# Public Key encryptie: oorsprong (1)

Twee problemen aan grondslag:

1. sleutelverdeling:
   - beveiligde communicatie tussen A en B vereist ofwel
     - **gedeelde sleutel, vroeger uitgewisseld**
     - **KDC**
   - KDC druist in tegen essentie van cryptografie
     Withfield Diffie & Martin Hellman (Stanford University, 1976)
     - **volledige geheimhouding ?**
     - **KDC gecompromitteerd ?**

hoe beveiligde verbinding opzetten zonder een KDC te moeten vertrouwen met jouw sleutel ?

# Public Key encryptie: oorsprong (2)

Probleem 2:

2. digitale handtekeningen
   - equivalent van handtekening op papier
   - noodzakelijk bij groter gebruik van elektronische gegevensuitwisseling

   hoe nagaan dat een boodschap
   - intact is
   - afkomstig is van de “echte” afzender

# Public Key encryptie: principes

- conventionele cryptografie
  - gebruikt substitutie en permutatie
  - is symmetrisch
- publieke sleutel cryptografie
  - gebruikt wiskundige functies (concepten van getaltheorie)
  - is asymmetrisch
- is publieke sleutel cryptografie beter ?
  - biedt geen sterkere beveiliging (sleutellengte is bepalend)
  - weerstaat even goed aan cryptanalyse
- is conventionele encryptie overbodig ?
  - minder rekenkracht vereist
  - te implementeren in devices
- probleem van sleutelverdeling blijft bestaan

# Public Key encryptie: principes (2)

gebruikt twee sleutels

- publieke sleutel
  - door iedereen gekend
  - gebruikt voor
    - **encrypteren van boodschappen**
    - **verifiëren van handtekeningen**
- private sleutel, alleen gekend door
  - ontvanger bij decoderen van boodschap
  - zender bij zetten van handtekening

public key encryptie is asymmetrisch omdat

- wie boodschap encrypteert en handtekening verifieert
- kan NIET boodschap decrypteren of handtekening zetten

# Basis publieke sleutel algoritmen

- het is computationeel onuitvoerbaar om
  - decodeersleutel te bepalen indien alleen
    - **codeeralgoritme**
    - **encryptiesleutel**
  - gekend is
- het is computationeel gemakkelijk om
  - boodschap te coderen en te decoderen indien juiste codeersleutel resp. decodeersleutel gekend is
- elke “gebruiker” genereert sleutelpaar
  - private & publieke
  - tweelingsleutels
- elke sleutel kan gebruikt worden voor encryptie als andere gebruikt wordt voor decryptie (RSA)

# Schema's en toepassingen (1)

Schema's:

- geheimhouding
- authenticatie
- geheimhouding en authenticatie

Toepassingen:

- encryptie / decryptie
- digitale handtekening
- uitwisseling van sessiesleutels

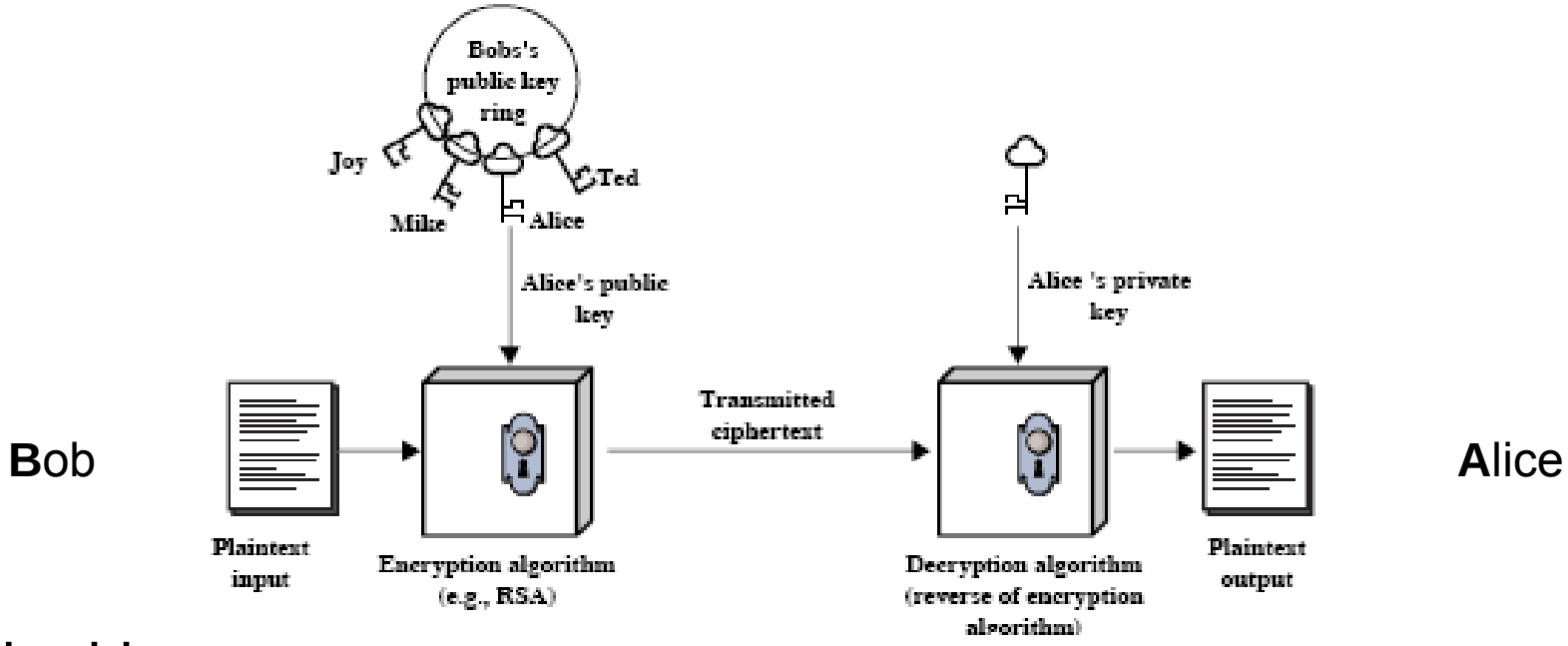
Welbepaalde algoritmen kunnen gebruikt worden voor

