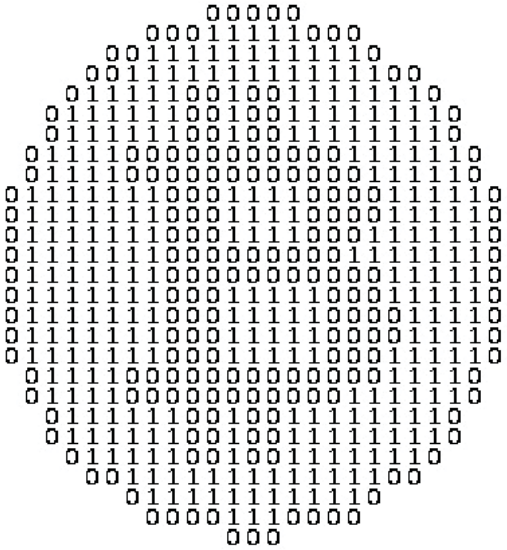
De cryptografie achter Bitcoin



Benne de Weger
b.m.m.d.weger@tue.nl

augustus 2018

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

digitale handtekeningen

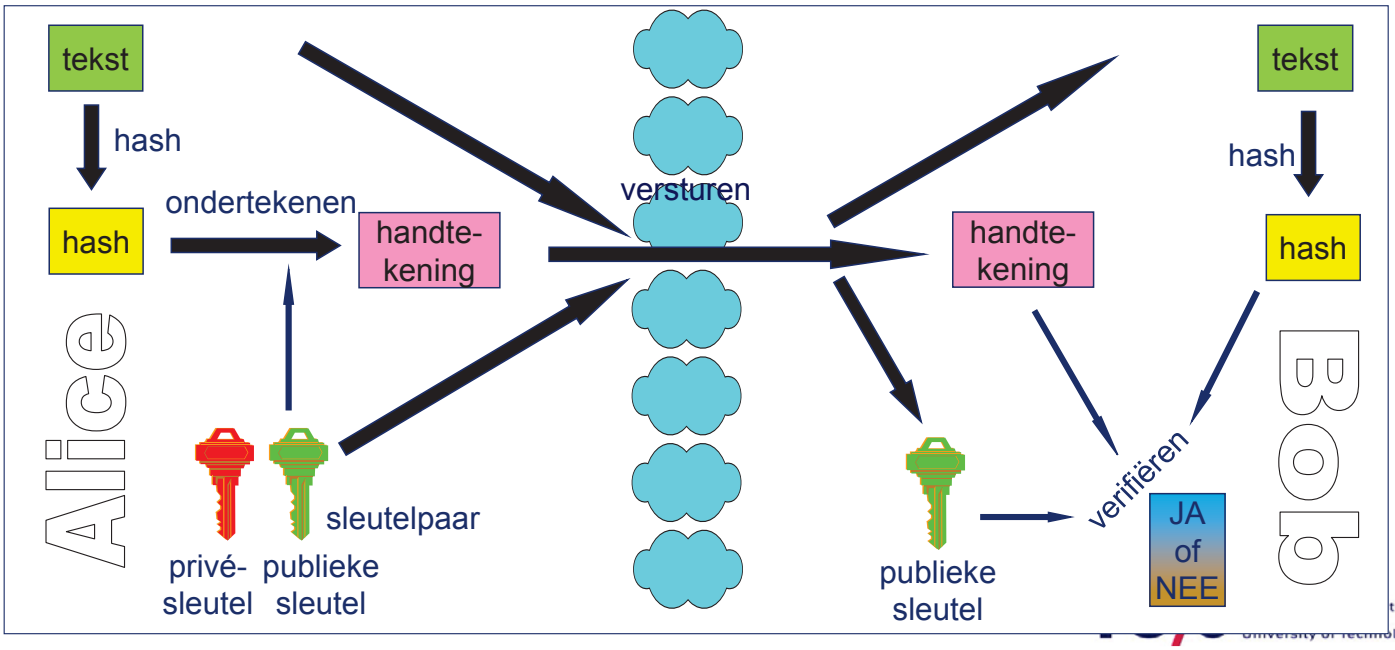
doel: authenticatie

- **sterke verbinding aanleggen tussen een document en een identiteit**
- **wordt doorgaans *handtekening* genoemd**

