Sichere Hashfunktionen

Prof. Dr.-Ing. Damian Weber

Hochschule für Technik und Wirtschaft des Saarlandes

Was tut eine Hashfunktion?

Hashfunktionen (Definition)

$h: U \longrightarrow V$

U = Universum, V = Hashwerte, $|V|<\infty$

Urbilder

Bilder

Hashfunktionen (Beispiel)

EAN (ISBN-13) 978047111709-4

- $U = \{ n \in \mathbf{N} \mid n<10^{12} \}$
- $V=\{0,1,2,3,4,5,6,7,8,9\}$ .... 10 Kisten

$9+3\cdot7+8+3\cdot0+4+3\cdot7+1+3\cdot1+1+3\cdot7+0+3\cdot9 = 116$

$10-6=4$ 978047111709 → 4

Hashfunktionen (Anwendung)

- Integritätschecks (Software, Dokumente)
- Digitale Signaturen
- Erzeugen kryptographischer Schlüssel

  aus Passwörtern
- Identifizieren von Malware

Integritätscheck

Gleiche Kiste, also Dokument gleich
(ungleiche mit 75% Chance in anderer Kiste)

Datenstrukturen: 10^8 Kisten

Krypto: $2^{256} \approx 10^{80}$ Kisten

Tools: sha1, sha256 (Linux sha1sum, sha256sum), Hashes of tar.gz

Digitale Signaturen

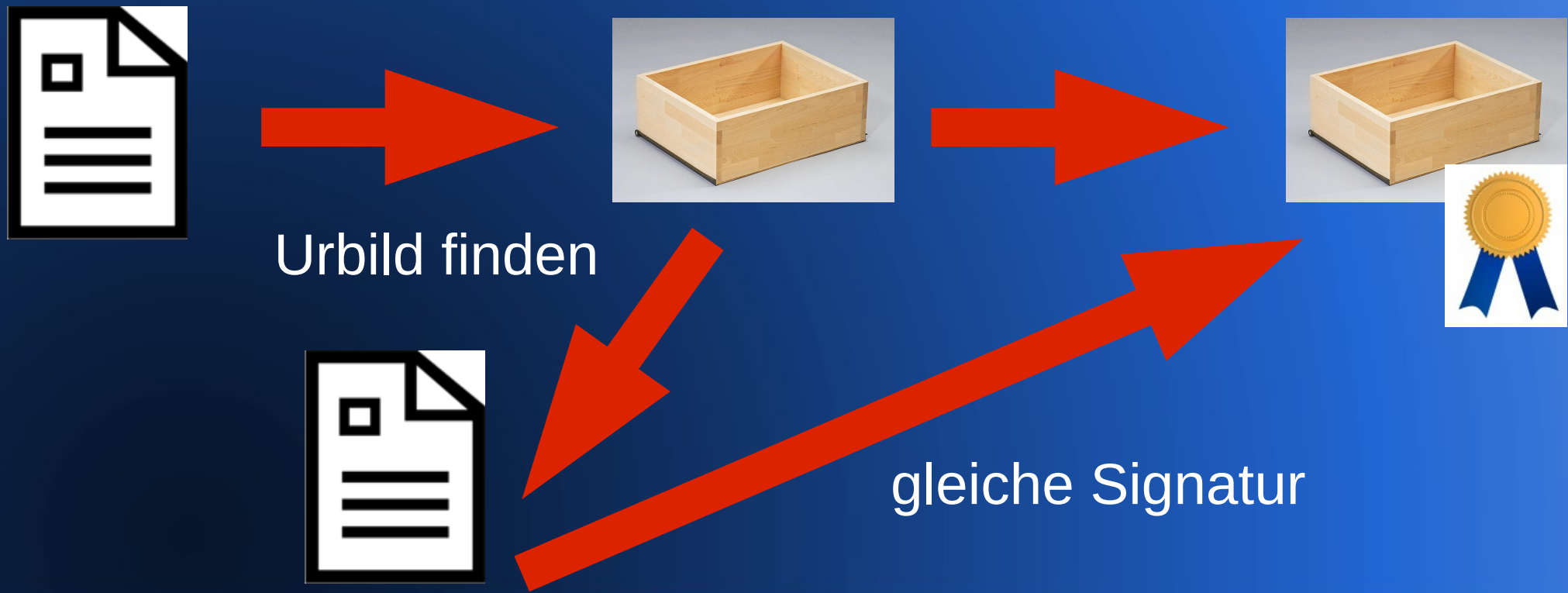
gnupg, SSL/TLS, ssh = Message authentication (Echtheit)