- ofwel alle toepassingen
- ofwel één toepassing

# Schema's en toepassingen (2)

| Algoritme | Encryptie / Decryptie | Digitale handtekening | Sleutel-uitwisseling |
|---|---|---|---|
| **RSA** | ja | ja | ja |
| **Diffie-Hellman** | nee | nee | ja |
| **DSS** | nee | ja | nee |

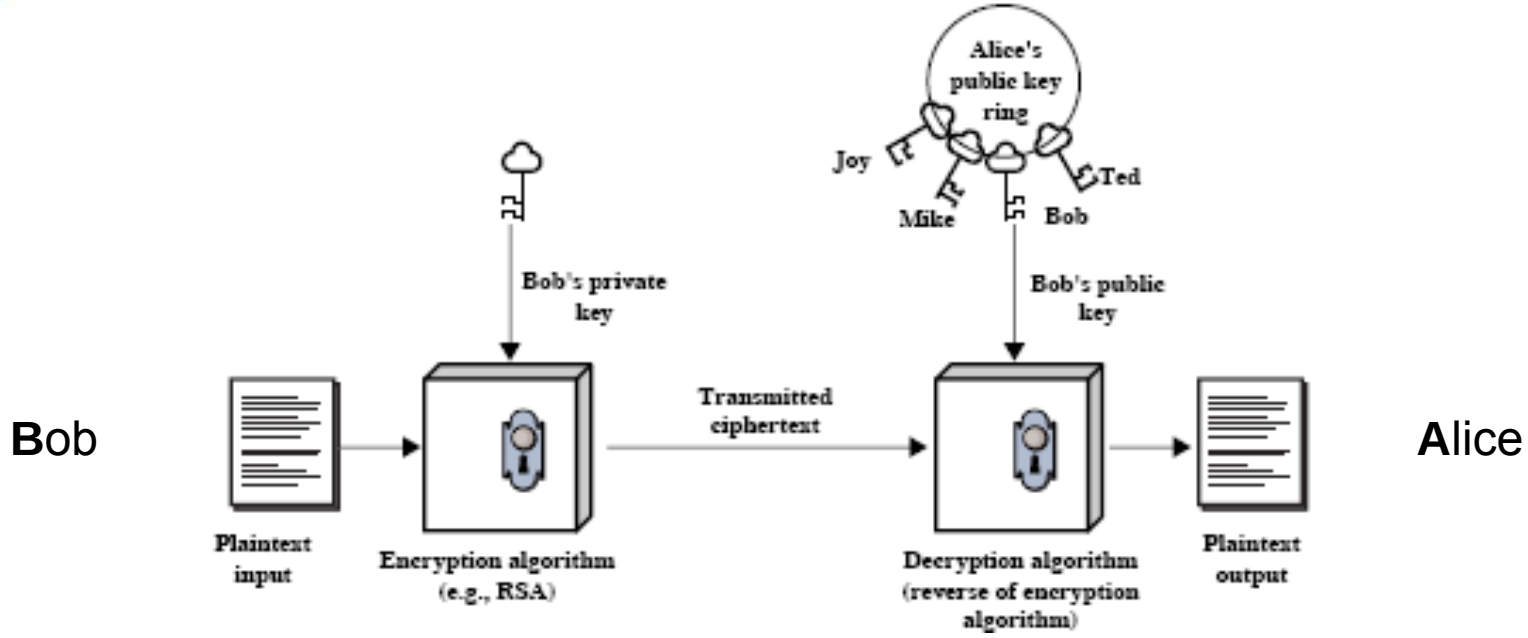
# Principe van geheimhouding




Bedenkingen:

- private sleutel moet onder controle blijven
- van wie is de publieke sleutel echt?
- op elk moment kan nieuw sleutelpaar gegenereerd worden

# Principe van authenticatie




Bedenkingen:

