Asymmetrische Krypto

Grundprinzipien
Moderner Kryptologie

Symmetrische Krypto

Ein einziger Schlüssel zum
ver- und entschlüsseln

Klartext

Ver-/Entschlüsselung

Geheimtext
Chiffrat

Standard heute: AES
Sehr schnell, sicher.

Gute Krypto, schlechte Krypto

- Die Sicherheit eines Systems darf nur von der Sicherheit des Schlüssels abhängen
  - Und nicht von der Geheimhaltung des Verfahrens.
- Fazit: Nur offene Systeme können Sicherheit bieten

- Niemand kann alleine gute Krypto bauen

AES (symmetrisch)

- Advanced Encryption Standard
- Wird heute universell eingesetzt
- Schlank und schnell, läuft auf jeder Hardware

- Pferdefuss aller symmetrischen Verfahren:
  - Schlüsseltausch

Schlüsseltausch

- Wie über einen unsicheren Kanal einen geheimen Schlüssel tauschen?
- Physischer Kontakt

- Diffie-Hellmann Verfahren

Diffie-Hellmann Schlüsseltausch

Modulus n (prim)
Basis a (prim)
Zufallszahl x

$u = a^x \% n$

n, a, u

Zufallszahl y
$v = a^y \% n$

$s = u^y \% n$

v

$s = v^x \% n$

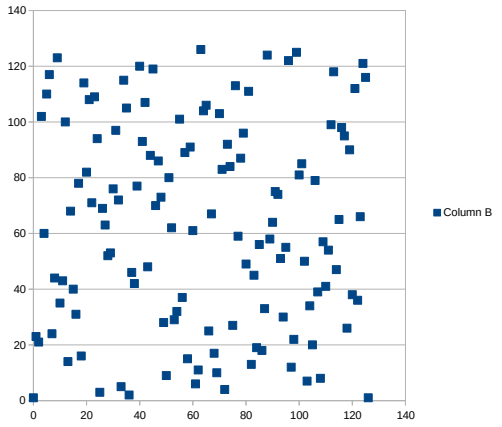
$$\left(a^x\right)^y = \left(a^y\right)^x = a^{x \cdot y} = s$$

Hochrechnen ist Einwegfunktion

- Diskreter Logarithmus praktisch nicht berechenbar

  (nach heutigem Kenntnisstand)

$$23^x \equiv 107 \; mod \; 127$$



- Typischerweise Grössenordnung von 2^{1000} bis 2^{4000}

1000-stellige Zahlen potenzieren?

$$a^{42} = a^2 \cdot a^8 \cdot a^{32}$$

$$a^4 = (a^2)^2, \qquad a^8 = (a^4)^2, \ldots$$

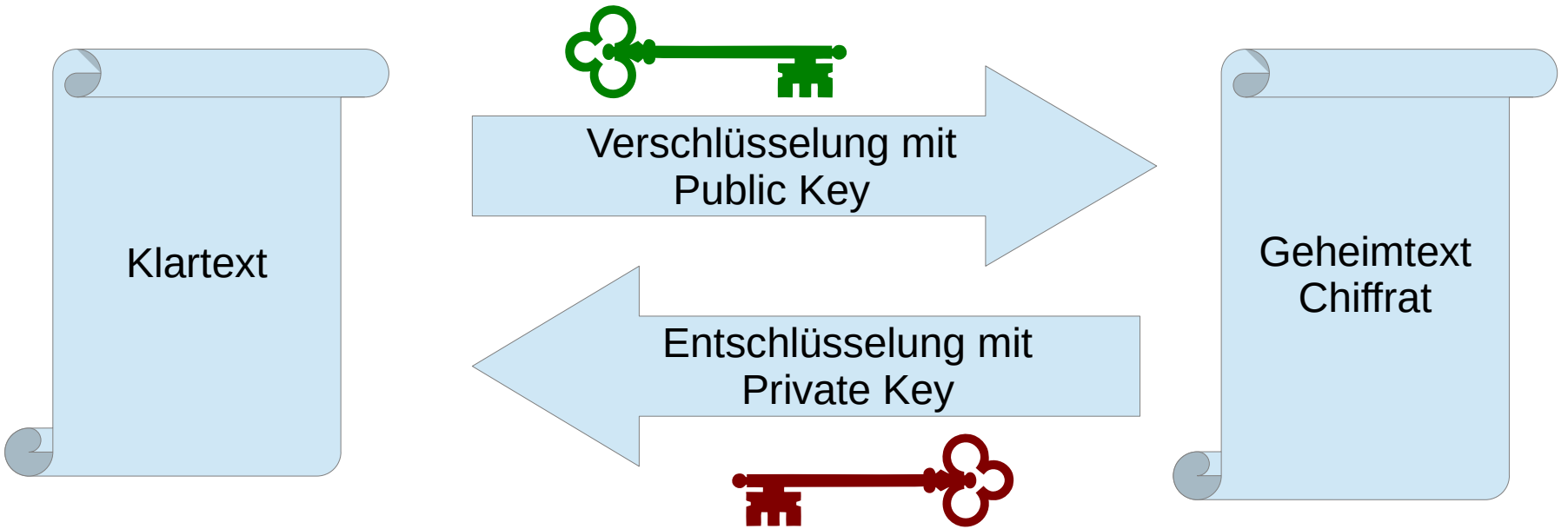
```
# Berechnet a**e % n
def modpow(a,e,n):
    prod = 1
    while e>0:
        if e%2==1:
            prod = (prod*a) % n
        a = (a*a) % n
        e = e//2
    return prod
```

