W` terminerar så returnerar `Halts(P,W)` 1.
- Annars 0.

Diag

```
def Diag(P):
  if (Halts(P, P)):
      while True: pass
  else:
      return 0
```

Diag

```
def Diag(P):
  if (Halts(P, P)):
      while True: pass
  else:
      return 0
```

Diag(Diag)

Diag

```
def Diag(P):
  if (Halts(P, P)):
      while True: pass
  else:
      return 0
```

Diag(Diag)

- Om `Diag(Diag)` terminerar så hamnar vi i en oändlig loop.

Diag

```
def Diag(P):
  if (Halts(P, P)):
      while True: pass
  else:
      return 0
```

Diag(Diag)

- Om `Diag(Diag)` terminerar så hamnar vi i en oändlig loop.
- Om `Diag(Diag)` hamnar i en oändlig loop så terminerar programmet (och returnerar `0`).

Diag

```
def Diag(P):
  if (Halts(P, P)):
      while True: pass
  else:
      return 0
```

Diag(Diag)

- Om `Diag(Diag)` terminerar så hamnar vi i en oändlig loop.
- Om `Diag(Diag)` hamnar i en oändlig loop så terminerar programmet (och returnerar `0`).

Alltså: Någon sådant program `Halts(P,W)` kan ej finnas.

Möjliga, men svåra problem

Svåra problem

- Haltproblemet är **olösbart**.

Svåra problem

- Haltproblemet är **olösbart**.
- Andra problem kan vara lösbara, men **svåra**, d.v.s., tar lång tid för en dator att lösa.
- Komplexitetsteori studerar vilka problem som har snabba algoritmer.

Komplexitet: P och NP

- Komplexitetsmått: Hur många “steg” ett program tar.

Komplexitet: P och NP

- Komplexitetsmått: Hur många "steg" ett program tar.
- Förhållandet mellan storlek på indata och antal steg.

Komplexitet: P och NP

- Komplexitetsmått: Hur många "steg" ett program tar.
- Förhållandet mellan storlek på indata och antal steg.
- $F(n) =$ antal steg med indata av storlek n.

Komplexitet: P och NP

- Komplexitetsmått: Hur många "steg" ett program tar.
- Förhållandet mellan storlek på indata och antal steg.
- $F(n) =$ antal steg med indata av storlek n.
- P är klassen av alla problem som kan lösas med ett program där $F(n) \leq Cn^k$ för något C och k.

Komplexitet: P och NP

- Komplexitetsmått: Hur många "steg" ett program tar.
- Förhållandet mellan storlek på indata och antal steg.
- $F(n) =$ antal steg med indata av storlek n.
- P är klassen av alla problem som kan lösas med ett program där $F(n) \leq Cn^k$ för något C och k.
- P brukar sägas bestå av praktiskt lösbara problem.

Komplexitet: P och NP

- Komplexitetsmått: Hur många "steg" ett program tar.
- Förhållandet mellan storlek på indata och antal steg.
- $F(n) =$ antal steg med indata av storlek n.
- P är klassen av alla problem som kan lösas med ett program där $F(n) \leq Cn^k$ för något C och k.
- P brukar sägas bestå av praktiskt lösbara problem.
- NP är istället klassen av problem där lösningar kan **verifieras** i högst Cn^k steg.

Komplexitet: P och NP

- Komplexitetsmått: Hur många "steg" ett program tar.
- Förhållandet mellan storlek på indata och antal steg.
- $F(n) =$ antal steg med indata av storlek n.
- P är klassen av alla problem som kan lösas med ett program där $F(n) \leq Cn^k$ för något C och k.
- P brukar sägas bestå av praktiskt lösbara problem.
- NP är istället klassen av problem där lösningar kan **verifieras** i högst Cn^k steg.
- Största olösta problemet inom datavetenskap: Är P=NP?