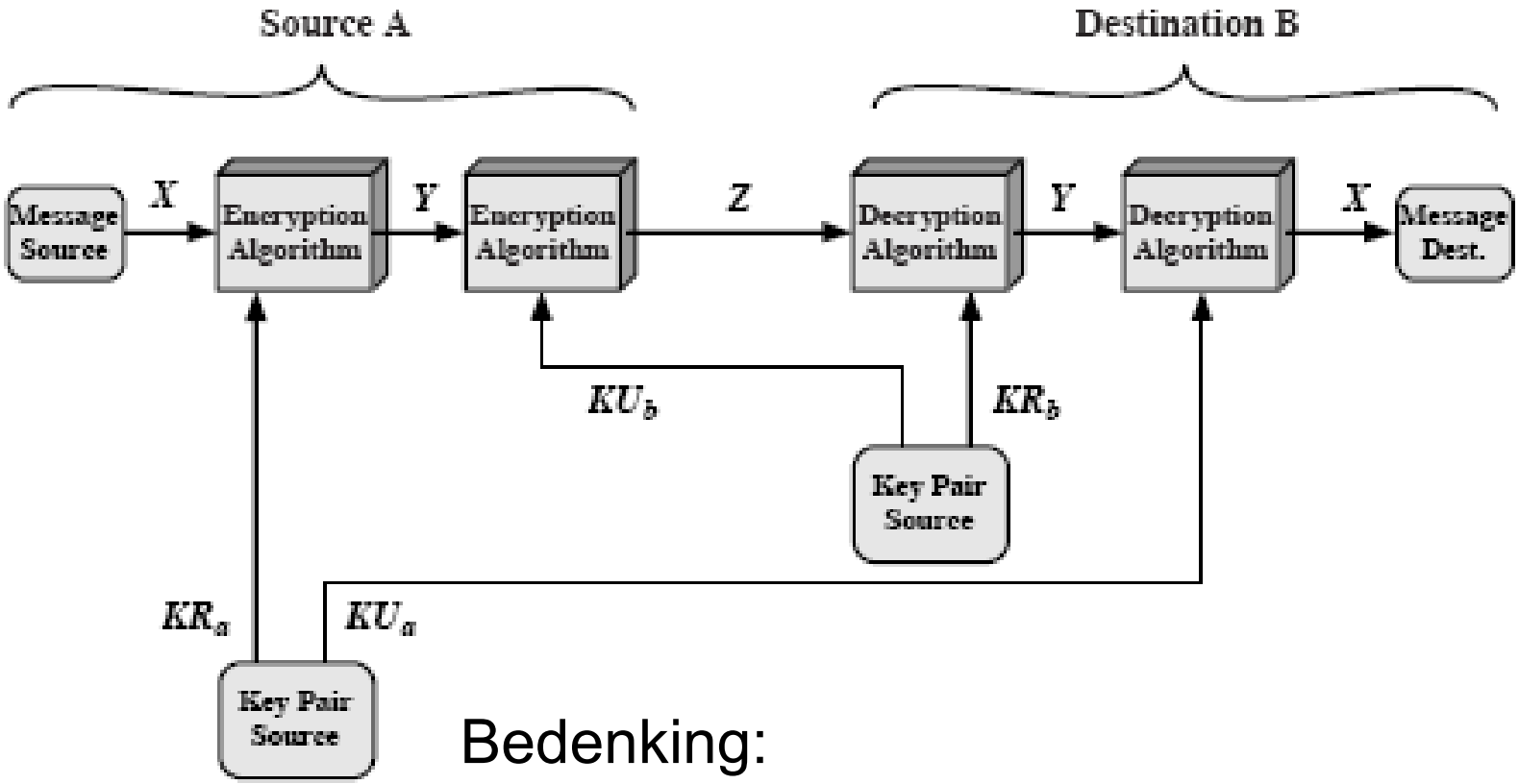
- private sleutel is alleen gekend door afzender
- van wie is de publieke sleutel echt?
- hoe kan PT output herkend worden?
- er is geen vertrouwelijkheid

# Hoe is authenticiteit bepaald?

Twee mogelijkheden voor de handtekening:

- $E_{KRB}[M_B]$
  - wijzigingen van boodschap M kan alleen met kennis van $K_{RB}$
  - M is geauthentiseerd
    - **afkomstig van bron B**
    - **integriteit**
  - bewaren van zowel $M_B$ als $E_{KRB}[M_B]$ voor betwistingen
  - is bekomen M de juiste?
- $E_{KRB}[f(M_B)]$
  - alleen authenticator wordt geëncrypteerd
  - authenticator is beperkt in omvang
  - wijzigen van M wijzigt f(M)
  - digitale handtekening

# Geheimhouding + authentisering




Bedenking:

algoritme moet 4X worden uitgevoerd

# RSA algoritme

- Ronald Rivest, Adi Shamir, Leonard Adleman (MIT, 1977)
- implementeren ideeën van Diffie en Hellmann
- best gekend en meest gebruikt algoritme
- gebaseerd op machtsverheffing in een eindig Galois veld over gehele getallen modulo een priemgetal
- gebruikt grote integers (bv. 1024 bits, 309 decimale cijfers)
- beveiliging op basis van moeilijke ontbinding in priemfactoren

# Totiëntfunctie van Euler

Euler is Zwitsers wiskundige (1707-1783)

$\Phi(n)$ =

- aantal gehele en positieve getallen
  - kleiner dan n
  - *relatief priem* met dat getal (geen gemene delers, bv. 8 en 9)
- $\Phi$(priem) = priem-1
- $\Phi(9) = 6$
  9 is relatief priem met 8,7,5,4,2,1
- $\Phi(5) = 4$
  5 is relatief priem met 4,3,2,1

# Eigenschap 1 (Euler)

voor elk geheel getal i relatief priem met gehele n (*ggd(i,n)=1*) geldt

$$i^{\Phi(n)} \bmod n = 1$$

Voorbeeld:

n=9 , Φ(9)=6, i=2

$2^6 \bmod 9 = 1$ (63+1)