***digitale* handtekening kan met behulp van *asymmetrische* cryptografie**

- **ondertekenaar moet iets gebruiken wat alléén zij heeft**
  - **daar kan een privé-sleutel voor dienen**
- **een handtekening moet voor iedereen te checken zijn**
  - **daar kan een publieke sleutel voor dienen**

digitale handtekeningen



digitale handtekeningen met RSA

Alice heeft een bericht m
en een RSA sleutelpaar: publieke sleutel n, e ; privé-sleutel (n,) d

Alice berekent een hash h = h(m)
ondertekent met machtsverheffing met de privé-sleutel:

$$s \equiv h^d \pmod{n}$$

Bob berekent onafhankelijk van Alice de hash: h = h(m)
en Bob verifieert de handtekening:

is $s^e \equiv h \pmod{n}$ **?**

digitale handtekeningen met DSA

parameters:

- groep F_p^* met een Discrete Logaritme-Probleem
- voortbrenger g van een ondergroep van orde priemgetal q (deler van p-1)

Alice maakt sleutelpaar:

- privé-sleutel x willekeurig, $1 < x < q-1$, publieke sleutel $y \equiv g^x \pmod p$

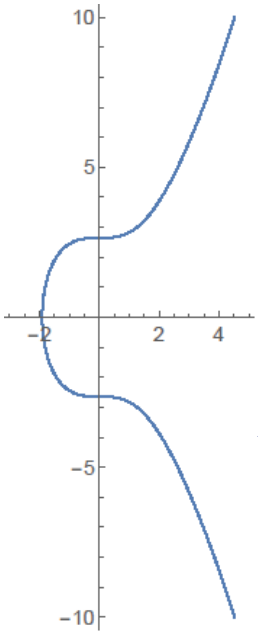
Alice ondertekent:

- berekent hash h
- kiest eenmalig getal k, $1 < k < q-1$
- berekent $r \equiv g^k \pmod p$ en $s \equiv (h + x\,r) / k \pmod q$, handtekening: r, s

Bob verifieert:

- berekent weer hash h
- berekent $u_1 \equiv (h / s \pmod p) \pmod q$, $u_2 \equiv r / s \pmod q$
- gaat na of $r \equiv (g^{u1} y^{u2} \pmod p) \pmod q$

de Bitcoin-kromme

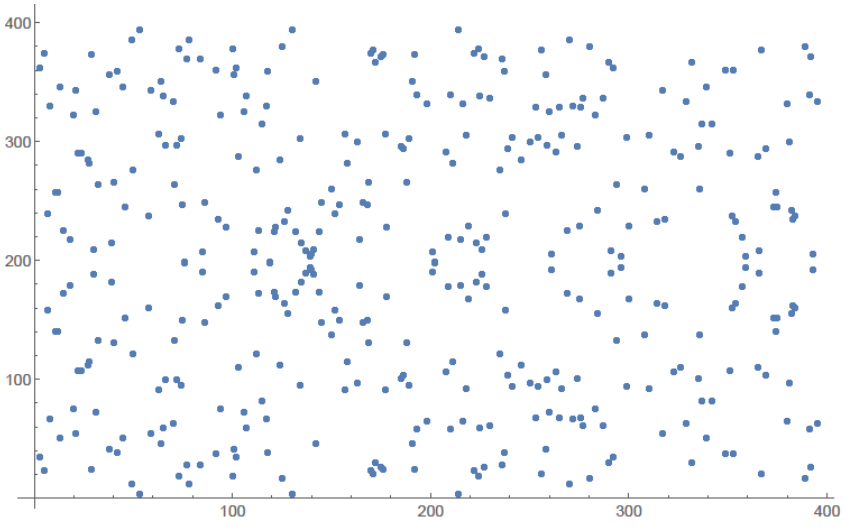


Bitcoin-kromme:

