

Asymmetrische encryptie

Cyberboswachters

Tim Dams

Deze cursus werd gemaakt met hulp van AI-assistentie. Sommige afbeeldingen zijn door AI gegenereerd.



Het sleutelprobleem

Per **eindpunt** een aparte sleutel.

Aantal sleutels: $n \times \frac{(n-1)}{2}$

15

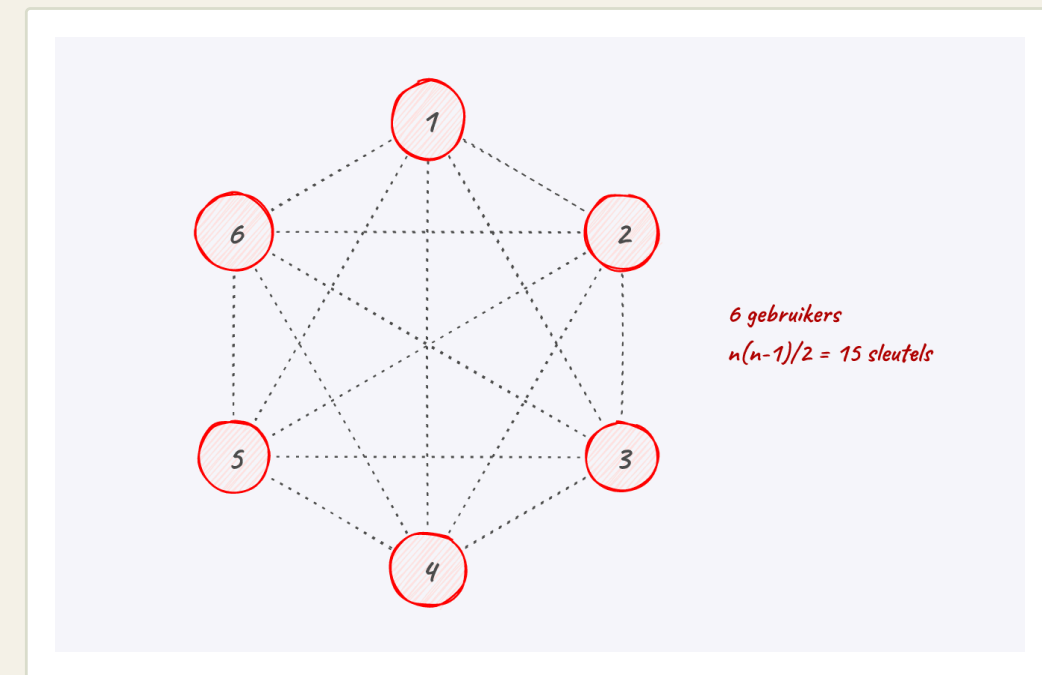
sleutels voor **6**
gebruikers

45

voor **10** gebruikers

4950

voor **100**
gebruikers



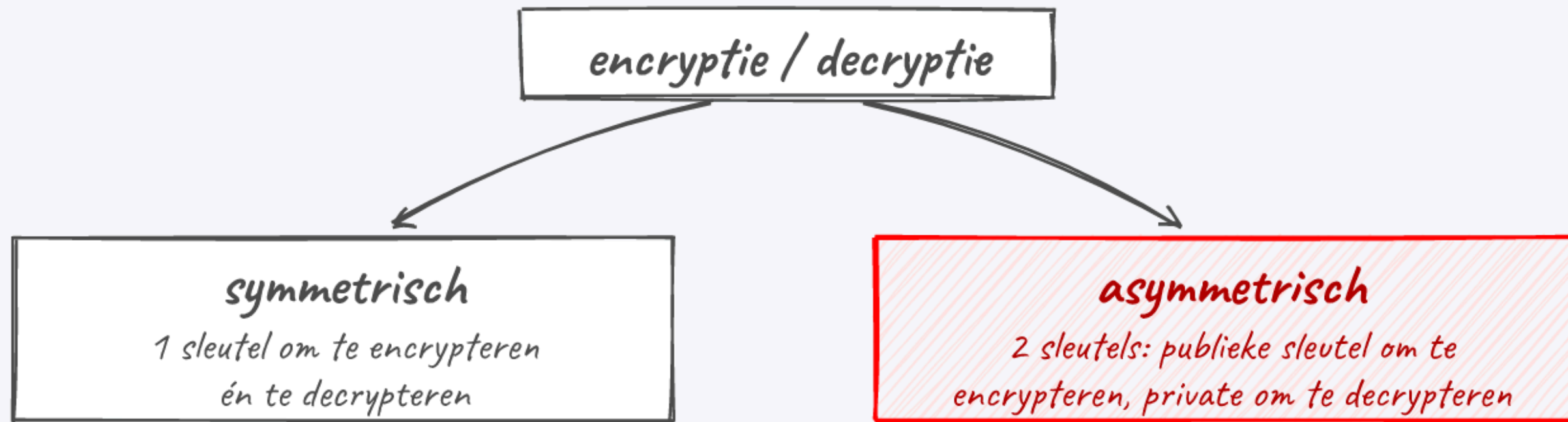
Bij 6 gebruikers zijn er al 15 sleutels nodig.

KEY DISTRIBUTION PROBLEM

Op de schaal van het Internet wordt **sleutelmanagement** onmogelijk.

Er is een andere oplossing nodig: **asymmetrische encryptie**.

Asymmetrische encryptie



Twée sleutels



Publieke sleutel

Iedereen mag hem gebruiken om berichten naar de eigenaar te **encrypteren**.

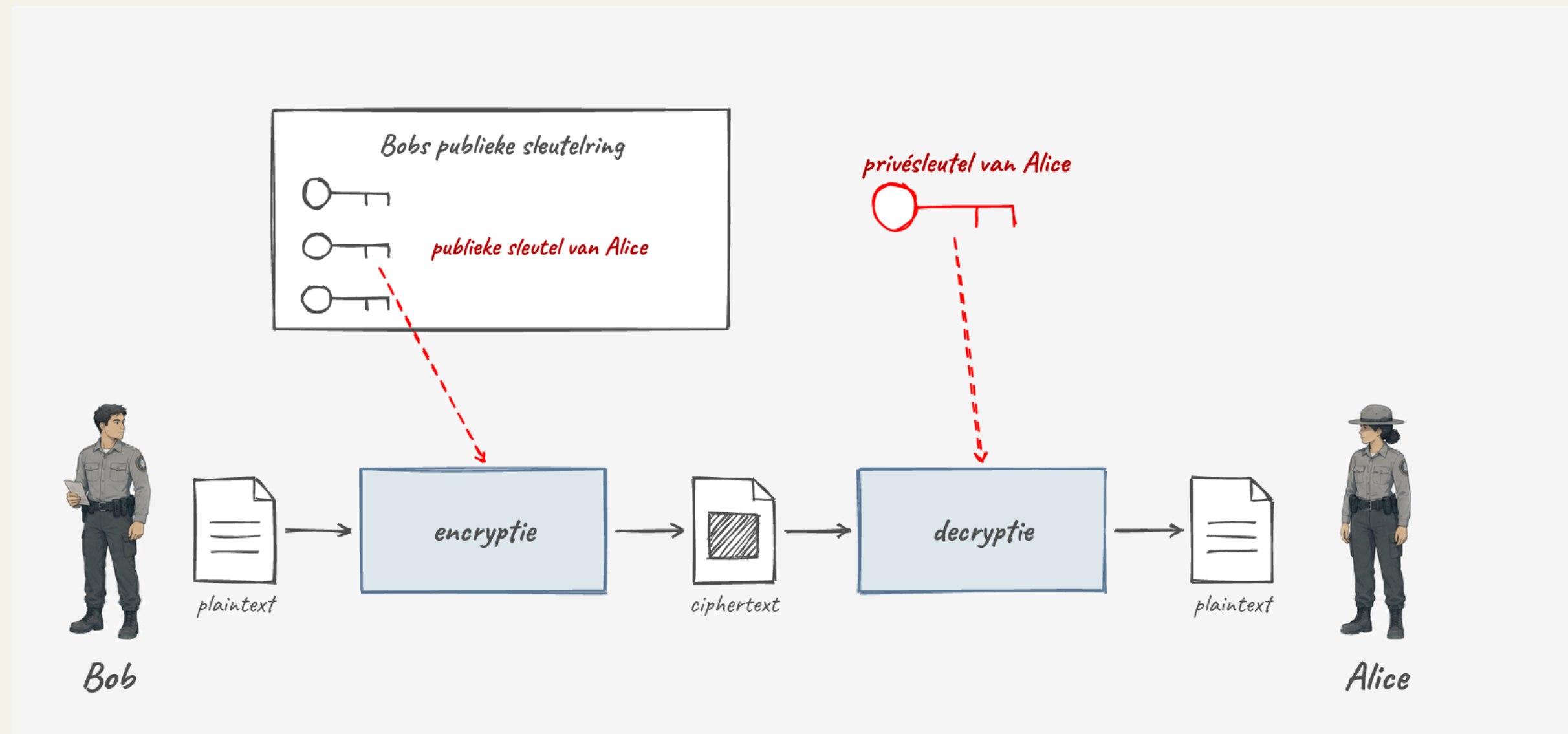


Private sleutel

Enkel de eigenaar kan ermee **decrypteren**.
Moet altijd **geheim** blijven (cfr. Kerckhoffs).

Asymmetrische encryptie: werking

Enorme impact op beveiligde communicatie via het Internet.



Publieke crypto: Bob gebruikt de publieke sleutel van Alice om haar een beveiligd bericht te sturen.

De kist-metafoor

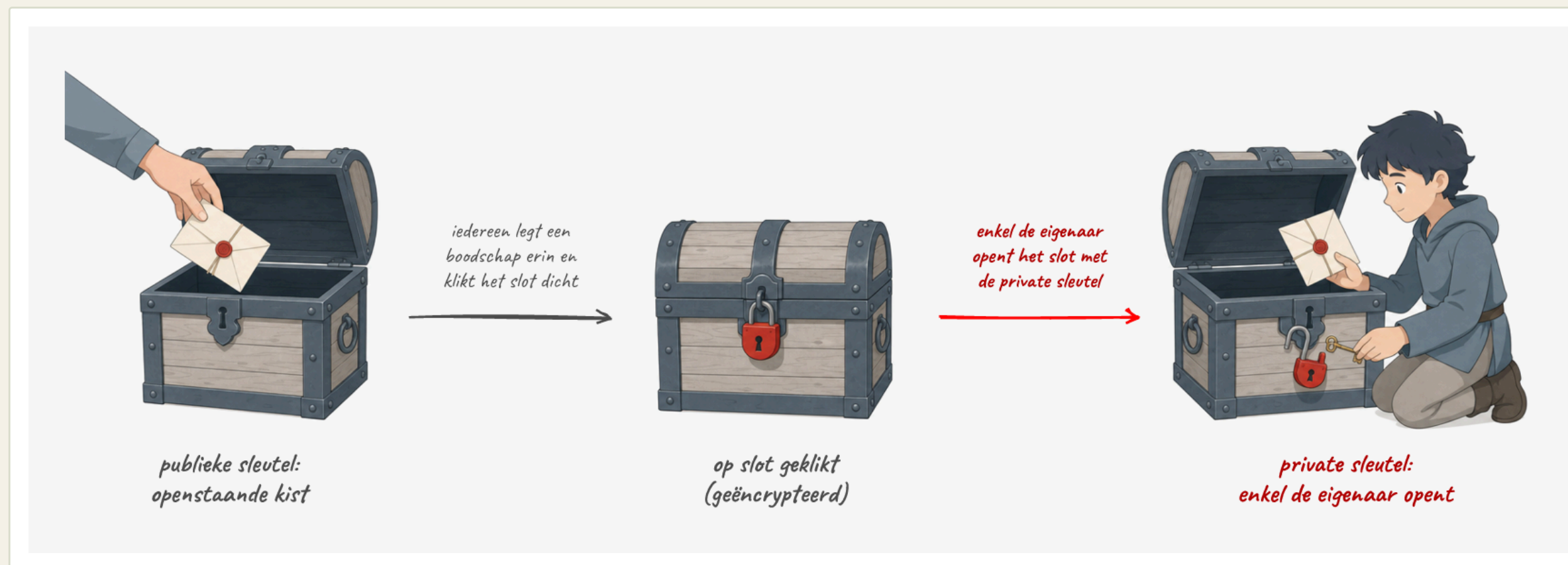


Tip

De **publieke sleutel** = een openstaande kist.