# Eigenschap 2 (Euler)

voor elke twee gehele getallen e en d waarvoor geldt dat

$ed = 1 \bmod \Phi(n)$ of

*e* en *d* zijn mekaars inverse als

*e* en *d* relatief priem met $\Phi(n)$

eigenschap: voor m geheel met $0<m<n$ geldt er dat

$(m^e)^d \bmod n = m$

en

$(m^d)^e \bmod n = m$

# Eigenschap 2

$(m^e)^d \bmod n = m = (m^d)^e \bmod n$

cryptografisch bekeken:

m = message, c = ciphertext

$c = m^e \bmod n$ $\qquad\qquad$ $m = c^d \bmod n$

Numeriek:

n=9 $\quad \Phi(9)=6$

e=5 $\quad$ d=11 $\qquad\qquad$ 5x11= 55 mod 6 = 1

m=2 (0<m<9=n)

$c=2^5 \bmod 9 = 5$ $\qquad\qquad$ $c=5^{11} \bmod 9 = 2$

$(2^5)^{11} = (2^{11})^5 = 36028797018963968$

mod 9 = 2

# RSA sleutelbepaling

- kies twee zeer grote priemgetallen ( p en q ), gelijke omvang, meerdere honderden cijfers, geheim
- bereken systeem modulus:  $n = p \times q$
- bereken $\Phi(n)$:  $\Phi(n) = (p-1)(q-1)$
- kies waarde voor encryptiesleutel e
  - $ggd(e,\Phi(n))=1$
  - $\max(p+1, q+1) < e < \Phi(n)$
- bereken decryptiesleutel d : er geldt
  - $ed = 1 \bmod \Phi(n)$
  - $0 \leq d \leq N$
- publiceer publieke sleutel  KU = {e,n}
- houd private sleutel geheim  KR = {d,p,q}

# RSA gebruik (A zendt naar B)

- boodschap M < n (opdelen in blokken indien nodig)
- A encrypteert boodschap M
  - verkrijgt publieke sleutel van B: $K_U$={e,n}
  - berekent C = $M^e$ mod n
- B decrypteert boodschap M
  - gebruikt zijn private sleutel: $K_R$={d,p,q}
  - berekent  M = $C^d$ mod n

# RSA: numeriek voorbeeld

- kies twee priemgetallen, p=7 en q=17
  bereken n=pxq=7x17=119
- bereken Φ(n)=(p-1)(q-1)=6x16=96
- kies een waarde voor e = 5
  - ggd(e,Φ(n))=ggd(e,96)=1
  - e<Φ(n)  [ eigenlijk: max(8,18)<e<96 ; voldaan?]
- Bepaal d zo dat
  - d.e = 1 mod 96 en d<96
  - correcte waarde voor d = 77
  - want 77x5=385=4x96+1
- Sleutels
  - KU={5,119}
  - KR={77,119}

# RSA: numeriek voorbeeld (2)

- voorbeeld plaintext  m=19
- berekening ciphertext c=66, want
  - $19^5 = 2476099$ ➔ mod 119 = 66
- decodering (berekening van plaintext)
  - $66^{77} = 1{,}27316015002712725024996823827 45e+140$
  - ➔ mod 119 = 19

# Sterkte van RSA

- publiek: n en e of d
- code breken =
  - berekenen van Φ(n)
  - n ontbinden in priemfactoren
  - kan meerdere honderden jaren duren
- sleutelomvang van 1024 of 2048 bits lijkt redelijke keuze
- Uit onderzoek blijkt volgende noodwendigheden:
  - p wijkt slechts enkele cijfers af van q
  - $10^{75} < p, q < 10^{100}$
  - p-1 en q-1 bevatten een groot priemgetal
  - ggd ( p-1, q-1 ) is klein

# Sleutelbeheer

Twee aspecten:

- distributie van publieke sleutels
  - publieke bekendmaking
  - publieke directory
  - publieke sleutelautoriteit
  - publieke sleutelcertificaten
- gebruik van publieke sleutel cryptografie voor verspreiding van geheime (sessie)sleutels voor conventionele encryptie

# Publieke bekendmaking

- verzending via mail
- forum, mailing list, usenet, ...
- nadelen:
  - iedereen kan zich voordoen als A en sleutel publiek bekend maken
  - vervalser kan gecodeerde boodschappen bestemd voor A onderscheppen en lezen
  - vervalser kan zijn berichten doen lijken alsof afkomstig van A