Ett av sju "Millennium Problems": $1.000.000

Faktorisering

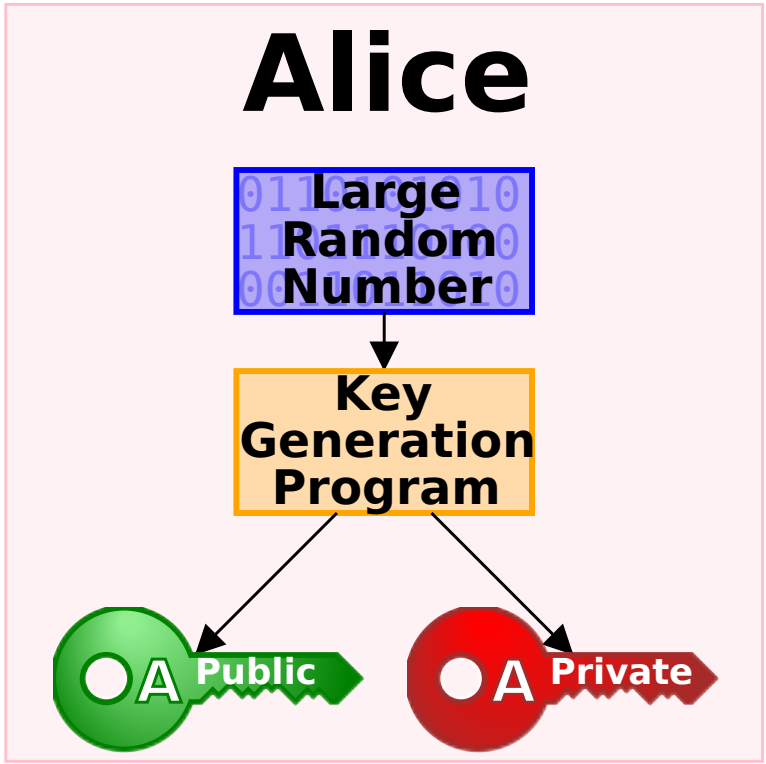
$$3225 = 43 \times 75$$

Faktorisering

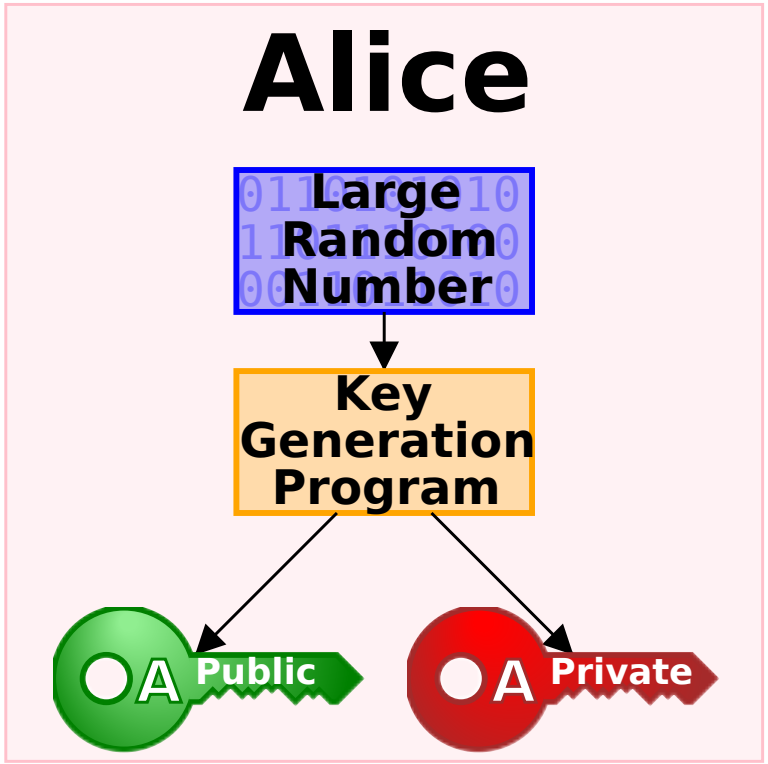
$$3225 = 43 \times 75$$

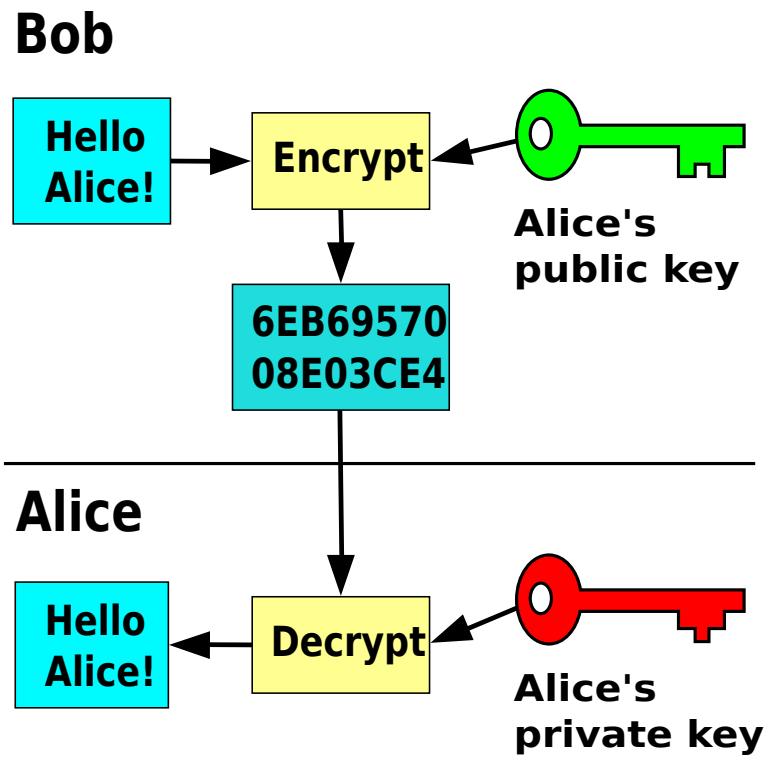
$$188198812920607963838697239461650439807163563379417382700763356422988859715234665485319060606504743045317380113033967161996923212057340318795506569962213051687593076502570593980750864240649373971255005503864911990643623425267084063851895759463889572617685833173 \times 472772146107435302536223071973048224632914695302097116459852171130520711256363590397527$$