Iedereen kan er een boodschap in leggen en op slot klikken (*encrypteren*).

Enkel de eigenaar heeft de **private sleutel** om de kist te openen (*decrypteren*).



Drie toepassingen van publieke crypto



Encryptie

Versleutelen met de **publieke** sleutel van de ontvanger.

Bv. RSA.



Sleuteluitwisseling

Over een onveilig kanaal een symmetrische sleutel afspreken.

Bv. Diffie-Hellman.

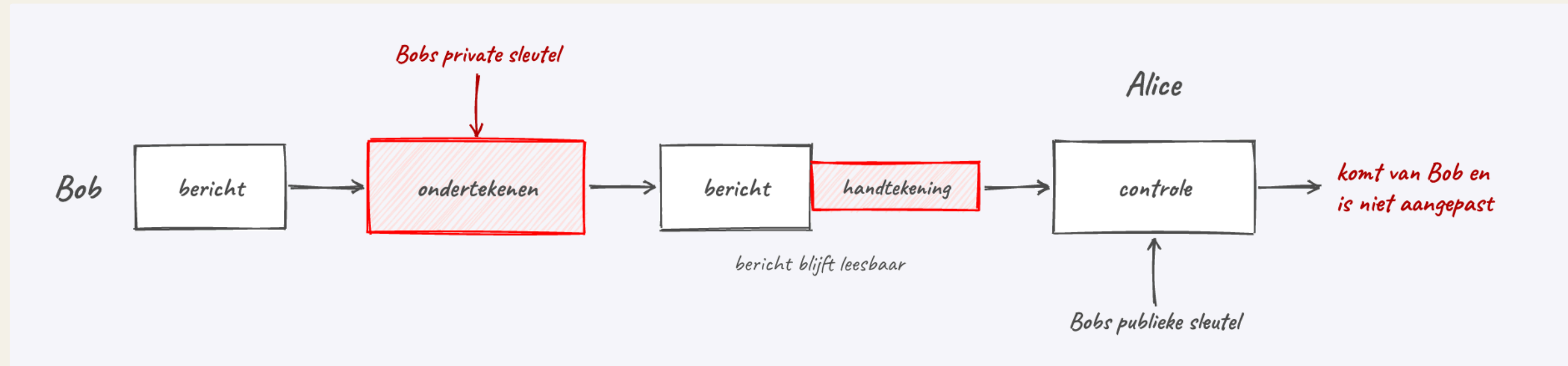


Digitale handtekening

Tekenen met je **private** sleutel.

Bewijs van identiteit.

Digitale handtekening: concept



Het ondertekenen van een document met behulp van je private sleutel.

Enkel de eigenaar heeft de private sleutel: een geldige handtekening bewijst zijn identiteit (**authenticatie**), toont dat het bericht niet werd aangepast (**integriteit**) en hij kan achteraf niet ontkennen dat hij tekende (**non-repudiation**).

SLEUTELUITWISSELING

Diffie-Hellman

WAAROM SLEUTELUITWISSELING?

Symmetrische crypto is **sneller**, maar hoe spreek je die sleutel veilig af?

Bij grote groepen loopt het sleutelbeheer vast. **Diffie-Hellman** laat twee partijen een gedeeld geheim (**shared secret**) afspreken over een **onveilig kanaal**.

Diffie-Hellman: werkwijze



Deze waarden zijn **geen sleutelpaar** zoals bij RSA: in de praktijk (bv. HTTPS) worden ze na de uitwisseling meestal weggegooid.

Diffie-Hellman: rekenvoorbeeld

Publiek afgesproken: basis $g = 7$ priemgetal $p = 11$

Stap	Alice	Bob
1	Kiest geheim getal $A = 3$	Kiest geheim getal $B = 6$
2	Berekent $7^A \% 11 = 343 \% 11 = 2 (= X)$	Berekent $7^B \% 11 = 117649 \% 11 = 4 (= Y)$
3	Stuurt $X=2$ naar Bob	Stuurt $Y=4$ naar Alice
4	Berekent $Y^A \% 11 = 4^3 \% 11 = 9$	Berekent $X^B \% 11 = 2^6 \% 11 = 9$

9

het **shared secret**: enkel Alice en Bob kennen dit getal

WAAROM WERKT DIT?

Dankzij de eigenschappen van de **modulo-operator**.

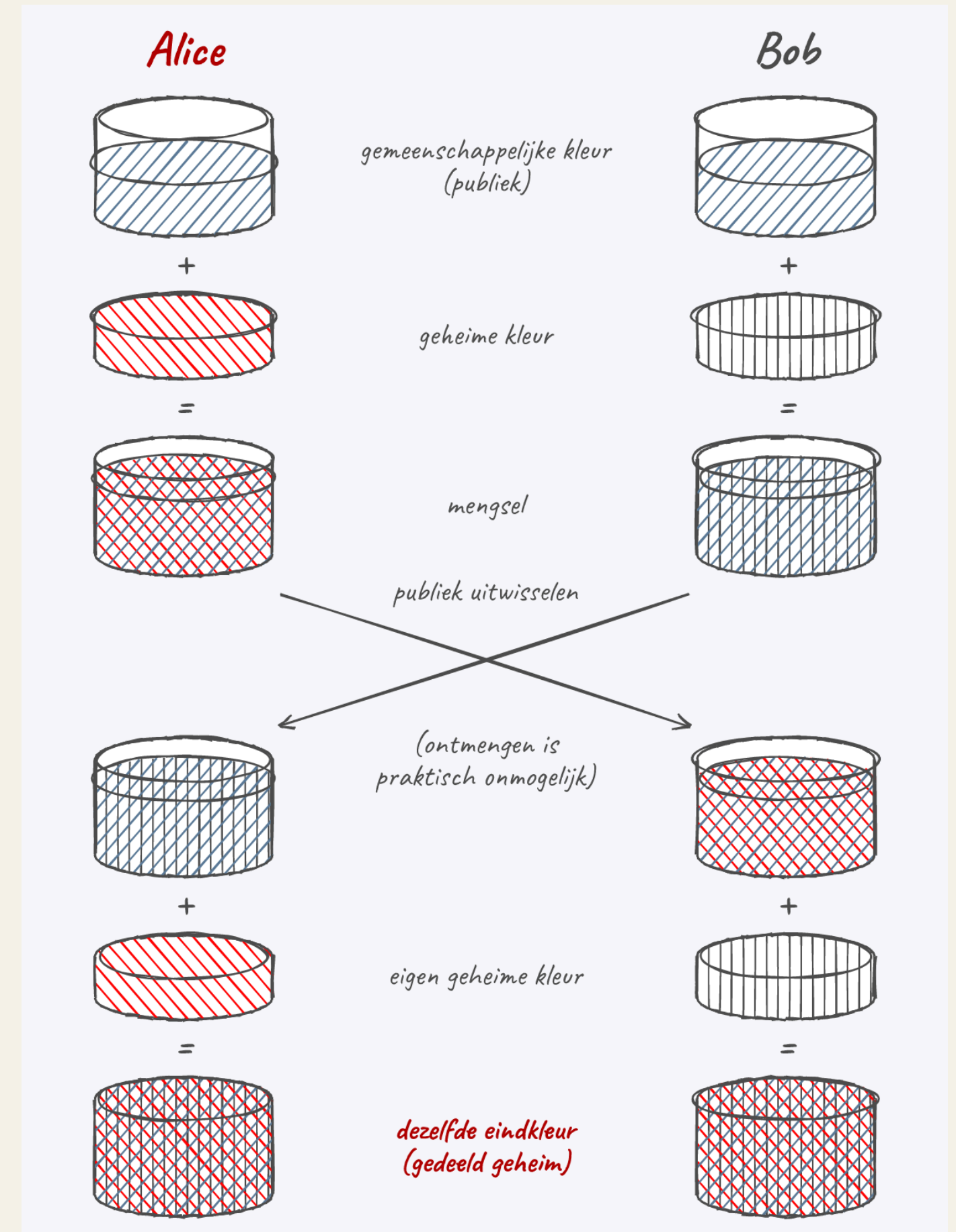
$g^a \bmod p$ uitrekenen is razendsnel. Omgekeerd uit X de geheime a terugvinden is voor grote getallen praktisch **onmogelijk**: het **discrete logaritme-probleem**.

In de praktijk kiezen Alice en Bob **véél** grotere getallen dan 3 en 6.

Verf mengen

1. **Publiek:** een gemeenschappelijke startkleur
2. Elk kiest een **geheime** kleur
3. Mengen en naar elkaar sturen
4. Eigen geheime kleur toevoegen
5. Beiden krijgen **dezelfde eindkleur**

Mengen is makkelijk, *ontmengen* praktisch onmogelijk. Digitaal speelt de **modulo-berekening** die rol.



Diffie-Hellman uitgelegd in filmpje



Alles samen uitgelegd

ASYMMETRISCHE ENCRYPTIE

RSA en ECC

RSA

1977

ontwikkeld door **R**ivest, **S**hamir en **A**dleman

1536-4096

bits sleutellengte

Relatief **traag**: vaak enkel gebruikt voor de sleuteluitwisseling, daarna schakel je over op een sneller symmetrisch cipher.

Gebaseerd op de **modulo-operator**, net als Diffie-Hellman.

RSA: sleutels aanmaken

1

Kies twee
priemgetallen:
 $p = 17$ en $q = 11$



2

Bereken
 $N = p \times q = 187$



3

Kies $e = 7$: geen
gemeenschappelijke
deler met
 $(p - 1)(q - 1) = 160$
(dus $e = 5$ zou níet
werken)



4

Zoek d zodat
 $(7 \times d) \% 160 = 1$
 $\rightarrow d = 23$

Algemeen: $(e \times d) \% ((p - 1) \times (q - 1)) = 1$

Publieke sleutel

$N = 187$ en $e = 7$ (openbaar)

Private sleutel

$d = 23$ (geheim!)

RSA: encryptie voorbeeld

Alice wil het ASCII-karakter **X** (waarde 88) naar Bob sturen.



Encryptie

Alice gebruikt de **publieke** sleutel:

$$C = data^e \% N$$

$$C = 88^7 \% 187 = 11$$



Decryptie

Bob gebruikt zijn **private** sleutel: $P = C^d \% N$

$$P = 11^{23} \% 187 = 88$$

= ASCII van X: het bericht is correct ontvangen!

RSA: waarom is dit veilig?

Vermenigvuldigen

$$113 \times 137 = 15481$$

Makkelijk.

vs

Factoriseren

$$15481 = ? \times ?$$

Computationeel **veel moeilijker**.

RSA EN CRYPTOCURRENCIES

Wie je **private sleutel** heeft, kan in jouw naam een transactie ondertekenen en zo je coins stelen.

Cryptocoins gebruiken dezelfde public crypto-concepten: je private sleutel is het bewijs dat de coins van jou zijn. Geef hem **NOOIT** aan derden.