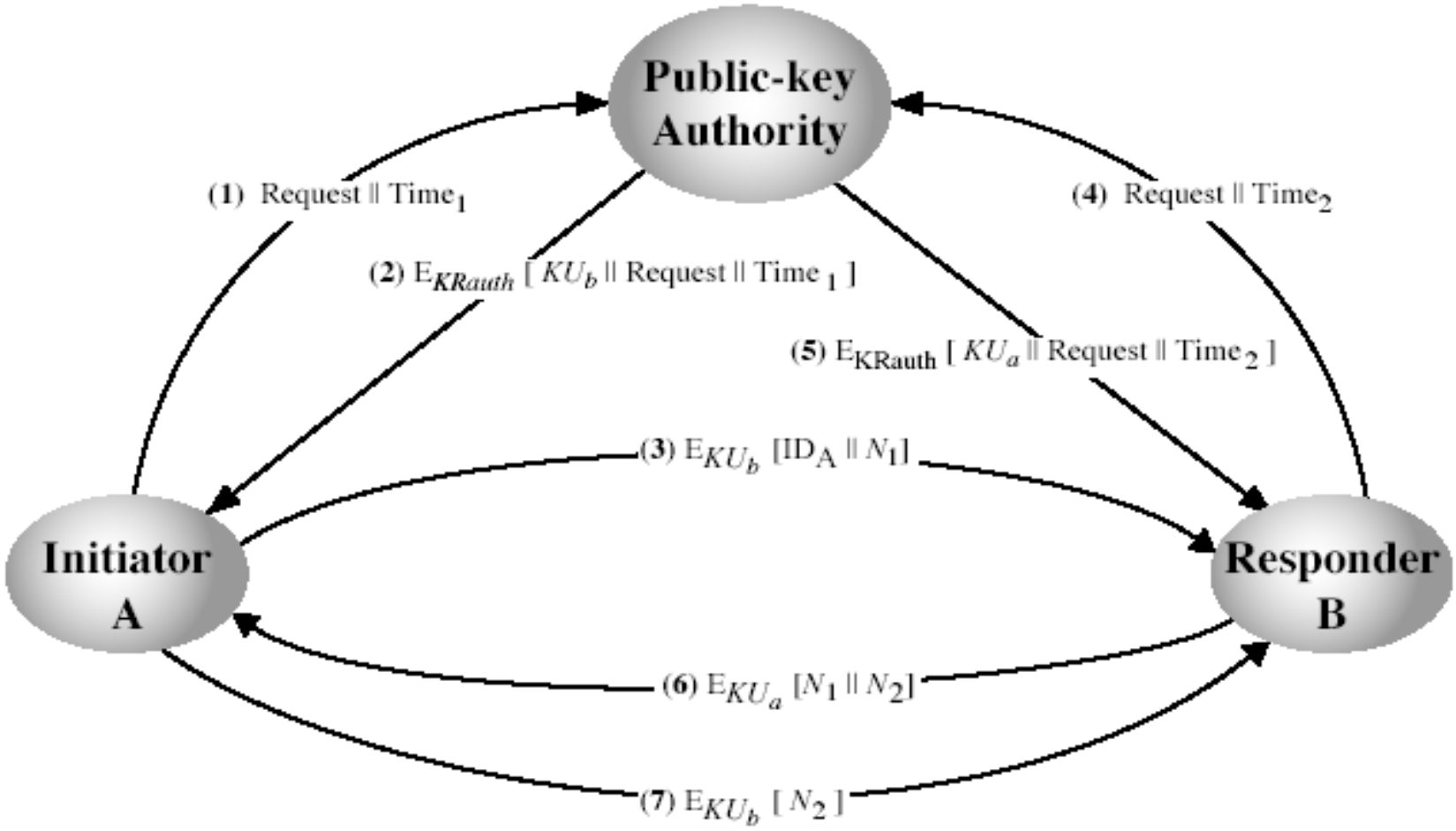
# Publieke directory

- beheerd door vertrouwde organisatie
- registratie van {naam, publieke sleutel}
- persoonlijk of via geauthentiseerde verbinding
- sleutel kan vervangen worden als gecompromitteerd of indien reeds te veel gebruikt
- directory kan elektronisch geraadpleegd worden
- nadeel:
  - sleutel directorybeheerder kan gekraakt worden
  - zelfde gevolgen als hiervoor

# Publieke sleutelautoriteit

- heeft de eigenschappen van een directory
- verbeterde beveiliging door sterkere controle over de distributie van de sleutels in de directory via scenario
- vereist dat gebruikers publieke sleutel van directory kennen
- gebruikers interageren met directory om gewenste publieke sleutel te bekomen
- vereist real-time toegang tot directory
- nadeel: autoriteit kan flessenhals zijn