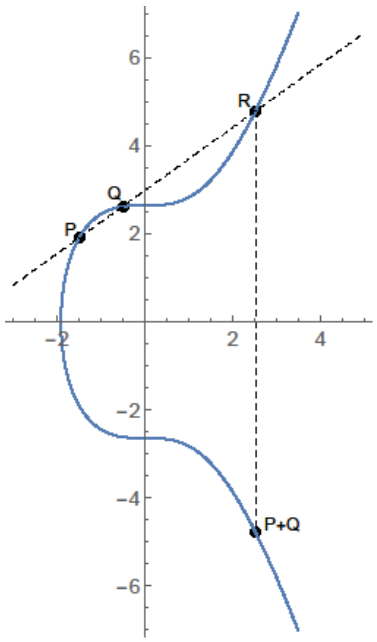
$y^2 = x^3 + 7$

x, y reëel

x, y in F_p met p = 397



optellen op de Bitcoin-kromme



S* = *P* + *Q

$$\begin{aligned} \lambda &= \frac{y_\mathrm{Q} - y_\mathrm{P}}{x_\mathrm{Q} - x_\mathrm{P}}, \\ x_\mathrm{S} &= \lambda^2 - x_\mathrm{P} - x_\mathrm{Q}, \\ y_\mathrm{S} &= -y_\mathrm{P} + \lambda(x_\mathrm{P} - x_\mathrm{S}). \end{aligned}$$

S* = 2*P
(raaklijn)

$$\begin{aligned} \lambda &= \frac{3x_\mathrm{P}^2}{2y_\mathrm{P}}, \\ x_\mathrm{S} &= \lambda^2 - 2x_\mathrm{P}, \\ y_\mathrm{S} &= -y_\mathrm{P} + \lambda(x_\mathrm{P} - x_\mathrm{S}). \end{aligned}$$

***P* = (*x*,*y*)**
-*P* = (*x*,-y)

punt op oneindig
∞ (geen coördinaten)

P* + ∞ = *P
***P* + (-*P*) = ∞**

TU/e Technische Universiteit Eindhoven University of Technology

optelwet op Bitcoin-kromme

met ∞ als de “nul” gelden alle gewone eigenschappen van optellen

- **commutativiteit: (volgorde doet er niet toe)**

  P* + *Q* = *Q* + *P

  - **volgt direct uit de definitie**
- **associativiteit (je mag haakjes verzetten)**

  (*P* + *Q*) + *T* = *P* + (*Q* + *T*)

  - **is echt lastig te bewijzen**

scalaire vermenigvuldiging:

n P* = *P* + *P* + … + *P

0 *P* = ∞

Bitcoin-kromme is een *additieve groep*

TU/e Technische Universiteit Eindhoven University of Technology

Bitcoin-kromme: klein voorbeeld, $p = 11$

12 punten, er is een voortbrenger: $P = (4,4)$

n	nP	n	nP
1	$(4,4)$	7	$(7,8)$
2	$(6,6)$	8	$(3,1)$
3	$(2,9)$	9	$(2,2)$
4	$(3,10)$	10	$(6,5)$
5	$(7,3)$	11	$(4,7)$
6	$(5,0)$	12	∞

orde van een punt Q:
de kleinste positieve n zodat $n\,Q = \infty$
(orde altijd een deler van aantal punten)

Bitcoin-kromme is een additieve groep
er zijn ondergroepen, bv. {(2,9),(5,0),(2,2),∞} voortgebracht door (2,9)

deze groep lijkt wel op F_p^* (multiplicatieve groep)
maar vertaal alles van multiplicatief naar additief:
machtsverheffen wordt scalair vermenigvuldigen

Bitcoin scalaire vermenigvuldiging

punt P met getal n vermenigvuldigen:
gebruik *verdubbel-en-optel*-methode

$23 = 2^4 + 2^2 + 2^1 + 2^0$ (binaire schrijfwijze)

dus $23P = 16P + 4P + 2P + P$ kost 3 optellingen na 4 verdubbelingen:

$$P \to 2P \to 4P \to 8P \to 16P$$

$n\,P$ kost maximaal $\log_2 n$ optellingen na $\log_2 n$ verdubbelingen

Bitcoin Discrete Logaritme

gegeven p, een voortbrenger P en een punt Q,
kun je de n vinden zodat $Q = n\,P$?