Elliptic Curve Cryptografie (ECC)

Modernere asymmetrische crypto, gebaseerd op **elliptische krommen**. Veel kleinere sleutels voor hetzelfde beveiligingsniveau:

256 bits

ECC is even sterk als **3072 bits** RSA

384 bits

ECC is even sterk als **7680 bits** RSA

Kleiner = **sneller**, minder data, minder energie.

Ideaal voor mobiel IoT

ECC: toepassingen



ECDH

Diffie-Hellman
sleuteluitwisseling via
elliptische krommen.



ECDSA

Digitale handtekeningen, o.a.
gebruikt door **Bitcoin**.



TLS/HTTPS

Moderne TLS gebruikt vrijwel
altijd ECC voor de
sleuteluitwisseling.

QUANTUMCOMPUTERS

Net als RSA is ook ECC **kwetsbaar** voor quantumcomputers.

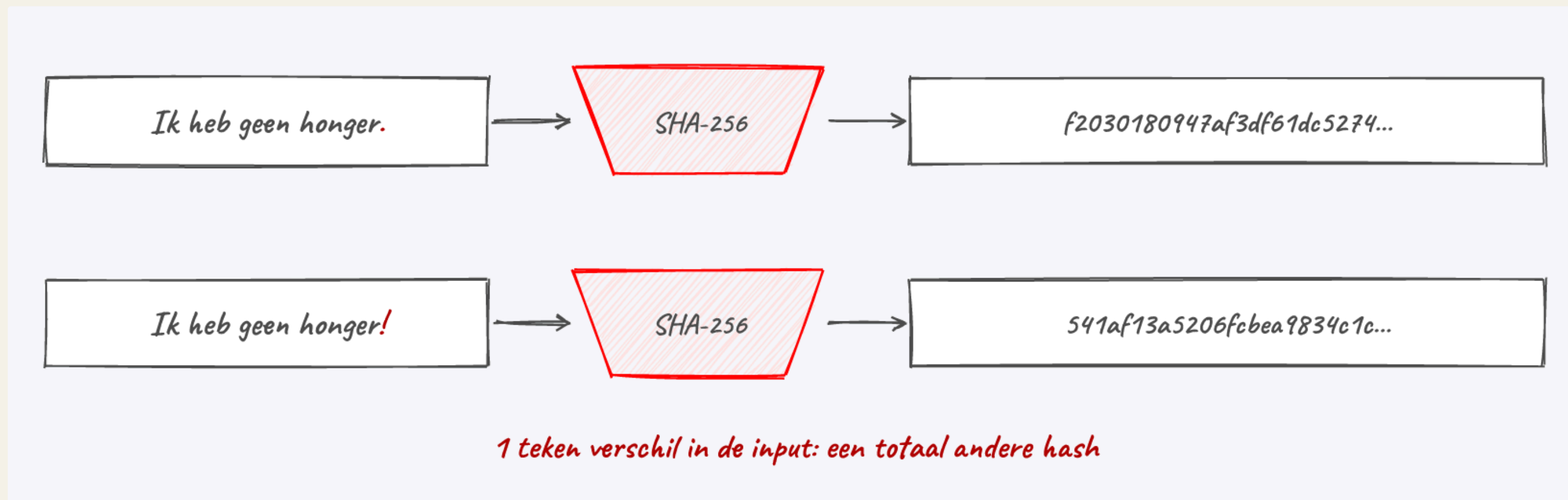
Daarom werkt men aan **post-quantum cryptografie**. NIST publiceerde in 2024 de eerste standaarden.

INTERMEZZO

Hashes en handtekeningen

Hashes

Een **hashfunctie** zet data om naar een code met **vaste lengte**, ongeacht de grootte van de input.



1 bit wijzigen → totaal andere hash

identieke input → identieke hash

Hashes: eigenschappen

Niet omkeerbaar

Van hash terug naar de originele tekst is onmogelijk: een **eenrichtingsfunctie**.

Hash collision

Twee verschillende inputs geven dezelfde hash. Theoretisch onvermijdelijk, want de hash is korter dan de input.

Goede hashfunctie

Houdt het aantal collisions zo **klein** mogelijk.

Bekende hashfuncties



MD5

128 bits

Verouderd, veel collisions.



SHA-256

256 bits

Veilig.



SHA3-512

512 bits

Nieuwste standaard.

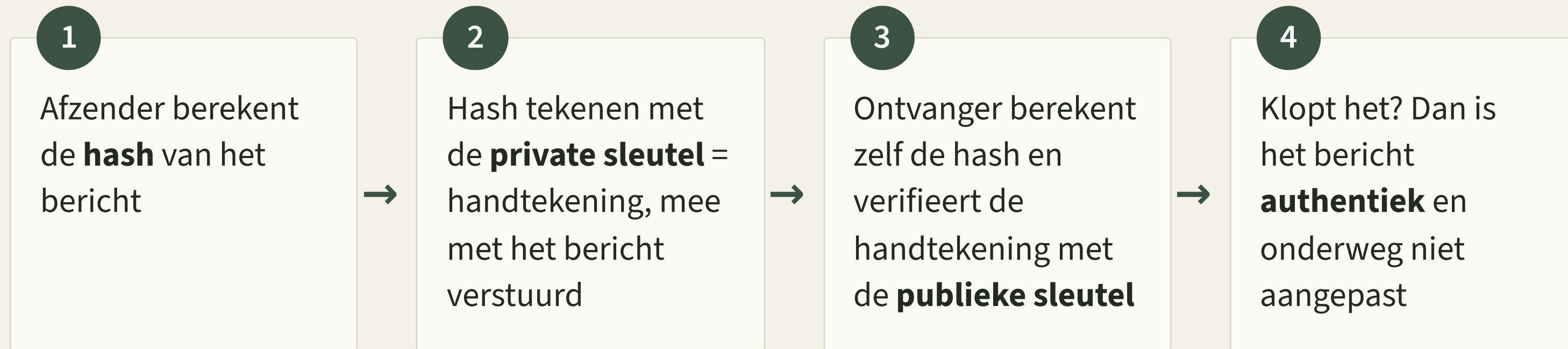
Een hash alleen bewijst niet **wie** het bericht stuurde. Combineer hem met een **private sleutel** en je kan ondertekenen.

DIGITALE HANDTEKENINGEN

Hoe weet je dat een bericht **echt** van de afzender komt?

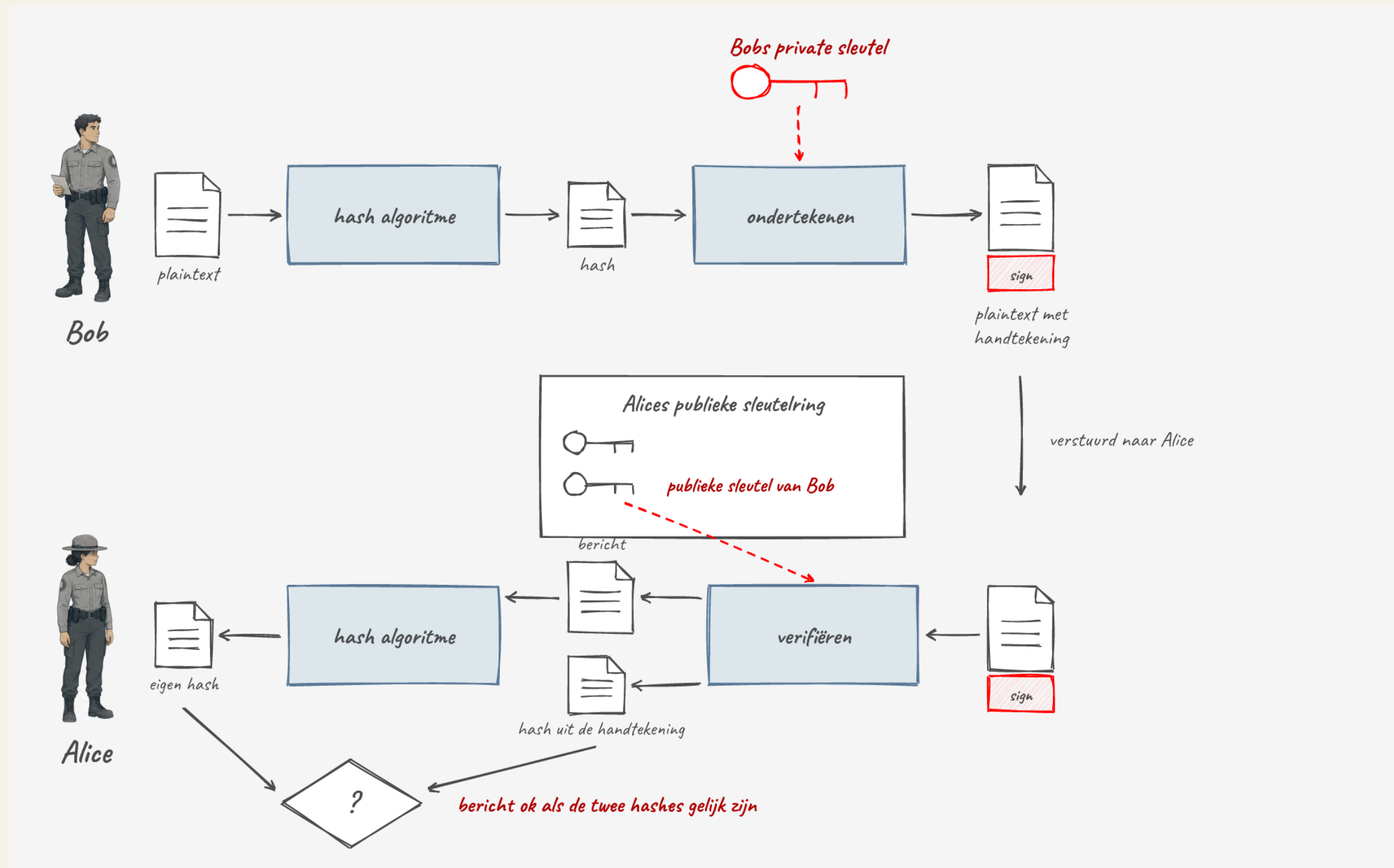
Met een **digital signature**: de afzender **tekent** de hash van het bericht met z'n **private sleutel**. De ontvanger **verifieert** met de bijhorende **publieke sleutel**.

Een handtekening maken en controleren



Bij RSA lijkt tekenen wiskundig op “versleutelen met de private sleutel”. Bij andere algoritmes (bv. ECDSA) niet, dus spreek van **tekenen** en **verifiëren**.