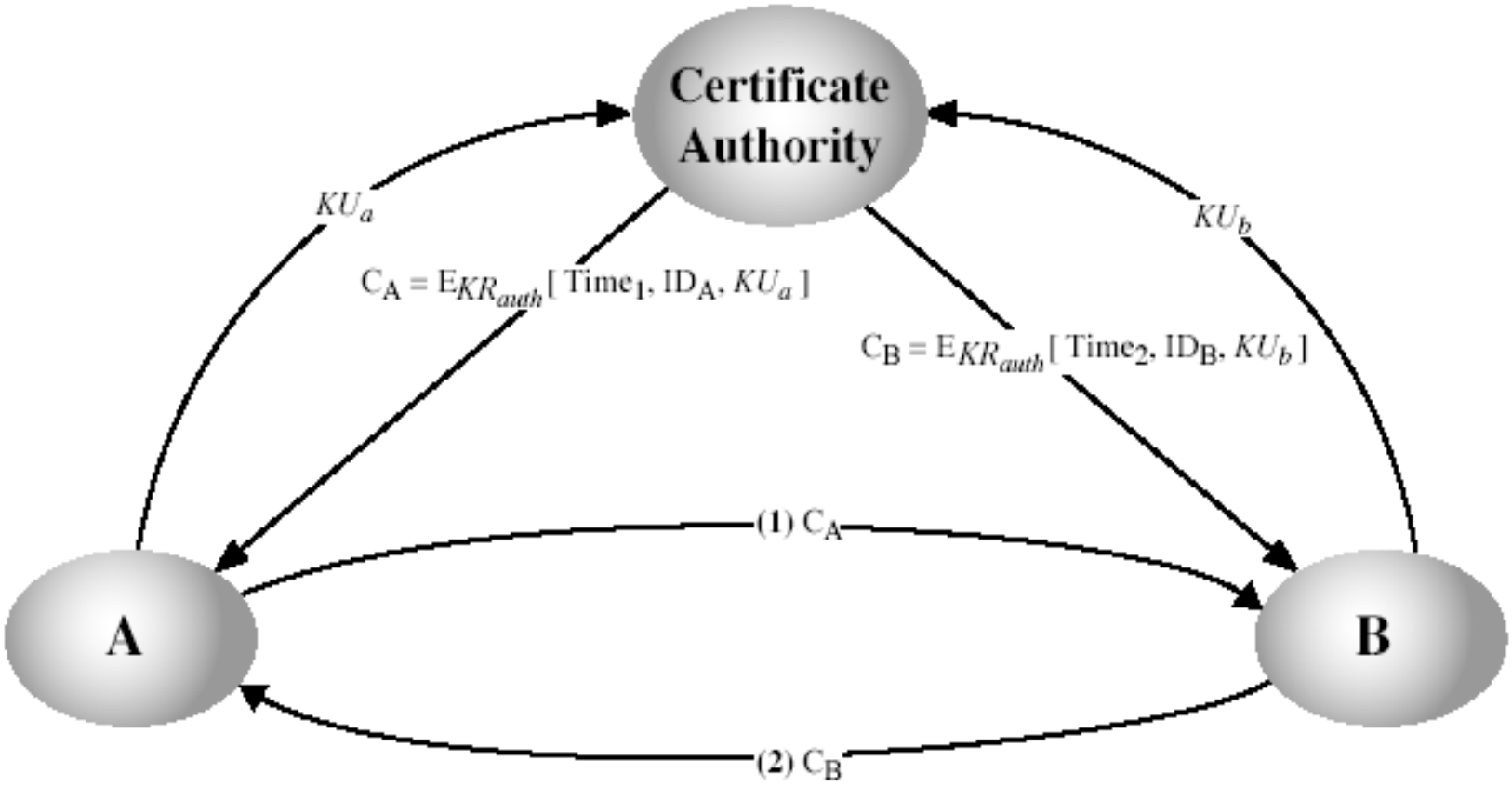
# Public Key Authority



# Publieke-sleutelcertificaten

- certificaten laten sleuteluitwisseling toe zonder tussenkomst van de autoriteit
- certificaat
  - verbindt **publieke sleutel** met **identiteit**
  - bevat nog andere informatie (geldigheidsperiode, gebruiksrecht, …)
- volledig certificaat is getekend door autoriteit (CA)
- wordt afgeleverd door autoriteit
- kan nagekeken worden door iedereen die publieke sleutel van CA kent

# Publieke-sleutelcertificaten



# Sleutelbeheer

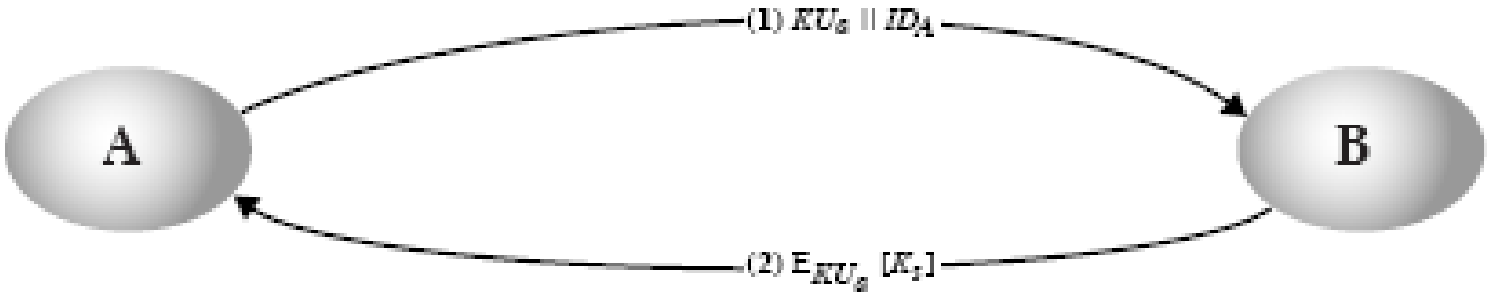
Twee aspecten:

- distributie van publieke sleutels
  - publieke bekendmaking
  - publieke directory
  - publieke sleutelautoriteit
  - publieke sleutelcertificaten
- gebruik van publieke sleutel cryptografie voor verspreiding van geheime (sessie)sleutels voor conventionele encryptie

# Distributie van sessiesleutels

- publieke sleutelalgoritmen zijn te traag om grote hoeveelheden plaintext te coderen
- nood aan sessiesleutels
- mogelijke manieren:
  - eenvoudige verdeling van geheime sleutels
  - geheime sleutelverdeling met beveiliging en authentisering
  - Diffie-Hellman sleuteluitwisseling