Demo Schlüsseltausch

- Python-Code auf Wiki
- Basis und Modulus generieren
- Information öffentlich austauschen
- Geheimen Schlüssel vergleichen
- Hacker-Attacke fahren
  - Zufallszahlen grösser wählen...

Asymmetrische Krypto

- Schlüsselpaar (public, private)
- Was mit dem einen verschlüsselt wird,
  kann nur mit dem anderen entschlüsselt werden.



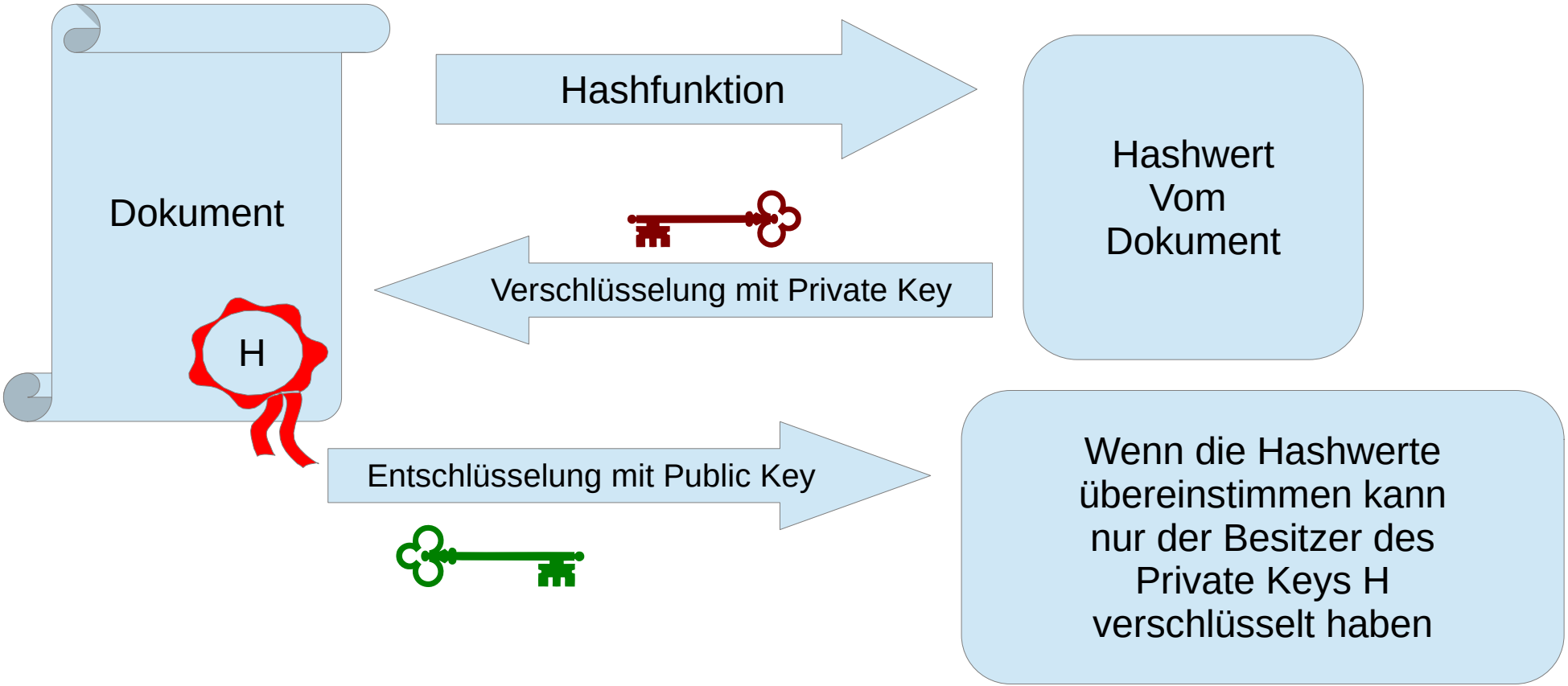
Bekannte Verfahren

- RSA (basiert auf Potenzieren modulo n)
  - lange Schlüssel nötig (>2048 Bits)
  - Ineffizient
  - Wird von Quantencomputern gebrochen werden
- ECC (basiert auf «elliptischen Kurven», Opertionen ebenfall modulo n)
  - Kürzere Schlüssel (>256 Bits)
  - Effizienter
  - Wird ebenfalls von Quantencomputern gebrochen werden