Digitale handtekeningen: hele proces



Digitale handtekeningen: rekenvoorbeeld

Gegeven sleutelpaar: publiek: $e = 5, n = 91$ privaat: $d = 29$



Ondertekenen

Hash van de boodschap: $m = 35$

$$s = m^d \% n = 35^{29} \% 91 = 42$$

Verstuur: de boodschap zelf + handtekening
42 (over de hash **35**)



Verifiëren

Ontvanger gebruikt de publieke sleutel e, n :

$$42^e \% n = 42^5 \% 91 = 35$$

Hash komt overeen: het bericht is **authentiek!**

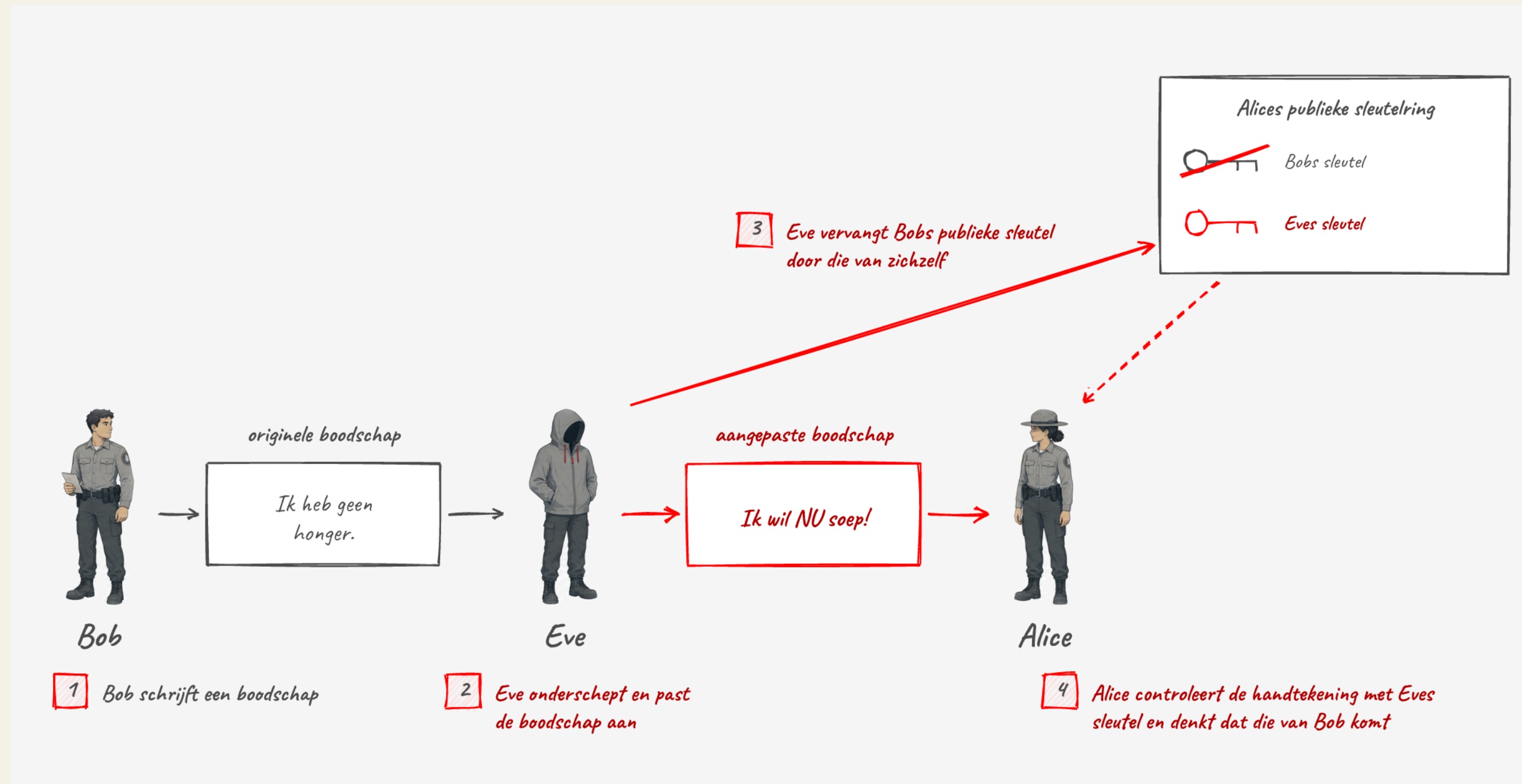
Het probleem met digitale handtekeningen

Publieke sleutels zijn... **publiek**. Hoe weet je dat een publieke sleutel **echt** bij persoon X hoort?

Een aanvaller kan:



Man-in-the-middle bij handtekeningen



Eve misbruikt het impliciete vertrouwen dat zit ingebouwd in het digitale handtekening proces.

VERTROUWEN

Digitale certificaten

We hebben een manier nodig om de **identiteit van de eigenaar** van een publieke sleutel te verifiëren.

DIGITAAL CERTIFICAAT

Een digitaal document dat een **identiteit** bindt aan een **publieke sleutel**.

Ondertekend door een **trusted third party**: vertrouw je die derde partij, dan vertrouw je het certificaat. Beschreven in de `x.509`-standaard.

Certificaat aanmaken

1

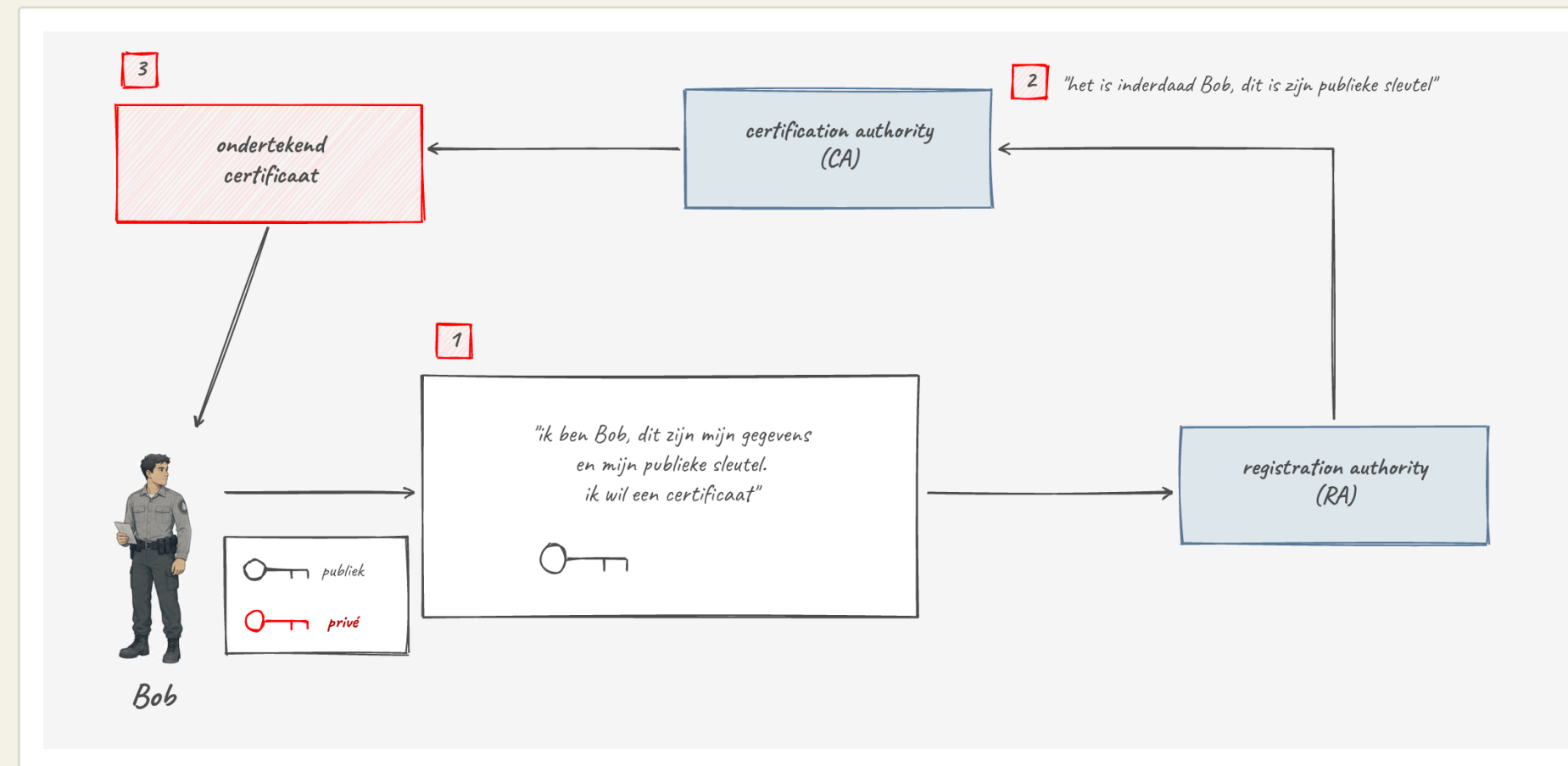
Bob bewijst zijn identiteit bij een **Registration Authority** (RA): ID-kaart, paspoort, soms fysiek op kantoor

2

De RA stuurt de aanvraag + Bobs publieke sleutel naar de **Certification Authority** (CA)

3

De CA maakt het certificaat aan én **ondertekent** het



Wat zit er in een certificaat?



Leesbare inhoud

Bobs **publieke sleutel** en info over Bob en de CA.



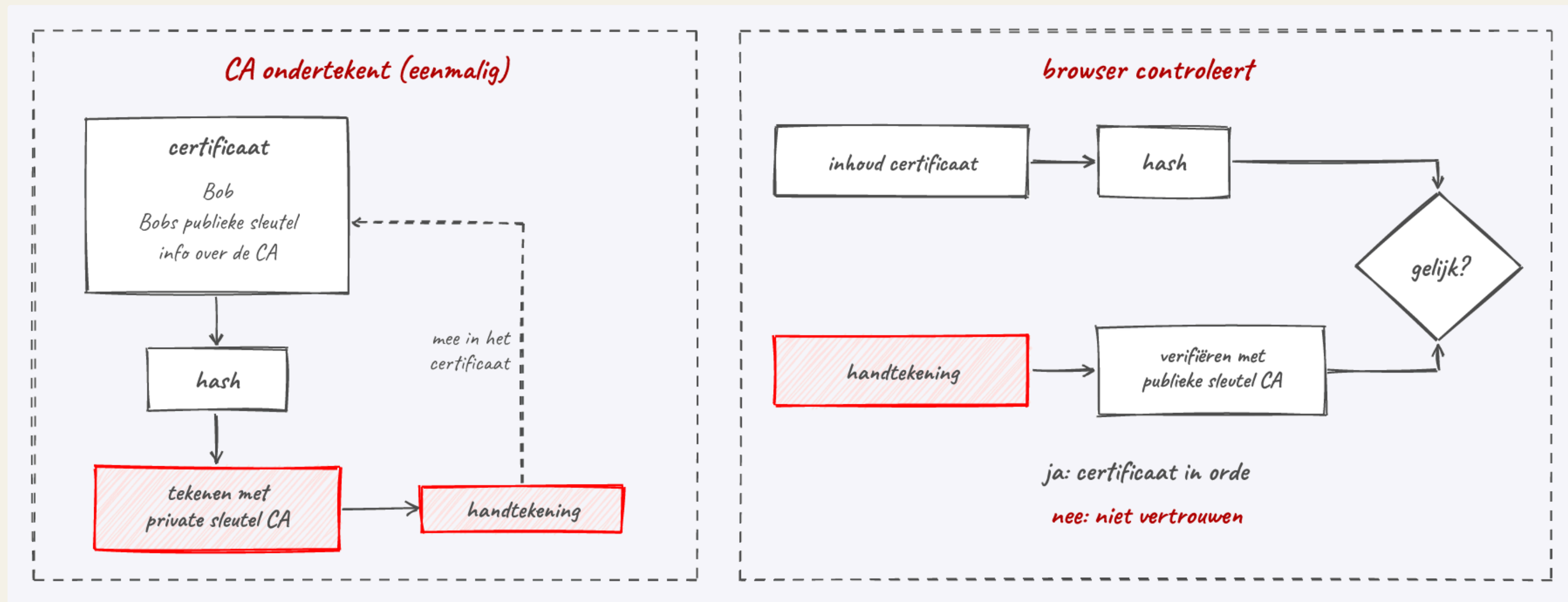
Handtekening van de CA

Een hash van die inhoud, getekend met de **private sleutel van de CA**.

Het hele systeem van CA's, RA's, ... voor het uitgeven, beheren en bewijzen van certificaten heet een **Public Key Infrastructure** (PKI).