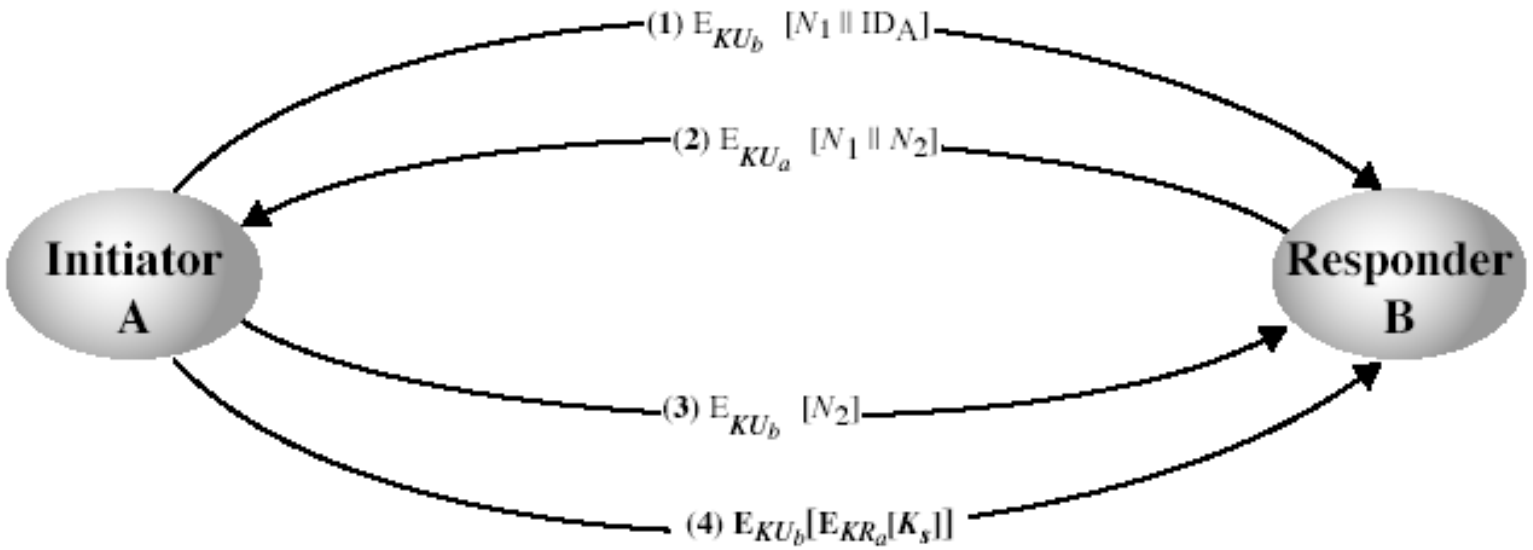
# Eenvoudige verdeling van sessiesleutels




- A genereert public key set en vraagt B om sessiesleutel
- B genereert KS en stuurt naar A
- A verwijdert public key set (zeer attractief, want alle sleutels verdwijnen)
- Gevaren:
  - boodschap 1 kan onderschept worden door C
  - C kan zich voordoen als A

# Verdeling sessiesleutel met beveiliging en authentisering

A en B hebben mekaars publieke sleutel vroeger reeds uitgewisseld



# Diffie-Hellman Key Exchange

- eerste uitwisselingsschema op basis van een publieke sleutel
- voorgesteld door Diffie & Hellman (1976), samen met hun ideeën over publieke sleutelcryptografie
- methode om een geheime sessiesleutel publiek uit te wisselen
- gebruikt in verschillende commerciële producten

# Diffie-Hellman Key Exchange (2)

- distributieschema op basis van publieke sleutel
  - kan niet gebruikt worden om een willekeurige boodschap uit te wisselen
  - dient om een gemeenschappelijke geheime sleutel te bepalen
  - is alleen gekend door de twee deelnemers
- waarde van de sleutel afhankelijk
  - van private en publieke sleutelinformatie van deelnemers
- gebaseerd op de machtsverheffing in een eindig Galois veld (modulo priemgetal) → eenvoudig
- veilig?
  - berekening van discrete logaritme is moeilijk (vergelijkbaar met ontbinding in priemfactoren)

# Primitieve wortel van priemgetal

- *a* is primitieve wortel van priemgetal *p*
  - als alle machten van a alle getallen genereren van 1..p-1
- formeel:
  - voor elk geheel getal i waarvoor geldt:
    0 < i <= p-1
  - $a^i \bmod p = b$
  - voor elke i zijn alle waarden b onderscheiden gelijk aan
    1 .. (p-1)

- 7 is primitieve wortel van 11

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 49 | 343 | 2401 | 16807 | 117649 | 823543 | 5764801 | 40353607 | 282475249 |
| 7 | 5 | 2 | 3 | 10 | 4 | 6 | 9 | 8 | 1 |

# Discrete logaritme

- *a* is primitieve wortel van priemgetal *p*
  - voor elke geheel getal i waarvoor geldt 0 < i <= p-1
  - $a^i$ mod p = b
  - voor elke i zijn alle waarden b onderscheiden gelijk aan 1 .. (p-1)

- voor elk geheel getal *b* en een welbepaalde waarde *a* die primitieve wortel is van priemgetal *p*
  - bestaat er een unieke waarde *i*
  - zo dat $a^i$ mod p = b