dit is het *Bitcoin Discrete Logaritme-probleem*
en dit is een erkend moeilijk probleem

beter dan brute kracht: er is een methode die het in $n^{1/2}$ stappen doet
nog steeds exponentieel

er is geen sub-exponentiële methode bekend zoals voor
Discrete Logaritmen over F_p^*

de echte Bitcoin-kromme secp256k1

$y^2 = x^3 + 7$ over F_p met (hexadecimale notatie)

p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F

voortbrenger van de kromme: $G = (x_G, y_G)$:

x_G = 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798

y_G = 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8

orde van G = aantal punten op de Bitcoin-kromme

n = FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141

dit is een priemgetal (dus er zijn geen echte ondergroepen)

Bitcoin Diffie-Hellman

alle cryptosystemen gebaseerd op Discrete Logaritmen in F_p^*
kun je direct overzetten naar cryptosystemen op de Bitcoin-kromme

Diffie-Hellman:
Alice maakt privé-sleutel x_A, publieke sleutel $P_A = x_A G$
Bob maakt privé-sleutel x_B, publieke sleutel $P_B = x_B G$

Alice maakt gedeelde geheim $S_A = x_A P_B$
Bob maakt gedeelde geheim $S_B = x_B P_A$

dan geldt $S_A = S_B$

Bitcoin ElGamal versleuteling

Alice en Bob maken een gedeeld geheim S met Bitcoin-Diffie-Hellman

neem nu bv. de x-coordinaat x_S van S

versleutelen: $c \equiv m\, x_S \pmod{p}$

ontsleutelen: $m \equiv c/x_S \pmod{p}$

Bitcoin ECDSA-digitale handtekening

Sleutelpaar van Alice:

privé-sleutel: random x between 1 and n

publieke sleutel: $Q = x\,G$

Alice

- berekent een *dubbele* hash h = SHA256(SHA256(m))
- kiest eenmalig getal k, $1 < k < n-1$
- berekent $R = k\,G$ en $r = x$-coördinaat van R
- berekent $s \equiv (h + x\,r) / k \pmod{n}$, handtekening: r, s

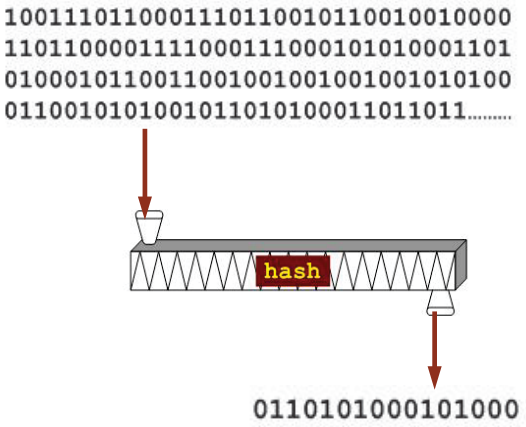
Bob verifieert:

- berekent weer hash h
- berekent $u_1 \equiv h / s \pmod{n}$, $u_2 \equiv r / s \pmod{n}$
- gaat na of r de x-coördinaat van $R = u_1G + u_2Q$ is

hashfuncties

doel van hashfunctie: unieke korte identificatie van digitaal bericht

- input: rijtje bits m van willekeurige lengte
- output: rijtje bits h(m) van vaste lengte n
  - tegenwoordig van 256 bits
  - *compressie*
  - heet vaak *hash, hashwaarde*, *digest*, *vingerafdruk*
- h(m) is makkelijk uit te rekenen uit m
- geen geheime informatie, geen sleutel



hashbotsingen

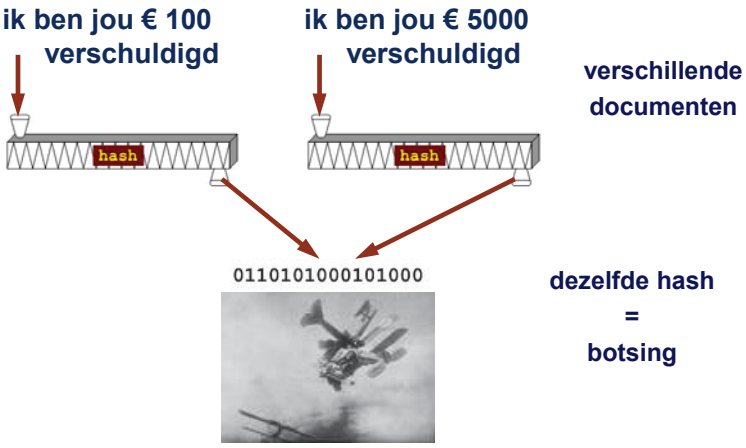
- m_1, m_2 zijn een *botsing* voor hashfunctie h als