Verificatie van certificaten

Zelf de hash berekenen → de handtekening verifiëren met de **publieke sleutel van de CA**.



X.509: inhoud van een certificaat

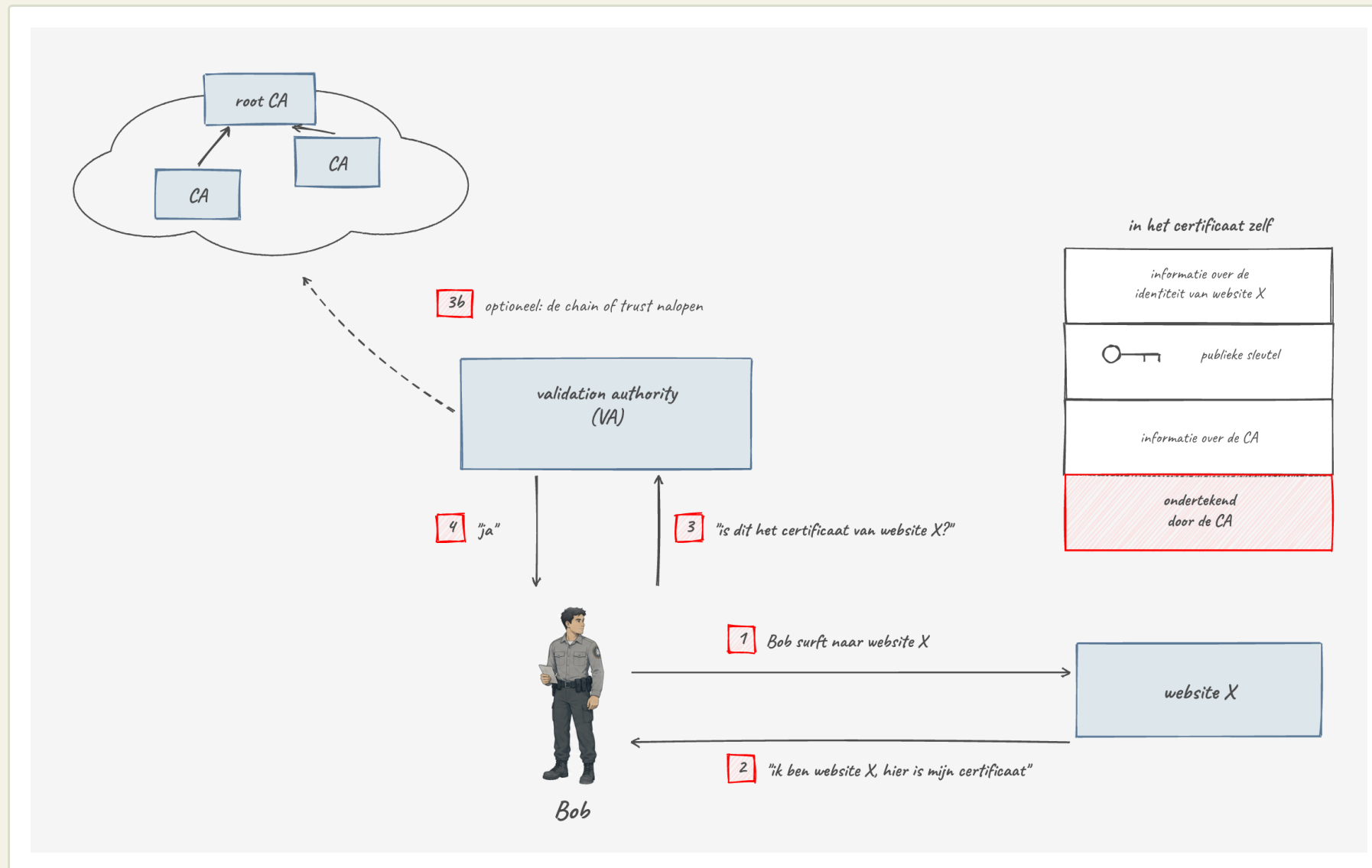
Versie	van de standaard (doorgaans v3)
Serienummer	uniek binnen de CA
Signature algorithm	bv. SHA-256 met RSA
Issuer	de CA die het certificaat uitgaf
Validity	de geldigheidsperiode
Subject	de eigenaar (domeinnaam of persoon)
Public key	de publieke sleutel van de eigenaar
Extensions	gebruiksbeperkingen, alternatieve domeinnamen, ...
Signature	de digitale handtekening van de CA

Chain of Trust



Vertrouw je de root CA? Dan vertrouw je de hele keten. Maar valt één CA in de keten, dan is **niets eronder** nog te vertrouwen.

Certificaat in de browser



Het certificaat tijdens het surfen.

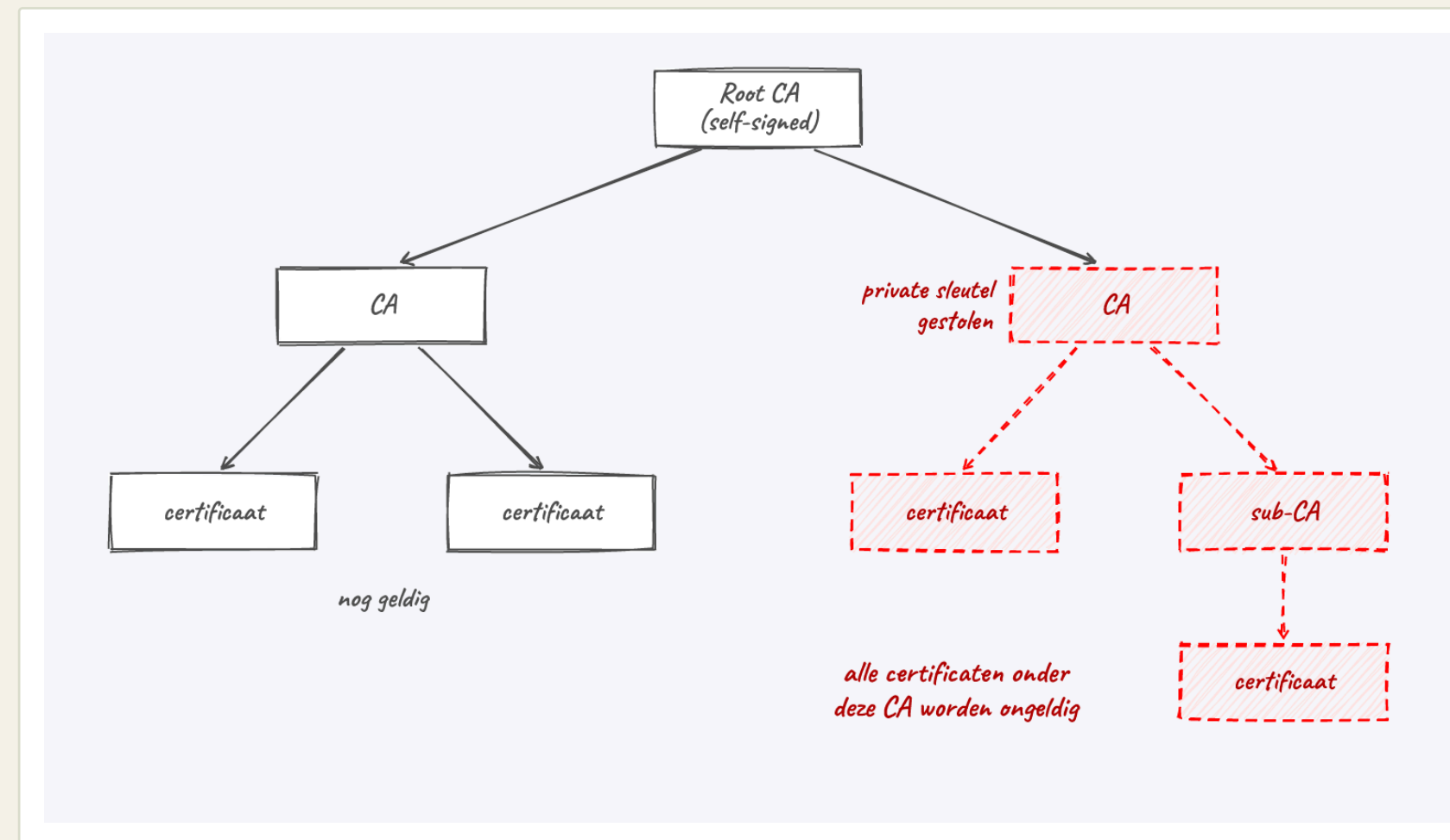
HTTPS en certificaten
werken **in tandem**:
encryptie én
identiteitsverificatie.

2017

pas dan biedt **meer dan de helft**
van de websites HTTPS aan

Breach of Trust

Het ergste scenario: de **private sleutel van een CA** wordt gestolen.



Aanvallers maken dan certificaten **op naam van de CA**. **Alle** certificaten van die CA (en sub-CA's!) worden ongeldig.

Wie vertrouw je nog?



AI GENERATED

2011

Case: DigiNotar



De hack

Nederlandse CA gehackt.
Valse certificaten voor o.a.
`google.com`, ingezet om
Iraanse Gmail-gebruikers te
bespioneren.



De reactie

Browsers verwijderen
DigiNotar als vertrouwde
root CA. **Alle** DigiNotar-
certificaten worden ongeldig,
ook die van DigiD en de
Nederlandse overheid.



Het gevolg

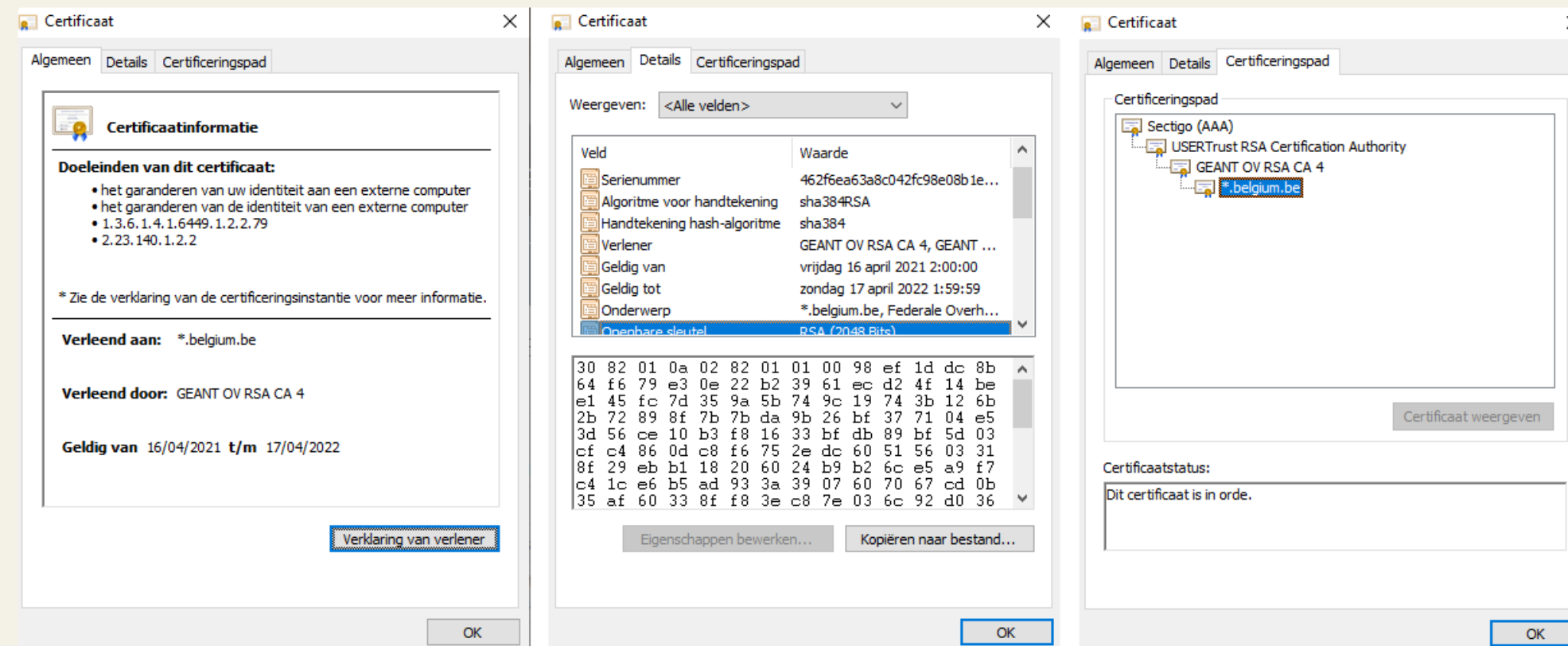
DigiNotar gaat binnen enkele
weken **failliet**.