  ➔ *i* is discrete logaritme van *b* in basis *a* modulo p

# Diffie-Hellman voorbereiding

- A en B gaan akkoord over globale parameters:
  - **p** groot priemgetal (11)
  - **a** primitieve wortel van p (7)
- A
  - kiest zijn geheim getal $x_A < p$ (3)
  - berekent zijn publieke sleutel $y_A = a^{x_A} \bmod p$ ( $7^3 \bmod 11 = 2$)
- B
  - kiest zijn geheim getal $x_B < p$ (6)
  - berekent zijn publieke sleutel $y_B = a^{x_B} \bmod p$ ( $7^6 \bmod 11 = 4$)
- A en B wisselen publieke sleutel uit:
  - B ➔ A: $y_B$ (4)
  - A ➔ B: $y_A$ (2)

# Diffie-Hellman sleutelbepaling

- **p** (11) en **a** (7) gekend
- A: $x_A$ (3), $y_A = a^{x_A} \bmod p$ ($7^3 \bmod 11 = 2$), $y_B$ (4)
- B: $x_B$ (6), $y_B = a^{x_B} \bmod p$ ($7^6 \bmod 11 = 4$), $y_A$ (2)

- gemeenschappelijke sessiesleutel voor A & B is $K_{AB}$:

  $K_{AB} = a^{x_A . x_B} \bmod p$

  $= y_A^{x_B} \bmod p$ (kan B berekenen: $2^6 \bmod 11 = 9$)

  $= y_B^{x_A} \bmod p$ (kan A berekenen): $4^3 \bmod 11 = 9$)

  $= y_B^{x_A} \bmod p = (a^{x_B} \bmod p)^{x_A} \bmod p$

  $= (a^{x_B})^{x_A} \bmod p = a^{x_B x_A} \bmod p$

  $= (a^{x_A} \bmod p)^{x_B} \bmod p$

  $= y_A^{x_B} \bmod p$

# Diffie-Hellman sterkte

- **p** (11), **a** (7), **$y_A$** (2) en **$y_B$** (4) gekend
  ook door eventuele tegenstander
- A: $x_A$ (3) privaat
- B: $x_B$ (6) privaat
- gemeenschappelijke sessiesleutel voor A & B is $K_{AB}$:
  $K_{AB} = y_A^{x_B} \bmod p = y_B^{x_A} \bmod p$

- tegenstander moet discrete logaritme i = $X_B$ berekenen zo dat
  - $y_B = a^{x_B} \bmod p$
  - is ondoenbaar

# Diffie-Hellman kleurenanalogie

| | A | | B |
|---|---|---|---|
| publiek uitgewisseld | geel | ←→ | geel |
| private kleur bijgevoegd | paars | | rood |
| mengsel uitgewisseld | geel + paars | → ← | geel +rood |
| sleutel=ontvangen mengsel+ private kleur | geel+rood + paars | | geel+paars + rood |

# Diffie-Hellman gebruik

- $K_S$ uitgewisselde sessiesleutel via Diffie-Hellman
- kan op 2 manieren gebruikt worden:
  - als sleutel van conventioneel algoritme zoals DES of AES
    - **CT=Encr($K_S$, PT)**

      **of omdat $K_S$ te groot is:**
    - **CT=Encr( hash($K_S$), PT)**
  - als sleutel bij ElGamal public key encryptie
    - **ElGamal is encryptiemethode die ideeën van DH implementeert**

# ElGamal

- Sleutelgeneratie door Alice
  - kiest $p$ en $a$ en $x_A$
  - berekent $y_A = a^{x_A} \bmod p$
  - publieke sleutel $K_{UA}$: $p$, $a$, $y_A$
  - private sleutel $K_{RA}$: $p$, $a$, $x_A$
- Encryptie door Bob
  - versleuteling voor Alice
  - gebruikt $K_{UA}$
  - kiest $x_B$
  - berekent $y_B = a^{x_B} \bmod p$ en $K_S = a^{x_A x_B} \bmod p$
  - encrypteert: $C = M \,.\, K_S \bmod p$
  - verzendt naar Alice: $y_B$ en $C$

# ElGamal: decryptie

- private sleutel van Alice $K_{RA}$: $p, a, x_A$
- encryptie door Bob
  - encrypteert: $C = M.K_S \bmod p$
  - verzendt naar Alice: $y_B$ en $C$
- decryptie door Alice
  - berekent $e = p-1-x_A$
  - berekent $y_B^e \,.\, C \bmod p = M$
  - bewijs:
    $y_B^e.C \bmod p = a^{x_B e}.\, C \bmod p =$

    $a^{x_B(p-1-x_A)}.M.K_S \bmod p = a^{x_B(p-1)}\,(a^{x_B})^{-x_A}.M.a^{x_A.x_B} \bmod p =$

    $a^{(p-1)x_B)}.\, M = M$ wegens stelling van Euler

# ElGamal efficiëntie

- decryptie: 1 machtsverheffing
  - vergelijkbaar met RSA
- encryptie: 2 machtsverheffingen
  - berekent $y_B = a^{x_B} \bmod p$ en $K_S = a^{x_A x_B} \bmod p$
  - RSA vereist er slechts 1, evenwel bij elke blokencryptie
  - lengte van p vergelijkbaar met p en q bij RSA
  - 1x vooraf berekenen $y_B$ en $K_S$
    - **slechts 1x vermenigvuldigen**
    - **efficiënter dan RSA**
    - **kan op smartcard opgeslagen worden**