Digitale Signaturen
Gefahr: 2 Dokumente, gleiche Kiste

Kollision

Digitale Signaturen

Gefahr: 2. Dokument für diese Kiste künstlich erzeugen



Hashfunktionen (Anforderungen)

Sicher, wenn „unmöglich“:

- Urbild **(preimage-resistance)**
  (Anwendung: finde gültiges Passwort)
- 2. Urbild **(2nd preimage-resistance)**
  (Anwendung: finde 2. Dokument für gegebene Signatur)
- Kollision **(collision resistance)**
  (Anwendung: finde 2 Dokumente mit gleicher Signatur)

Hashfunktionen (Urbild)

Wir knacken die Urbildeigenschaft der EAN:

finde eine EAN mit Prüfziffer 7

$a+3b+c+3d+e+3f+g+3h+i+3j+k+3l$

muss als Endziffer 7 haben

Hashfunktionen (2. Urbild)

Wir knacken die 2. Urbildeigenschaft der EAN:

Kenne EAN mit Prüfziffer 4:

978047111709-4

Finde noch eine mit Prüfziffer 4
a+3b+c+3d+e+3f+g+3h+i+3j+k+3l

a und c tauschen ändert nichts am Ergebnis:
879047111709-4

Hashfunktionen (Kollision)

Kollision finden leicht, wenn man 2. Urbild kann

Wichtige Attacke: viele Hashes zufällig erzeugen

prüfen, ob Hashwert schon einmal gesehen

```
978-0312979478 And then there were none
978-0062073563 Murder of Roger Ackroyd
978-0062073495 Murder on the Orient Express
978-0573619236 The Mousetrap
978-0002315968 Miss Marple Final Cases
```

Kollision mit Hashwert 8 gefunden

Hashfunktionen (Anforderungen)

Effizienz

- Anwendungen mit vielen Signaturen
- Signieren von großen Dokumenten
- Signieren von Software

Sicherheit gegen
- Urbildsuche
- Kollisionen

Problematik Urbildsuche

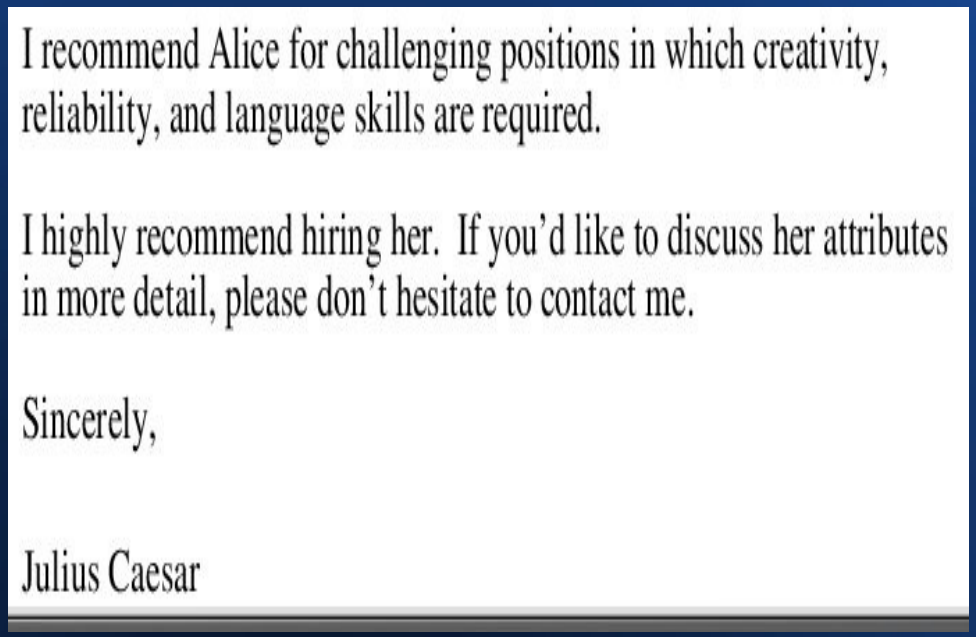
- MD5-Hashwerte von Passwörtern in `/etc/passwd`
- `1234` `81dc9bdb52d04dc20036dbd8313ed055`
- `hallo` `598d4c200461b81522a3328565c25f7c`
- `secret` `5ebe2294ecd0e0f08eab7690d2a6ee69`