DIGITALE CERTIFICATEN

In de praktijk

Certificaten bekijken in de browser

1. Klik op het **slotje** naast de URL
2. Je ziet geldigheidsduur, CA, publieke sleutel, algoritmes, ...
3. Tab **Certificeringspad** = de chain of trust



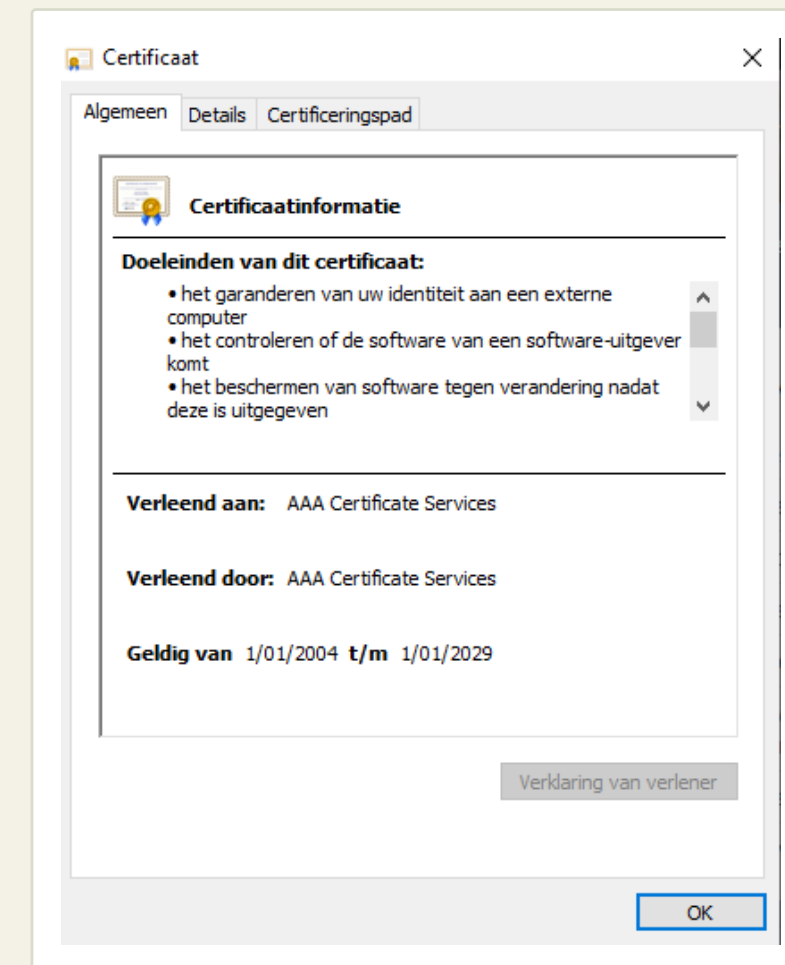
Het certificaat van België.

Self-signed certificaten

Root CA's hebben een **self-signed certificate**:

Verleend aan = Verleend door

Vertrouw je de root CA niet? Dan is de **hele keten** niet te vertrouwen.



Sectigo heeft een self-signed certificaat.

Soorten certificaten



SSL-certificaat

Bewijst de identiteit van een **webserver**.



Persoonlijk certificaat

Bewijst je **eigen identiteit**,
bv. bij e-mail.



Code signing

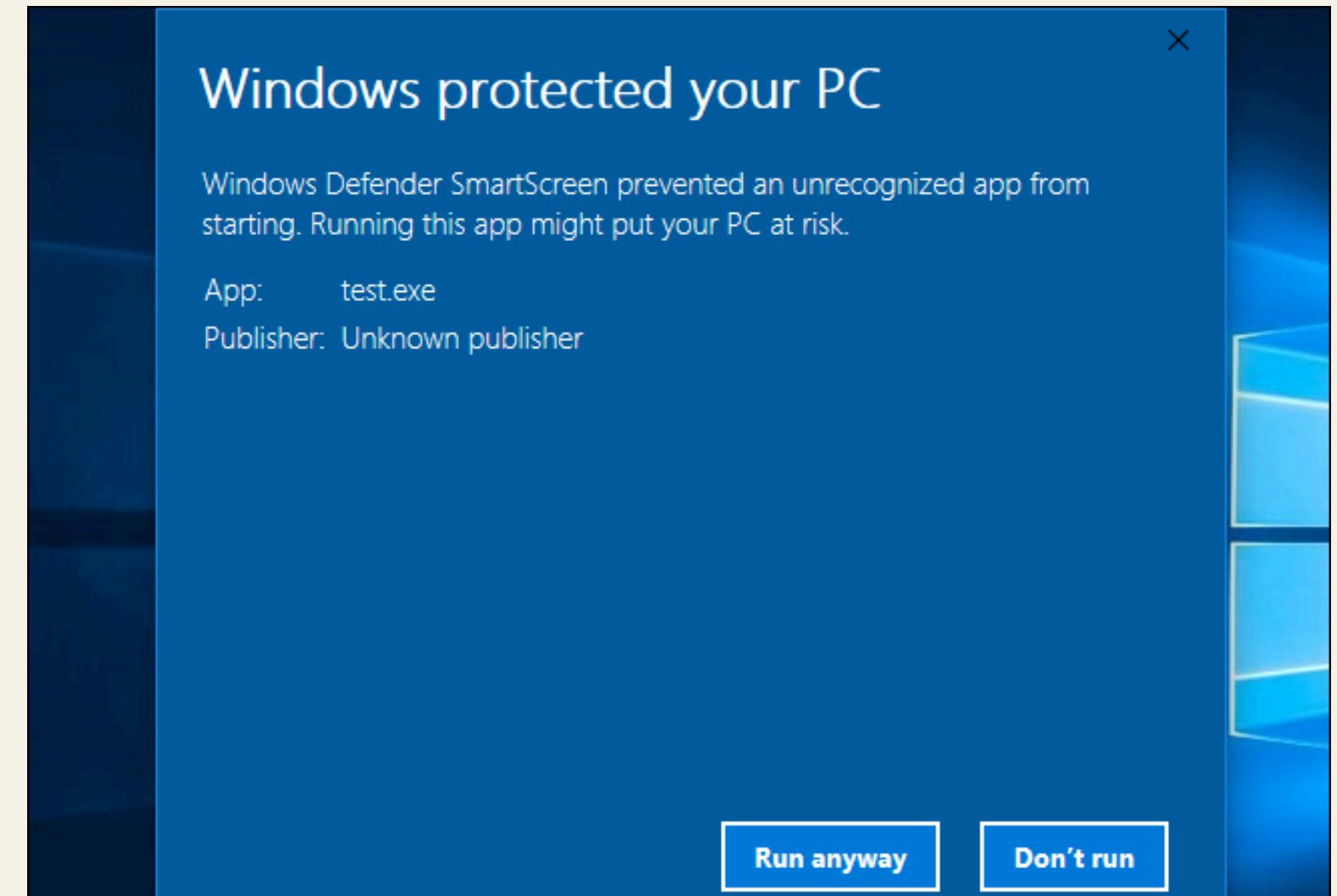
Bewijst de **echtheid van software**.

Windows SmartScreen

Controleert automatisch het **certificaat** van applicaties.

Geen certificaat? Dan krijg je een **waarschuwing**.

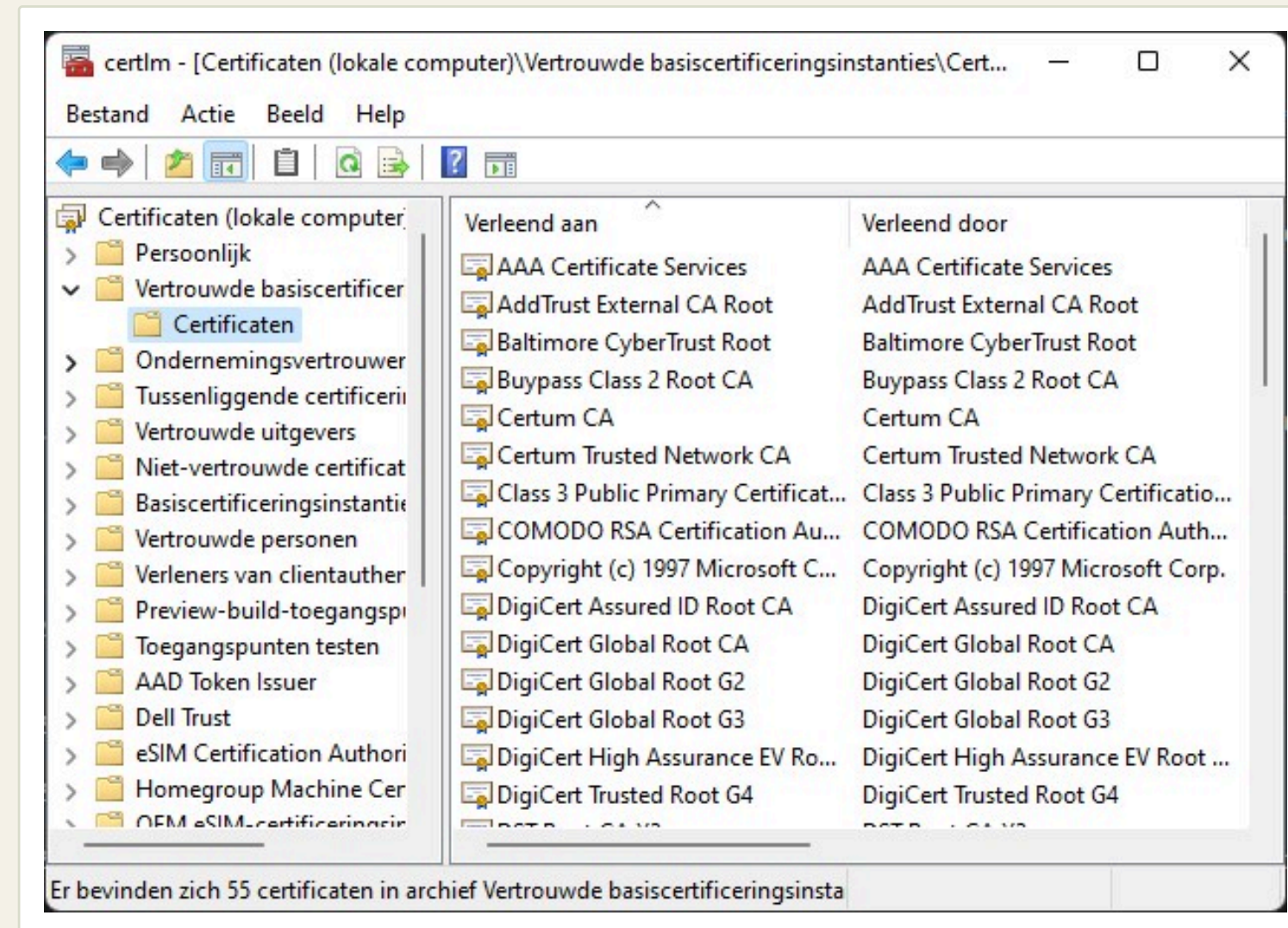
Geen certificaat $\neq$ automatisch onveilig.
Certificaten kosten geld en niet elke ontwikkelaar kan dat betalen.