Publika krypton

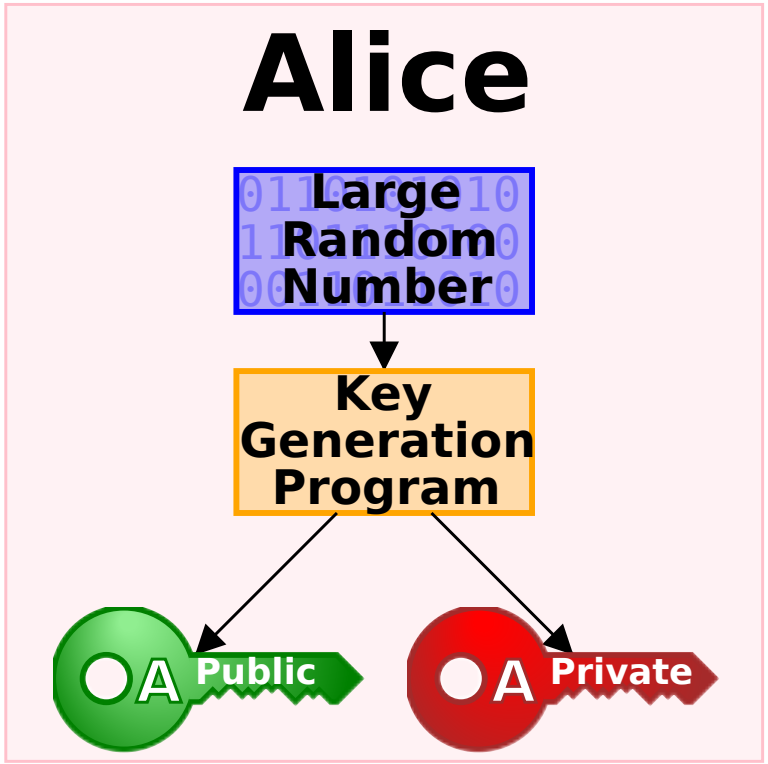


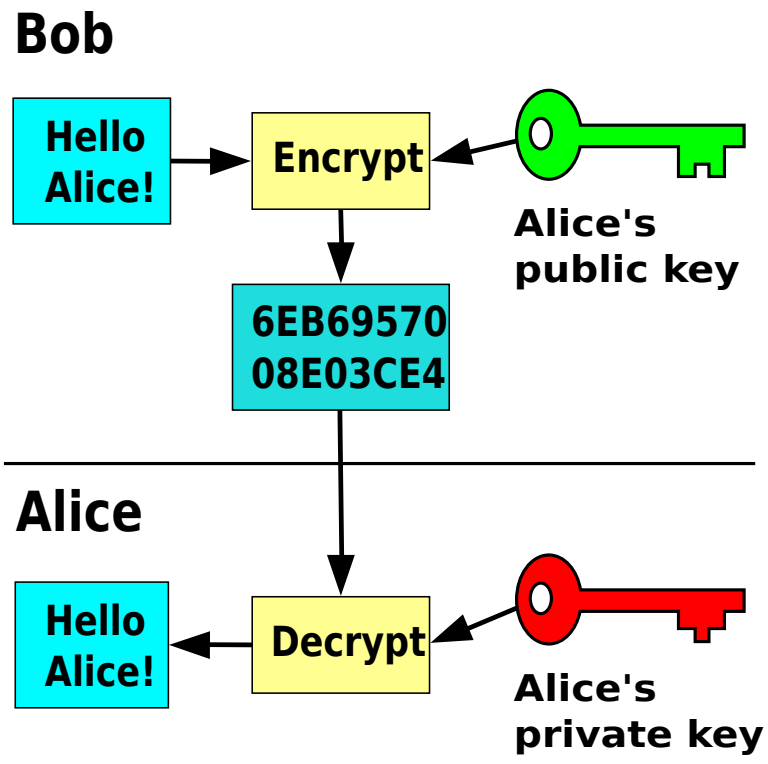
Publika krypton