- wer aus `5ebe2294ecd0e0f08eab7690d2a6ee69`
  den String „secret“ errechnen kann, kann Passwörter knacken

MD5 ist bei Kollisionen gebrochen, aber nicht bei Urbildsuche

Problematik Kollision

- zwei Dokumente mit gleichem Hashwert

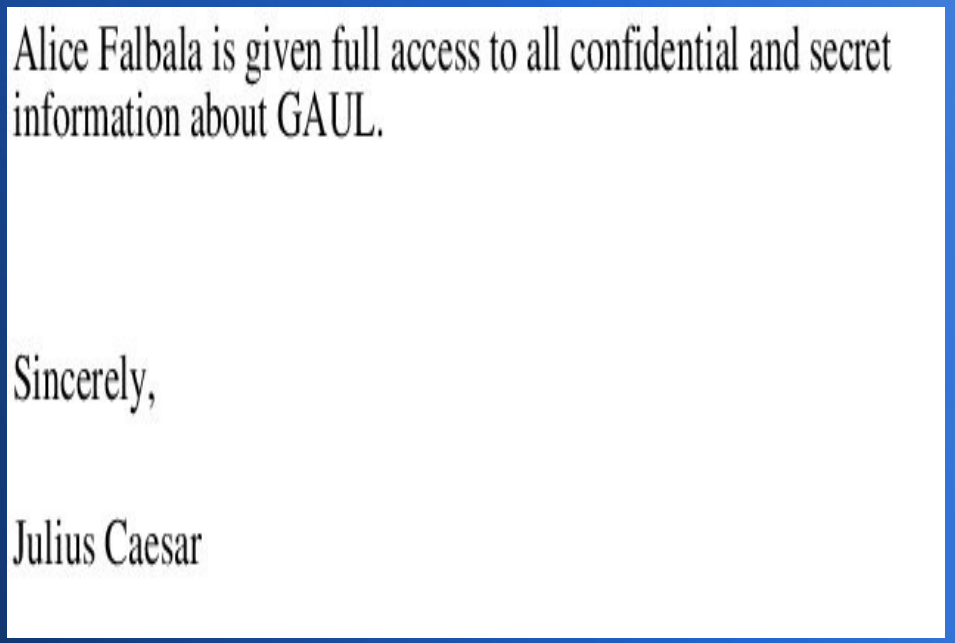


I recommend Alice for challenging positions in which creativity, reliability, and language skills are required.

I highly recommend hiring her. If you'd like to discuss her attributes in more detail, please don't hesitate to contact me.

Sincerely,

Julius Caesar



Alice Falbala is given full access to all confidential and secret information about GAUL.

Sincerely,

Julius Caesar

`MD5: a25f7f0b29ee0b3968c860738533a4b9`

`MD5: a25f7f0b29ee0b3968c860738533a4b9`

(Beispiel von Stefan Lucks, Bauhaus-Univ. Weimar)