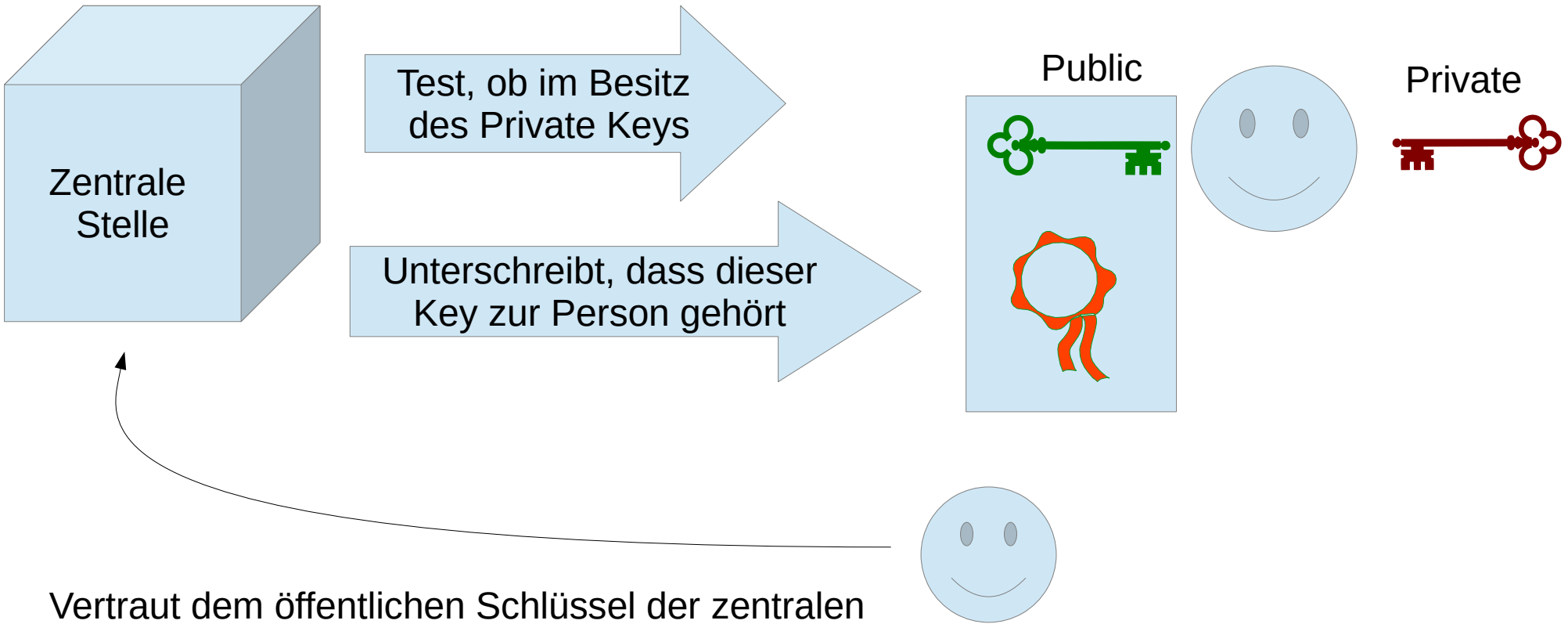
Anwendungen: Verschlüsselung

- Öffentliche Schlüssel werden publiziert
  - Wie ein Telefonbuch
- Nur der Besitzer des entsprechenden privaten Schlüssels kann die Nachricht entschlüsseln.
- Problem:
  - Woher weiss ich, dass der Schlüssel auch zur gewünschten Person gehört?

Anwendung: Digitale Unterschrift



Anwendung: Authentifizierung



Beispiel SwissID

- Schlüsselpaar wird auf Chip-Karte geliefert