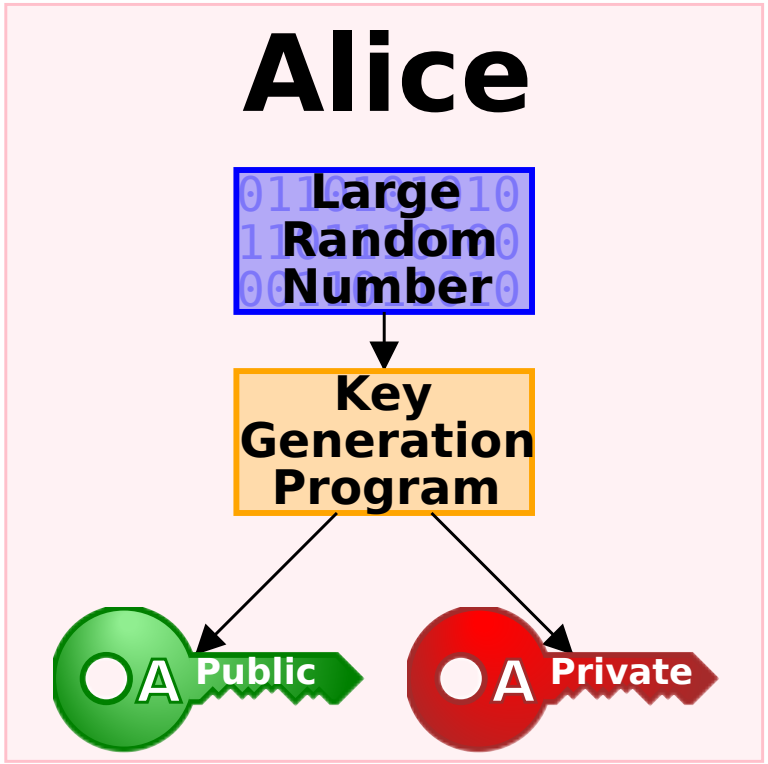
Publika krypton

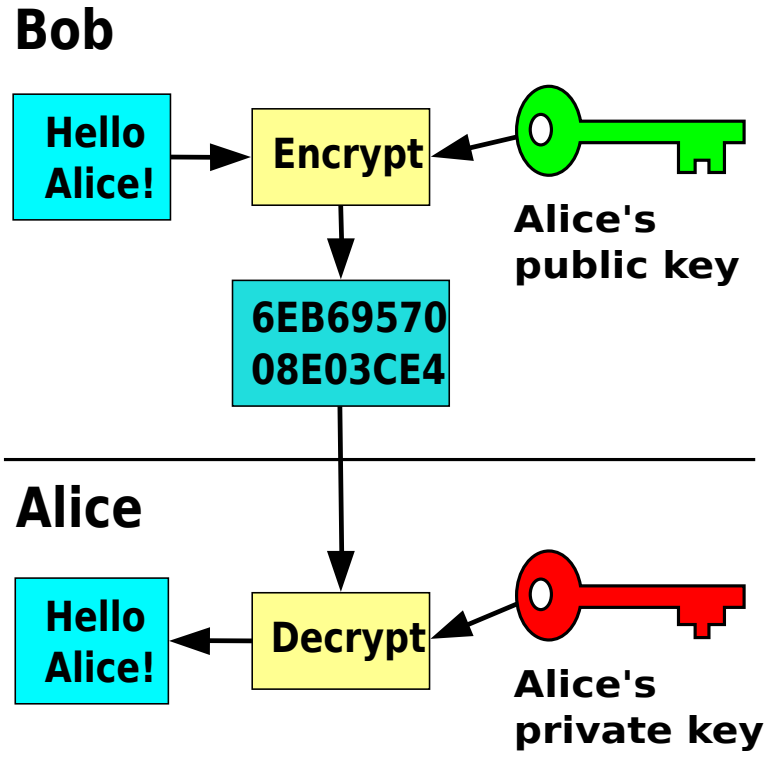




- RSA används bla i SSL.