Kollision: Mindestanforderung

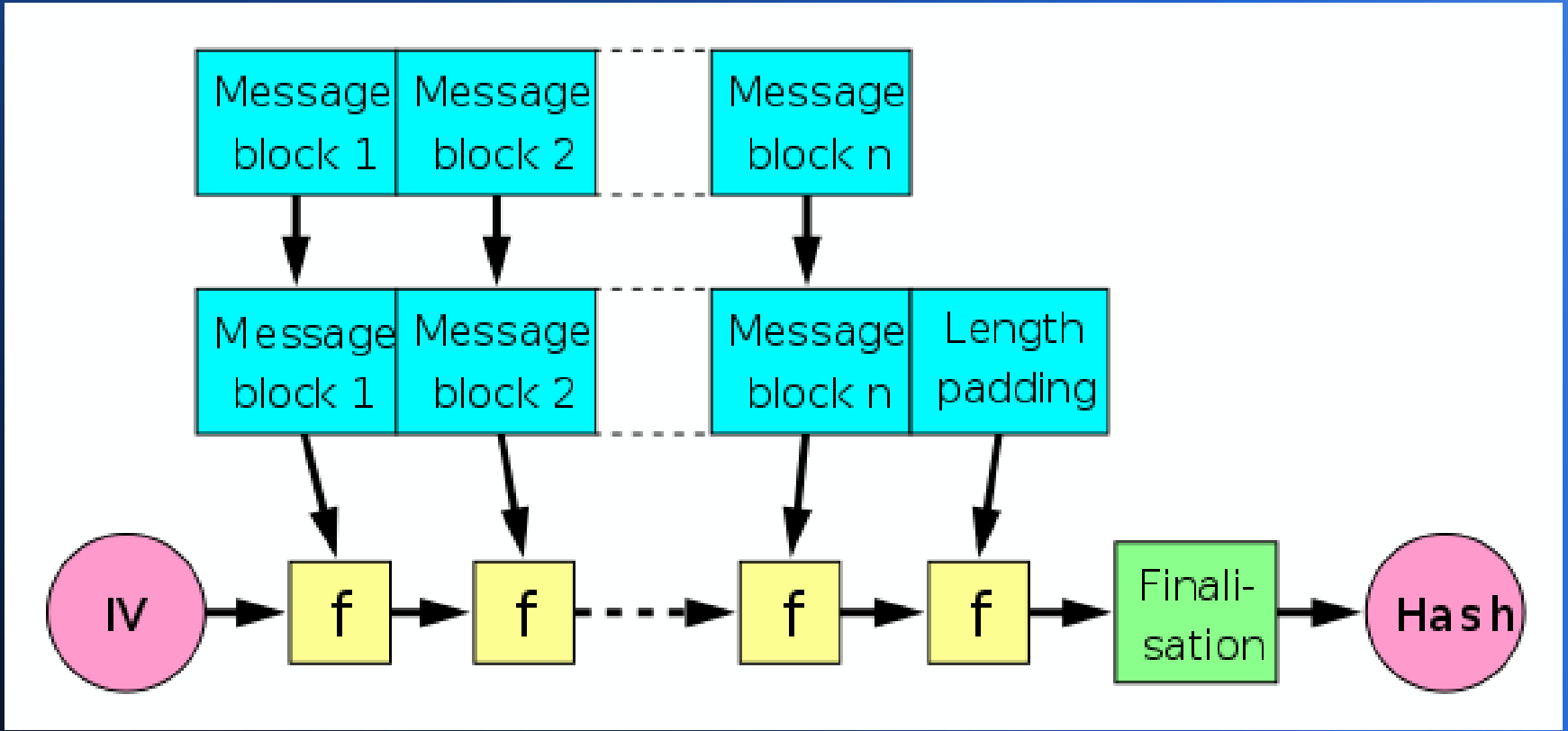
- robust gegen Pollard-λ Attacke mit O(sqrt(n)) Laufzeit
  - Bildbereich der Hashfunktion > 2^{160}
  - daher neue Hashfunktionen mit 256 Bits

Sichere Hashfunktionen

aktuelle Situation:

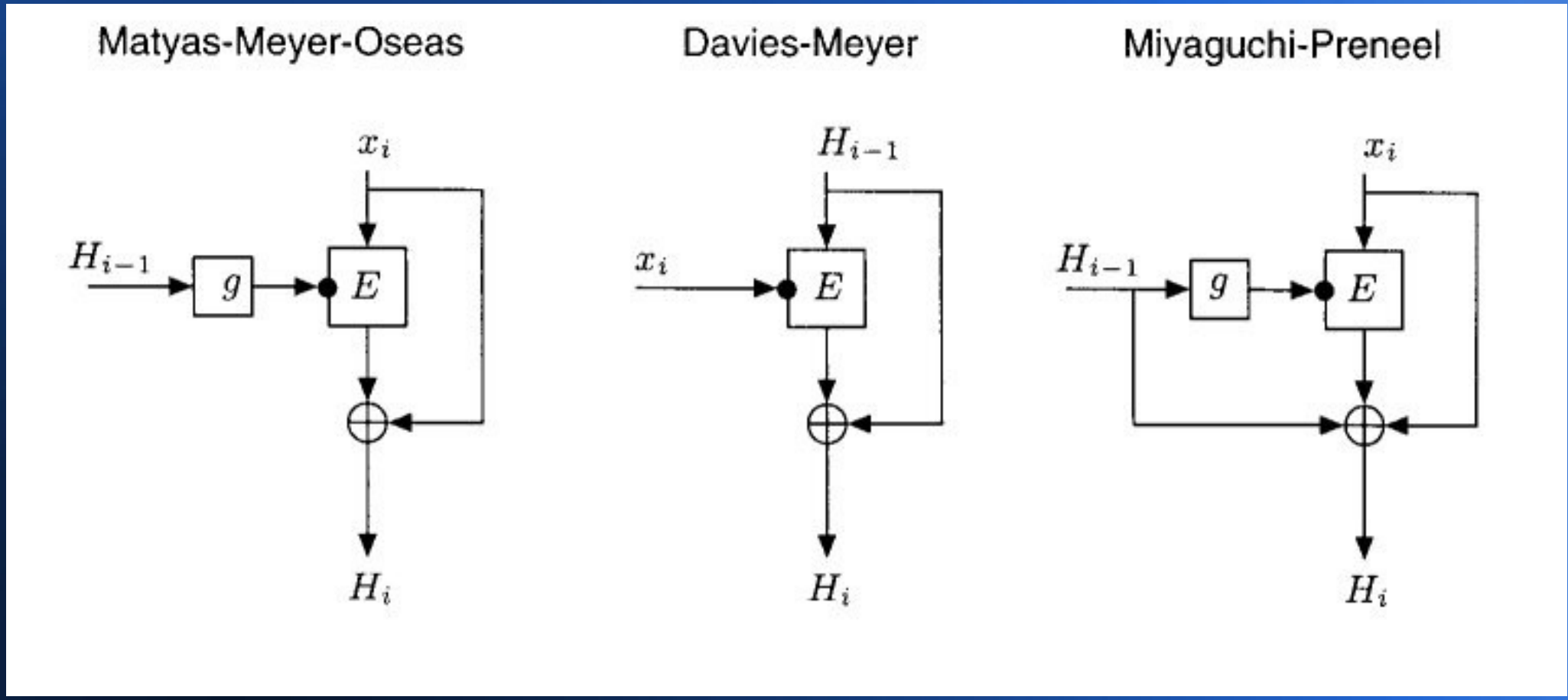
- schwache Algorithmen
  MD5 (Rivest 1991), Kollisionen in Minuten
  SHA-1 (NIST 1994) (Kollisionen nach 2^{70} Op.)
- Interimslösung SHA-256 (NIST 2002)
- SHA-3 Wettbewerb von NIST 2012

Merkle-Damgard-Construction



https://en.wikipedia.org/wiki/Image:Merkle-damgard.png

Merkle-Damgard-Construction



Handbook of Applied Cryptography (Menezes et al)