  $h(m_1) = h(m_2)$ terwijl

  $m_1 \neq m_2$

er bestaan oneindig veel botsingen

maar vind er maar eens een...

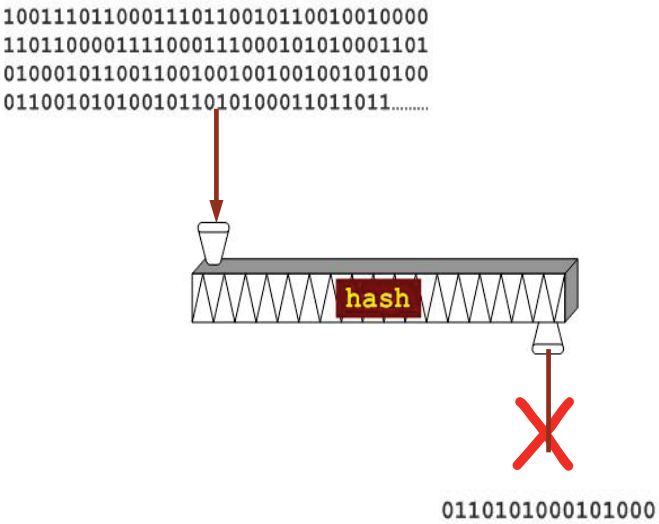


origineel

- gegeven h_0, dan heet m een *origineel* van h_0 als

  $h(m) = h_0$

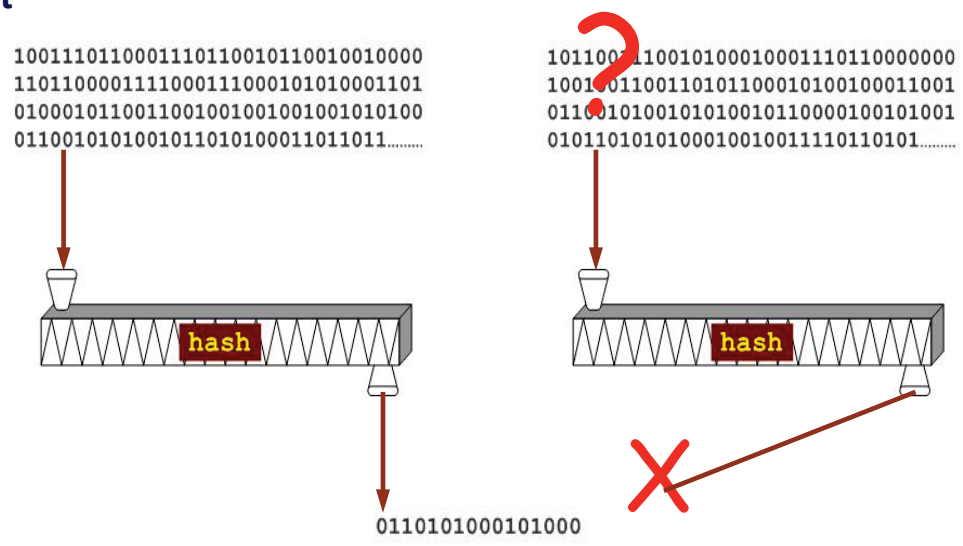
originelen moeten "onmogelijk" te vinden zijn



tweede origineel

- gegeven m_0, dan heet m een *tweede origineel* van m_0 als
  $h(m) = h(m_0)$
  terwijl $m \neq m_0$

tweede origineel moet ook moeilijk te vinden zijn



toepassingen van hashfuncties

- asymmetrische digitale handtekeningen
- bescherming integriteit (checksum)
- eenrichtings-versleuteling
  voor bescherming van wachtwoorden
- commitment op kennis
  voorspellingen
- bewijs van werk
  cryptomunten als Bitcoin
- pseudo-random getallen maken