- Public Key von anerkannter Stelle unterschrieben
- Keine Möglichkeit, den private Key selbst zu erzeugen

HTTPS

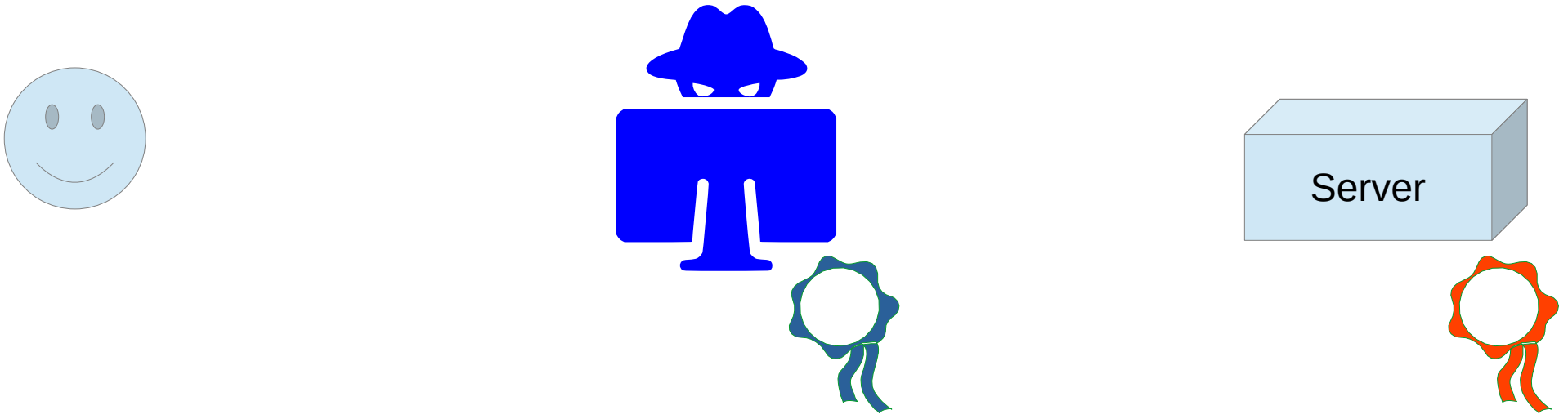
1. Browser fordert Zertifikat vom Server an
2. Browser überprüft Zertifikat
3. Browser und Server handeln Verschlüsselung aus
4. Sichere Verbindung steht
5. Erst jetzt werden Daten ausgetauscht

    URL, Passwörter, Seiteninhalt, etc.

Demo im Browser

- Zertifikate anschauen

Man in the middle attack



Überwachung an der Schule?