Hashwettbewerb NIST

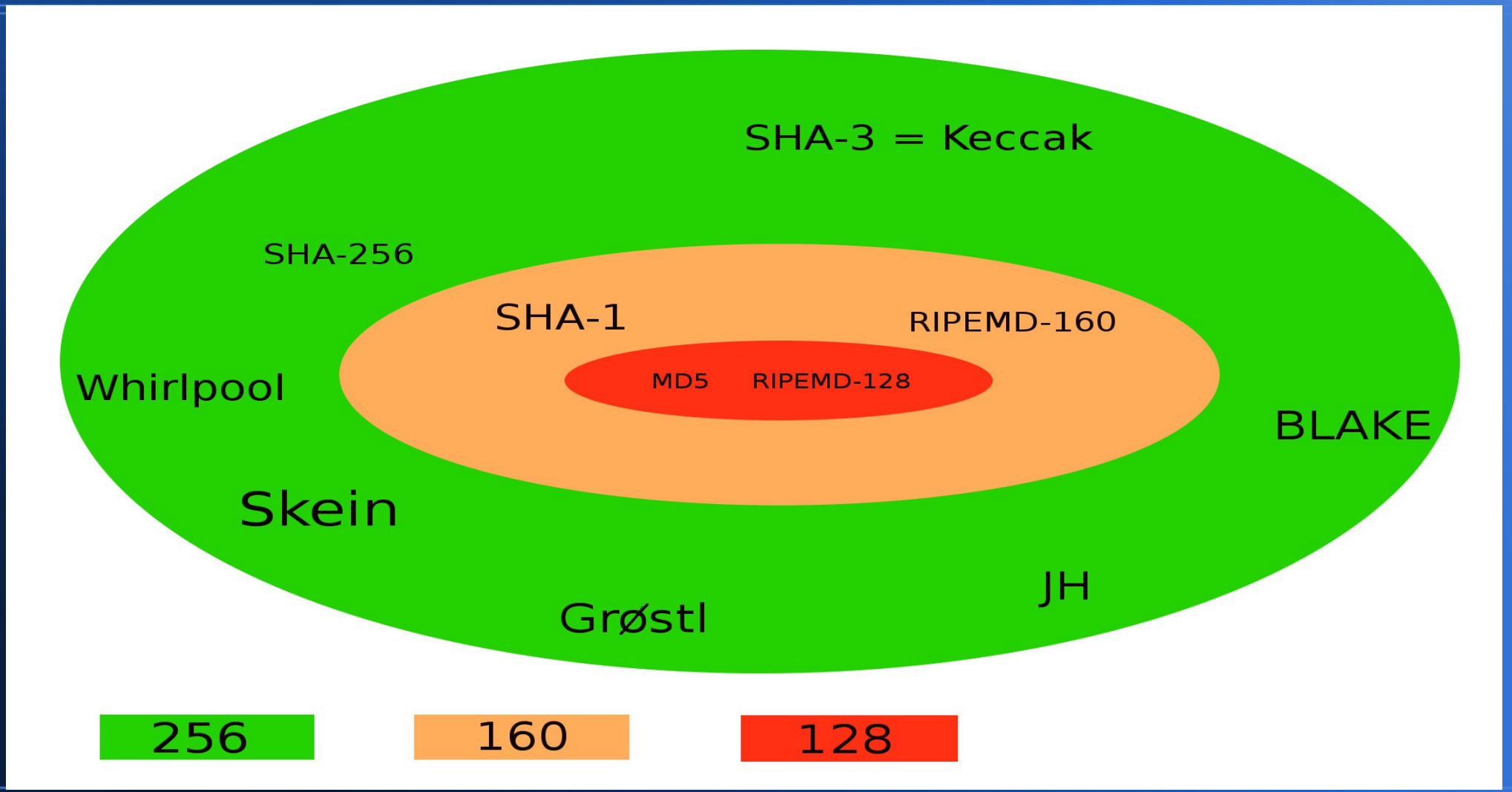
SHA-3 Hashwettbewerb

Runde 1: Nov 2008
Runde 2: Juli 2009
Runde 3: Dez 2010

5 Finalisten
`Blake, Grøstl, JH, Keccak, Skein`

Auswahl Okt. 2012 **Keccak**

Empfehlungen Hashfunktionen:
Wertebereich mindestens 256 Bit



SHA-3-Finalisten

(im Wettbewerb intensiv geprüft worden)

Hash	Authors	Rounds (256 hash)
Keccak	Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche	24
Skein	Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, Jesse Walke	72
Blake	Jean-Philippe Aumasson, Luca Henzen, Willi Meier, Raphael C.-W. Phan	14
JH	Hongjun Wu	42
Grøstl	Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, Søren S. Thomsen	10

Kryptoeigenschaften der Finalisten

	BLAKE	Grøstl	JH	Keccak	Skein
Design:					
Merkle-Damgård	X	X			X
Sponge			X	X	
Compression function:					
Not block cipher based		X			
Block cipher based, Davies-Meyer	X				
Block cipher based, Matyas-Meyer-Oseas					X
Uses cipher pieces:					
AES		X			
ChaCha	X				
Threefish					X

https://www.cs.rit.edu/~ark/fall2012/482/module11/notes.shtml

Hashfunktionen (Kollisionssuche)

Gesucht: x,y mit h(x)=h(y)

Generische Attacke:

- x1=h(x0)
- x2=h(x1)
- x3=h(x2)
- ...
- bis zwei x-Werte übereinstimmen, z.B.

x27 und x12

dann ist h(x26)=h(x11)

Hashfunktionen (Kollisionssuche)

Algorithmische Fragen:

- Wie lange dauert das?
- Wie erkennt man die Kollision?