Windows Certificate Manager

Bekijk welke certificaten lokaal geïnstalleerd zijn en welke vertrouwd worden.

Start de tool met `certlm.msc`.



Let er altijd op welke informatie je deelt via screenshots.

WEB OF TRUST

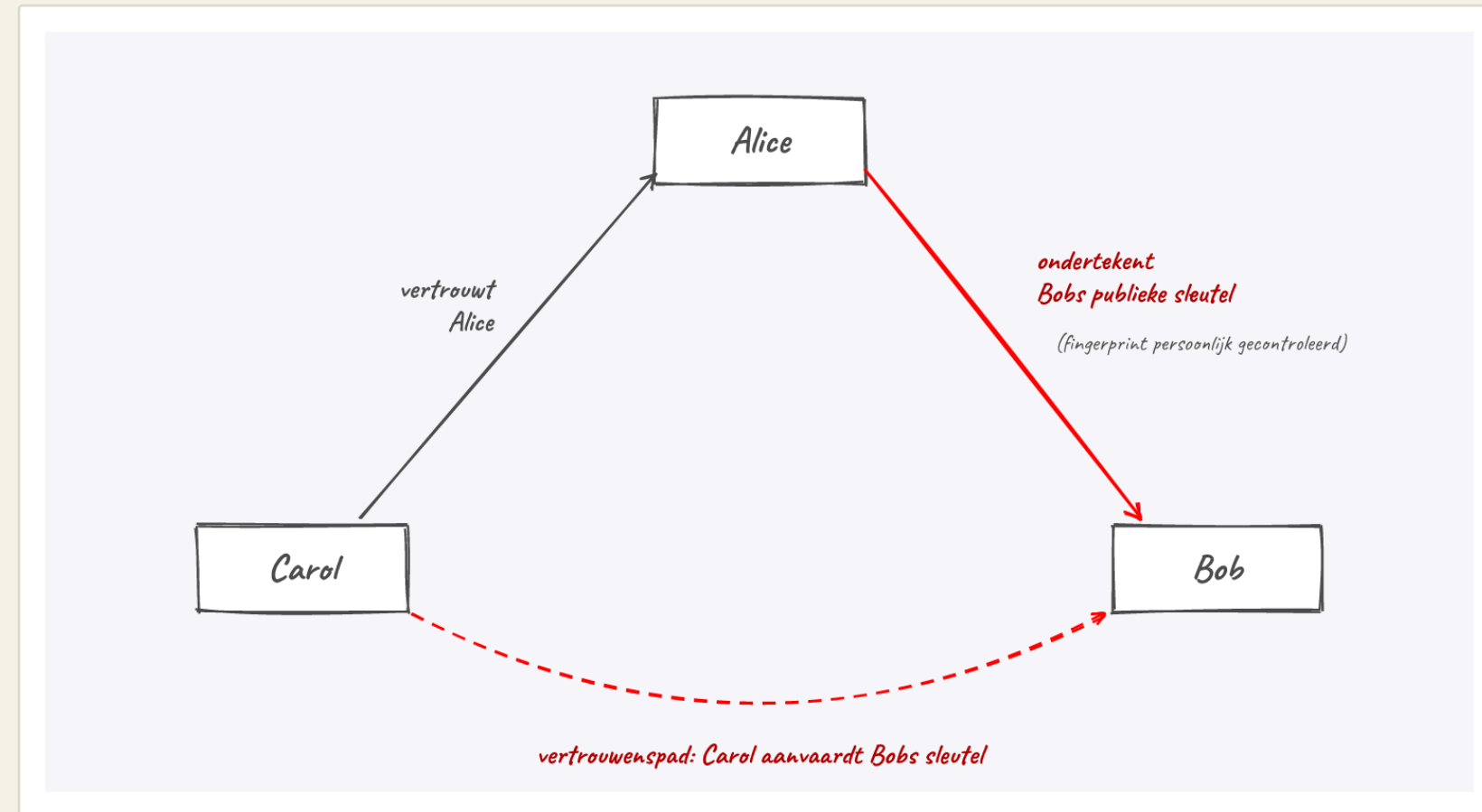
Geen centrale autoriteit: gebruikers ondertekenen **elkaars** publieke sleutels.

Een **gedecentraliseerd** alternatief voor PKI, bekend van PGP/GPG.

Alice verifieert Bobs sleutel persoonlijk en ondertekent die met haar private sleutel: “Ik bevestig dat deze sleutel van Bob is.”

Web of Trust: vertrouwenspaden

- Carol kent Bob niet, maar **vertrouwt Alice**
- Alice heeft Bobs sleutel ondertekend
- Via dat **vertrouwenspad** aanvaardt Carol Bobs sleutel
- Zo ontstaat een **web** van onderlinge vertrouwensrelaties



PKI vs. Web of Trust

PKI

- **Hiërarchisch:** vertrouwen top-down
- Validatie door Certificate Authorities
- Zwak punt: 1 CA gehackt = iedereen getroffen
- HTTPS, S/MIME, code signing

vs

Web of Trust

- **Peer-to-peer**
- Validatie door de gebruikers zelf
- Zwak punt: vergt actieve deelname
- PGP/GPG e-mail, softwarepakketten

WoT bleek moeilijk **schaalbaar**: key signing parties, vertrouwenspaden zoeken, ...
Daarom domineert PKI op het Internet.

VEILIG SURFEN

HTTPS en TLS

Certificaten vormen het hart van een veilige manier van surfen.

HTTPS en TLS

HTTPS = HTTP + een beveiligde encryptietunnel.

443

de poort voor **HTTPS**

80

de klassieke poort voor HTTP

TLS *Transport Layer Security*: het protocol dat de tunnel opzet

SSL *Secure Sockets Layer*: de voorganger van TLS

Wat doet TLS?

1

Identiteitscontrole: het certificaat (asymmetrische crypto) verifieert de website



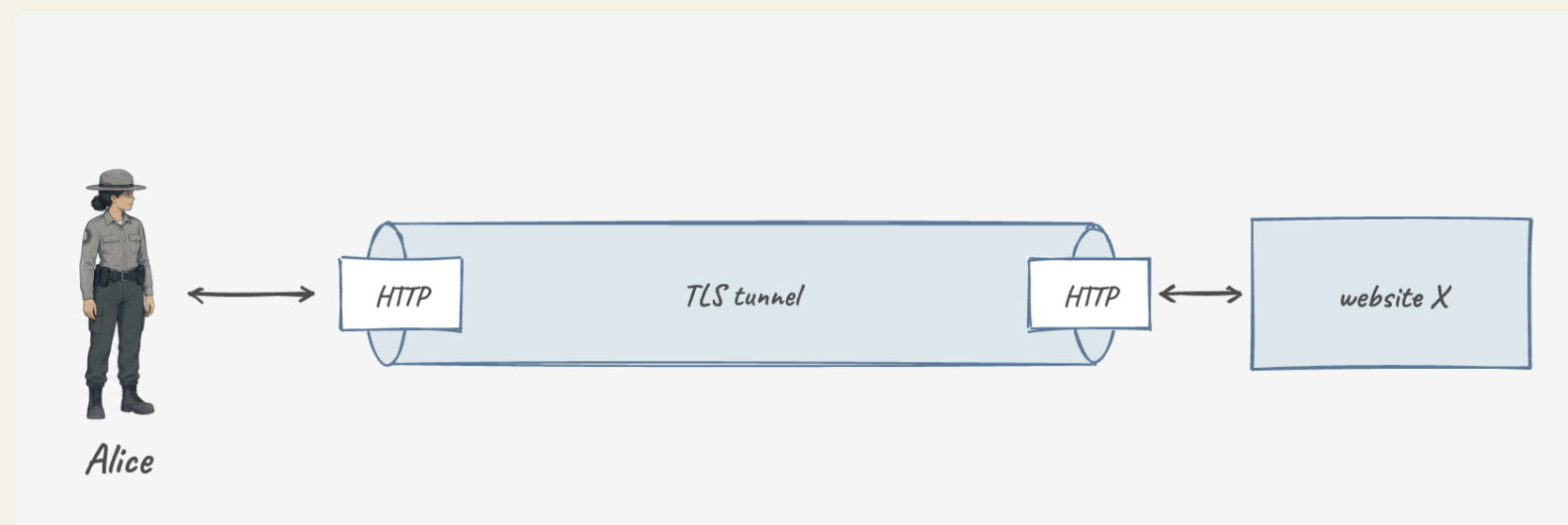
2

Sleuteluitwisseling: een gemeenschappelijke sleutel afspreken (meestal Diffie-Hellman op elliptische krommen, *ECDHE*). Nieuwe waarden per sessie = **forward secrecy**



3

Tunnel: vanaf nu kunnen derden de trafiek niet meer sniffen



De erg belangrijke TLS tunnel tijdens het surfen met HTTPS.

TLS: het beste van beide werelden



Asymmetrisch

Veilige **sleuteluitwisseling**, maar traag.



Symmetrisch

Snelle **data-encryptie**.

TLS/HTTPS combineert de sterktes van beide soorten crypto.

Meer weten? Bekijk de briljante visualisatie op tls.ulfheim.net

Mutual TLS (mTLS)

Gewone HTTPS

Enkel de **server** heeft een certificaat.

vs

mTLS

Ook de **client** authenticceert met een certificaat. De server weet zeker wie de client is.

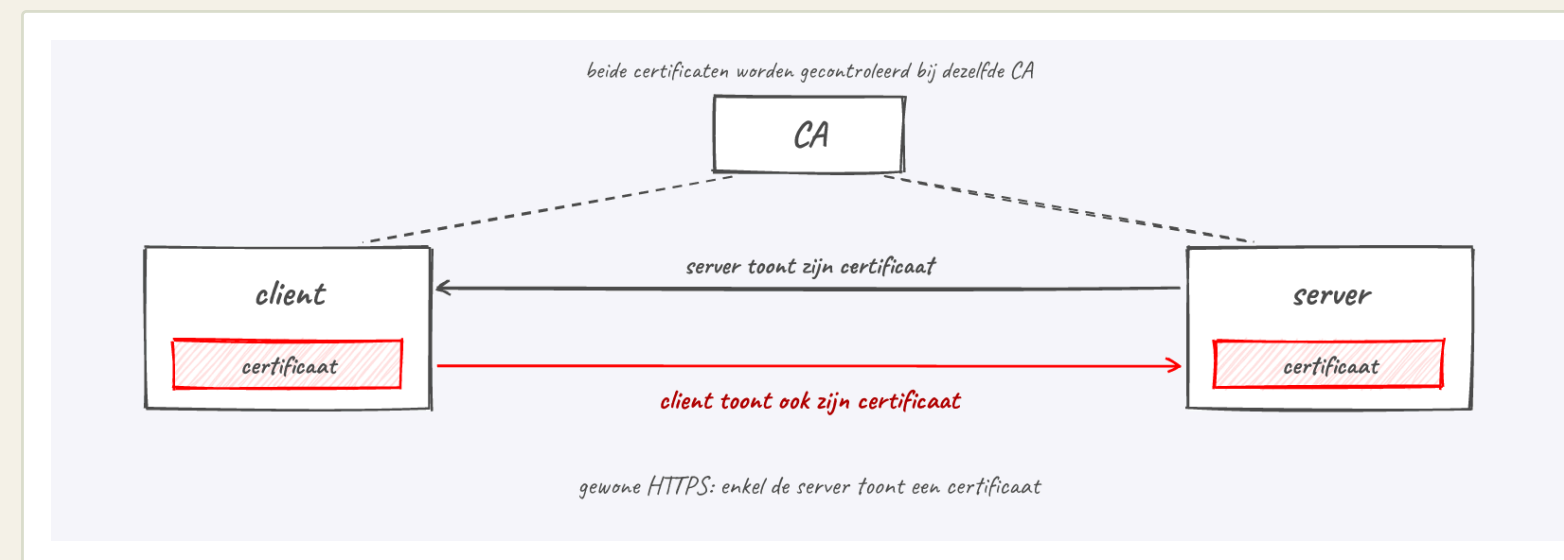
Typisch gebruikt in:

bankomgevingen

e-government

bedrijfs-VPN's

backend-servers



HTTPS: nadelen

HTTPS maakt je verbinding een pak veiliger, maar voor je ISP en de website zijn er ook nadelen.