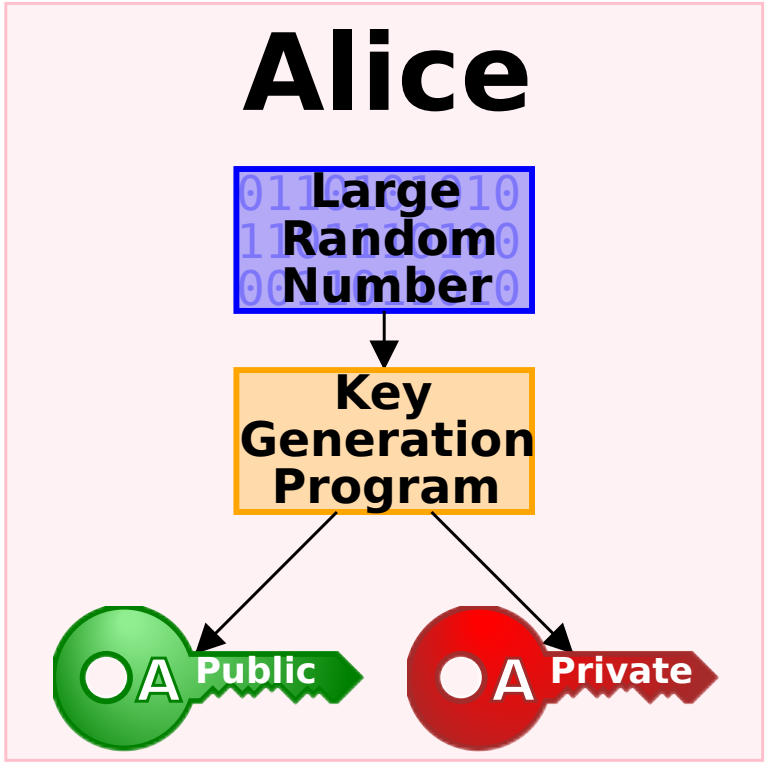
Publika krypton

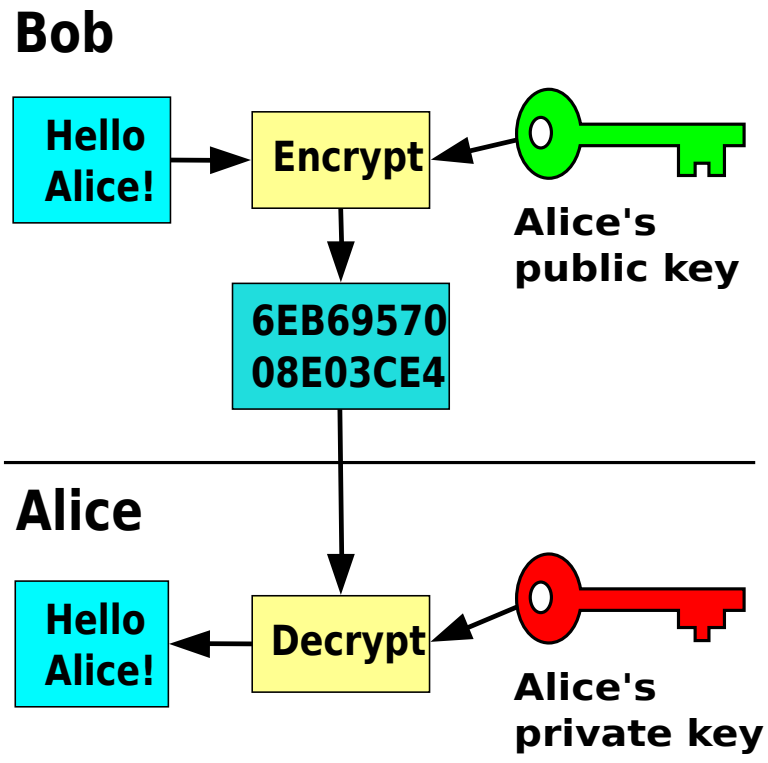




- RSA används bla i SSL.
- Bygger på att faktorisering är svårt.

Publika krypton





- RSA används bla i SSL.
- Bygger på att faktorisering är svårt.
- Om P=NP så betyder det att RSA (och alla nu kända publika krypton) enkelt kan knäckas.

Kul. Vad gör jag nu?

- Logik är studiet av formella system och giltiga argument.

Kul. Vad gör jag nu?

- Logik är studiet av formella system och giltiga argument.
- Innefattar t.ex. beräkningsbarhet och komplexitetsteori.

Kul. Vad gör jag nu?

- Logik är studiet av formella system och giltiga argument.
- Innefattar t.ex. beräkningsbarhet och komplexitetsteori.
- Kan läsas på universitet och högskolor inom ramen för matematik, datavetenskap, filosofi eller lingvistik (allmän språkvetenskap).
- Finns som eget ämne på Göteborgs universitet (unikt!).
- Masterprogrammet i logik, Master in Logic, kan läsas efter grundutbildning i t.ex. matematik eller datavetenskap.

TACK!