enzovoorts

brute kracht-aanvallen op hashfuncties

hashlengte = *k* bits:

om (tweede) origineel te vinden: probeer willekeurige inputs,

aantal benodigd voor redelijke kans op een hit: 2^k

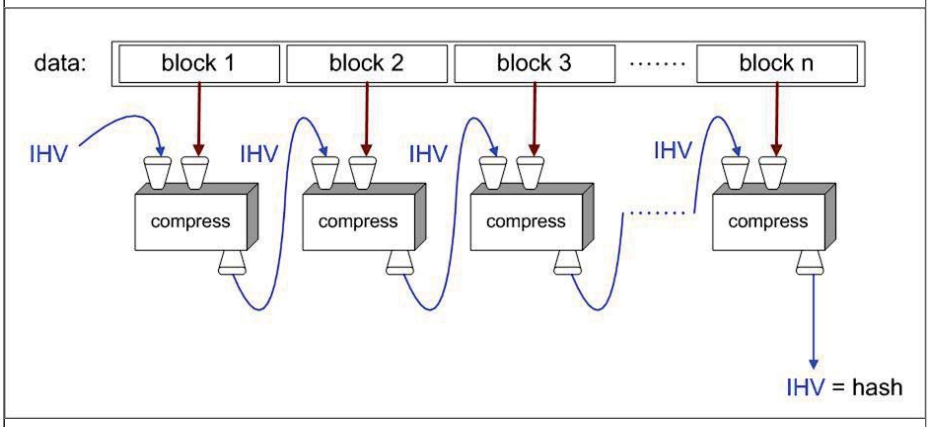
om botsing te vinden: probeer willekeurige inputs, vergelijk alle *paren*

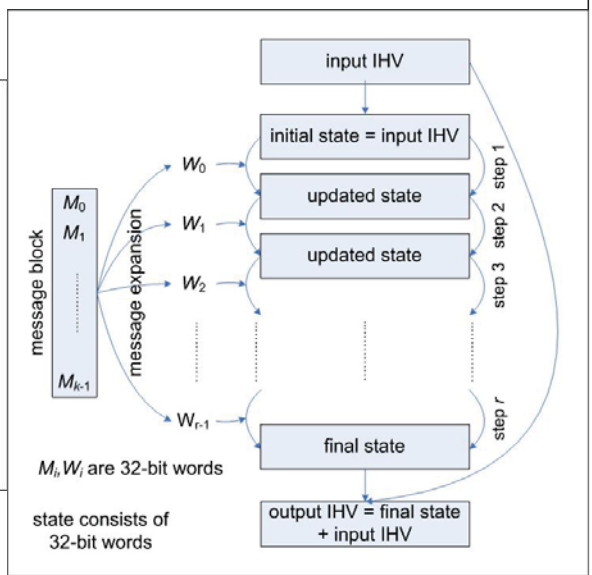
aantal benodigd voor redelijke kans op een hit: $2^{k/2}$

(verjaardagsparadox)

vandaar is hashlengte altijd *dubbele* van symmetrische sleutellengte

ontwerp SHA256 - Merkle-Damgård





ontwerp SHA256 - compressiefunctie

technieken lijken op ontwerp van symmetrische versleuteling

voor details: zie boekje

hashes als "bewijs van werk" (Bitcoin)

idee: een hashwaarde is zo goed mogelijk random

als je een hashwaarde met structuur ziet, is er net zolang gerekend tot die structuur een keer toevallig is opgedoken temidden van alle willekeur

bijvoorbeeld: probeer een input te vinden waarvan de hashwaarde de eerste 20 bits allemaal 0 heeft

- kans hierop: 1 op de 2^{20}
- verwachte aantal pogingen om dit te bereiken: 2^{20}

- de input is het bewijs:
  - m = `FRFLLCRHBXCYUOOXOTIETULOPMQFOEBTHUTYYJPPRSOJOCNHQQ`
- de hashwaarde verifieert het bewijs
  - SHA256(m) = `00000CC1 C71511D8 3641ED08 401A7603 D9FEB947-843F7DCB 96B4752C A269081B`