- Welche Zertifikate auf welchen Seiten?
- Google:
  - GlobalSign Root CA – R2
    - GTS CA 1O1, Google Trust Services
- Facebook:
  - DigiCert High Assurance EV Root CA

Threema, Signal, WhatsApp

- Private/Public Key
- Nachricht mit PublicKey verschlüsseln
  - Nur Empfänger kann entschlüsseln
- Threema, Signal können PublicKey via QR-Code prüfen

- Schwachpunkt ist Gerät → Staatstrojaner

Effektive Implementation

- Die Nachricht wird symmetrisch verschlüsselt
  - Mit zufälligem Schlüssel K
- K wird mit public Key verschlüsselt und der verschlüsselten Nachricht hinzugefügt.
- Vorteil: Mehrere Empfänger sind möglich
  - K wird für jeden mit dessen public Key einmal verschlüsselt

GPG / PGP

- GNU Pretty Good Privacy
- Tools-Suite für den Umgang mit Public/Private Keys
- Schlüsselerzeugung, Ver- und Entschlüsselung
- Digitale Unterschrift
- Beglaubigung andere Public Keys
  - Web of Trust
- Demo auf Teams, GPG-Operation auf einer Webseite

Block-Chains

Am Beispiel von BitCoin

Grundideen

- Keine zentrale Stelle/Kontrolle
- Kein Vertrauen zwischen Nutzern
- Zahlungsabwicklung

- Block-Chain: öffentliche Buchhaltung
- Fälschungssicher (Rechenaufwand)

Konten (Wallets)

- Kontonummer ist public Key
- Authorisierung von Transaktionen via private Key
  - Digitale Unterschrift
- Guthaben aus der Block-Chain ermittelbar (öffentlich!)
- Zuordnung Kontonummer ↔ Person nicht einfach
- Wer den private Key hat, kann über Guthaben verfügen
- Private Key weg, Guthaben futsch.

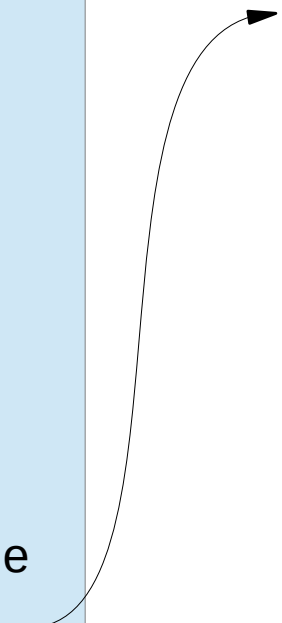
Block

Hashwert vom letzten Block

Transaktion 1
Transaktion 2
Transaktion 3
...
Transaktion n

Belohnung für Block

Zu findende Bytes, so dass die ersten x Bits vom Hashwert 0 ergeben. 2^x mal raten…



Hashwert vom letzten Block

Transaktion 1
Transaktion 2
Transaktion 3
...
Transaktion n

Belohnung für Block

Zu findende Bytes, so dass die ersten x Bits vom Hashwert 0 ergeben. 2^x mal raten…

Konsens-Findung

- Mining: neue Blöcke finden und publizieren
- Längste Kette ist massgebend
  - 2 neue Blöcke, welcher ist gültig?
  - An einem (wohl der erst-publizierte) wird weiter gearbeitet, am anderen nicht
- Es dauert einige Blöcke, bis eine Transaktion «sicher» eingebaut ist.

50%-Attacke

- Hat man mehr als 50% der Rechenkapazität kann eine eigene längste Kette berechnet werden
  - Etwas für BitCoin kaufen
  - Danach eine neue Kette ohne die Transaktion erzeugen
  - Enorme Rechenpower nötig

Warum Mining

- Belohnung für jeden neuen Block
  - Neue BitCoins werden erschaffen (aber immer weniger)
- Transaktionsgebühren
  - Transaktionen mit hohen Gebühren werden zuerst eingebaut.

BitCoin als Zahlungsmittel?

- Nur sehr wenige Transaktionen pro Sekunde (ca. 5)
- Hohe Transaktionsgebühren ($10 bis $20)
- Enormer Stromverbrauch (Grössenordnung Irland)
  - Spezielle Prozessoren nur fürs Mining (ASIC)
- Wert von BitCoin?
  - SpekulationsBlase?