Geen caching

Alles is geëncrypteerd: je ISP kan niet meer cachen.



Meer trafiek

De website moet meer trafiek verwerken, want er is geen tussenliggende cache.

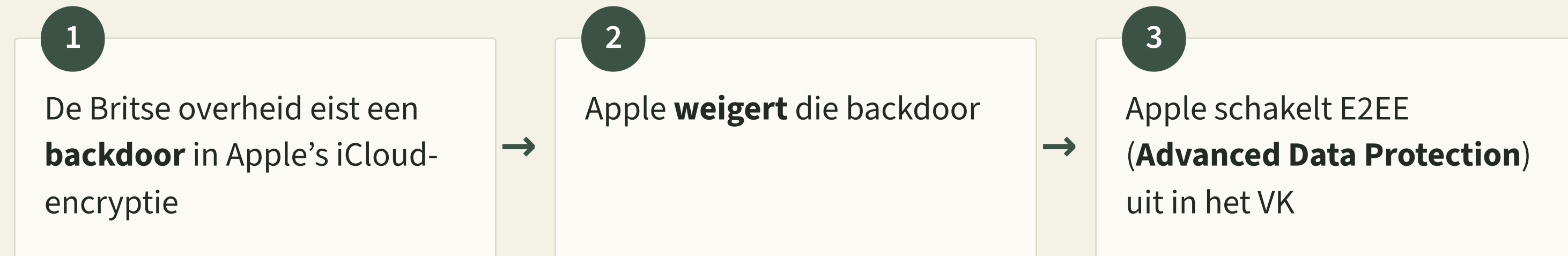
Een extra deurtje, enkel voor de goeden?



AI GENERATED

E2E-encryptie onder druk: Apple en de UK

End-to-end encryptie (E2EE): enkel zender en ontvanger kunnen de data lezen.



iMessage, FaceTime en iCloud Keychain behouden wel E2EE.



Een backdoor die enkel voor “de goeden” werkt, **bestaat niet**.

Een achterpoortje verzwakt de beveiliging voor **alle** gebruikers. Dat is in strijd met Kerckhoffs principe.

2022

Chat Control: ook in de EU

Het voorstel

EU-voorstel: de **Child Sexual Abuse Regulation**.

Verplicht scannen van **alle** chatberichten, ook E2E-versleutelde.

vs

De critici

Technisch is dit een **backdoor** of *client-side scanning* (spyware).

Na jarenlange controverse werd het verplicht scannen van E2EE in **2025** afgezwakt.

Je kan niet tegelijk **echte** E2E-encryptie garanderen én berichten laten meelezen.

Man-in-the-middle: mitmproxy

mitmproxy: tool (Linux, macOS, Windows) voor een MITM-aanval op HTTPS.

1

Aanvaller nestelt zich tussen **client** en **website**



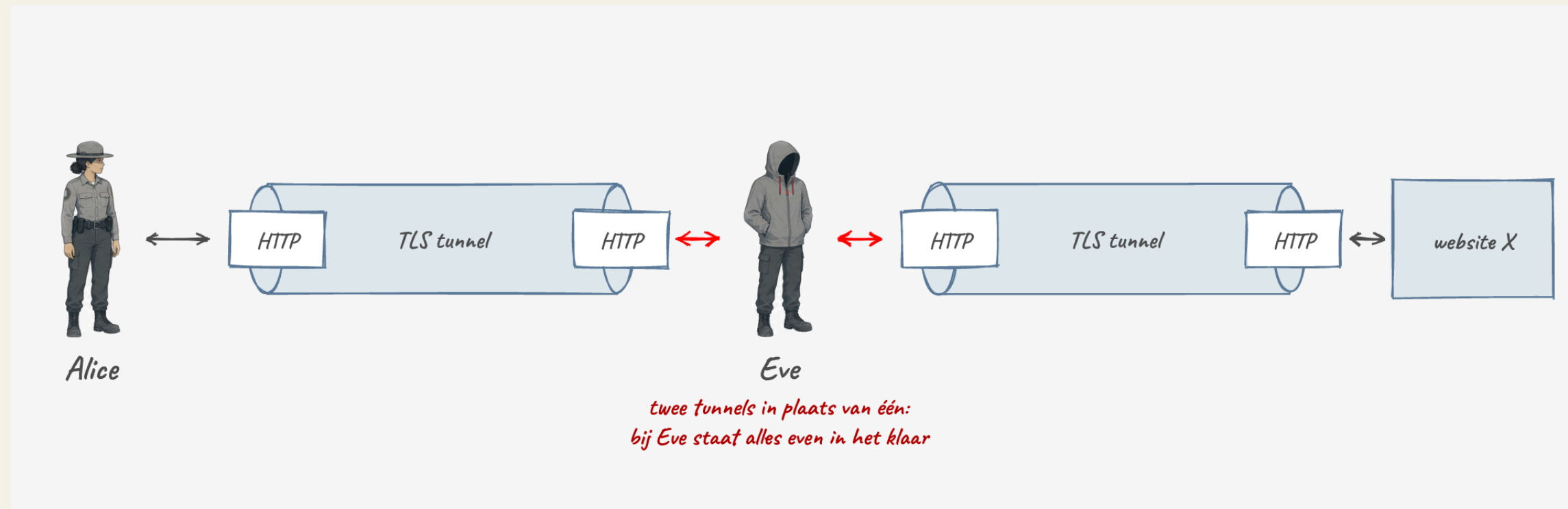
2

Zet **eigen TLS-tunnels** op naar beide kanten



3

Kan alle trafiek **lezen en aanpassen**



Een man-in-the-middle aanval met TLS.

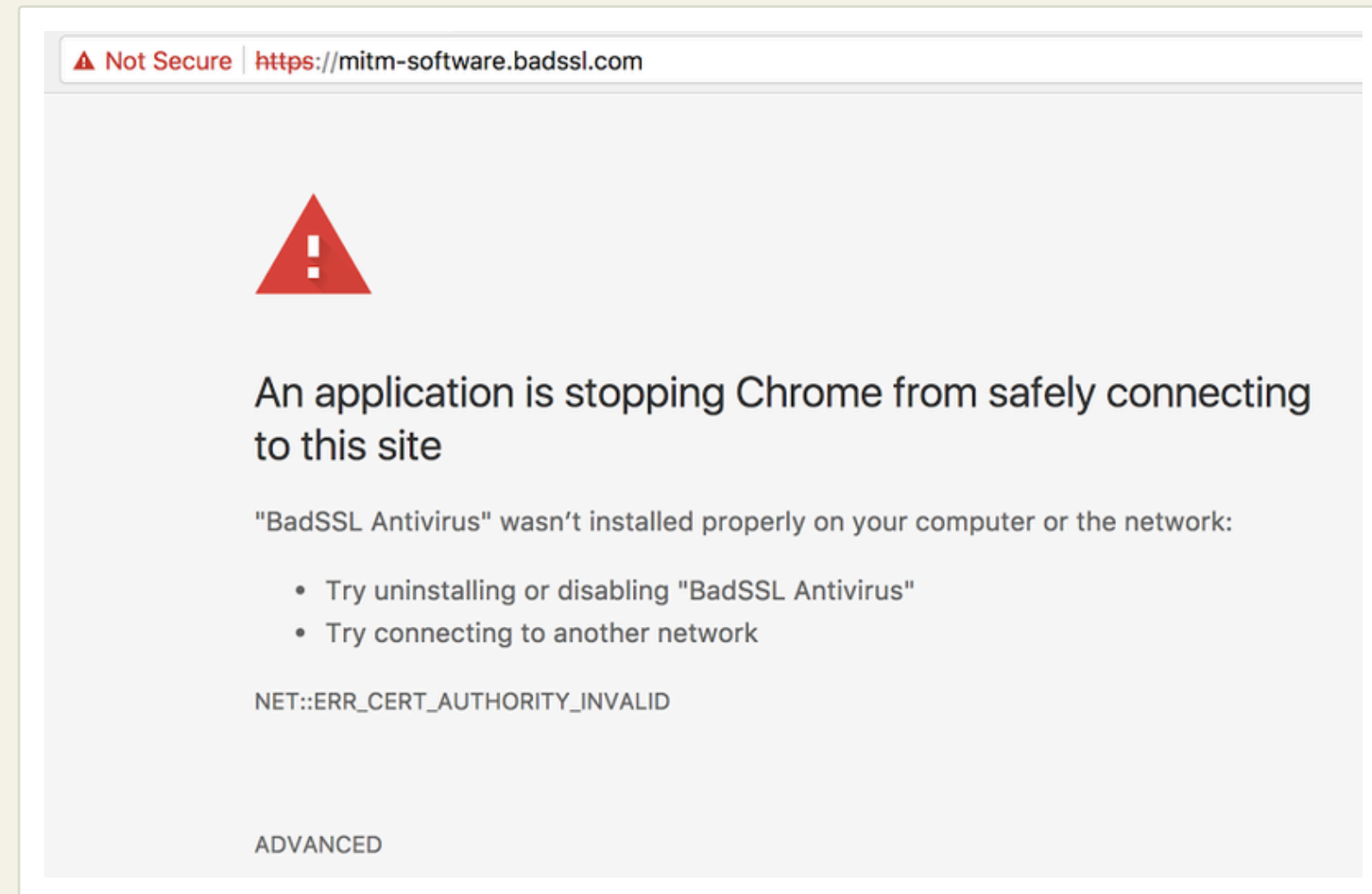
Mitmpoxy: verdediging

De aanvaller kan **geen geldige certificaten** genereren.

De browser geeft dan een **waarschuwing**. Tenzij de aanvaller zijn **eigen CA** als vertrouwd op je toestel krijgt: dan zie je niets.



Negeer nooit certificaat-waarschuwingen in je browser!



De waarschuwing die Chrome genereert wanneer het een, potentiële, mitm-aanval detecteert.

Conclusie

- ✓ **Asymmetrische crypto** lost het sleuteldistributieprobleem op
- ✓ Twee sleutels: publiek (encryptie, verificatie) en privaat (decryptie, ondertekenen)
- ✓ **Digitale handtekeningen** en certificaten bewijzen identiteit
- ✓ **TLS/HTTPS** combineert asymmetrisch en symmetrisch voor snelle, veilige communicatie
- ✓ De **chain of trust** (CA's) is de ruggengraat van vertrouwen op het Internet

ONTHOUD

Bescherm je **private sleutel**, verifieer **certificaten** en negeer nooit **browserwaarschuwingen**.