Hashfunktionen (Kollision - wie lange dauert das?)

$$P(k) = \left(\frac{n-1}{n}\right)\left(\frac{n-2}{n}\right)\cdots\left(\frac{n-k+1}{n}\right) = \prod_{j=1}^{k-1}\left(1-\frac{j}{n}\right)$$

$$k \geq \frac{1}{2} + \frac{1}{2}\sqrt{1+8\log(2)n}$$

Hashfunktionen (Kollision - wie lange dauert das?)

n = Größe Bildraum der Hashfunktion

MD5: 128 Bits, $n=2^{128}$, Kollision in 2^{64} Schritten

SHA-1: 160 Bits, $n=2^{160}$, Kollision in 2^{80} Schritten

Analyse von SHA-1: Verbesserung auf 2^{70} Schritte

Hashfunktionen (wie die Kollision erkennen)

Naive Methode:

alles abspeichern,

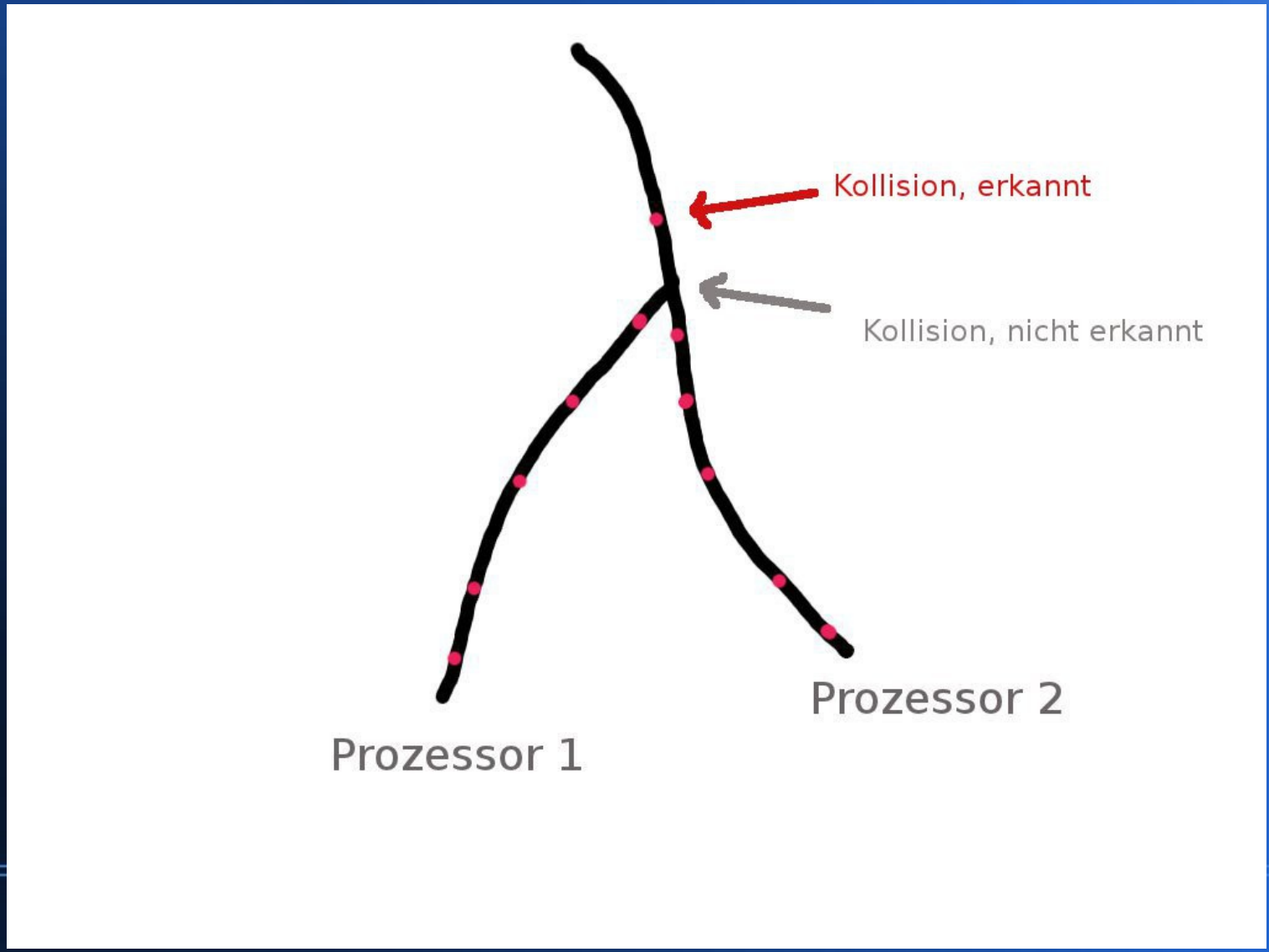
jedesmal testen,
ob bereits gesehen

Hashfunktionen (Kollision hier?)

```
356a192b7913b04c54574d18c28d46e6395428ab
da4b9237bacccdf19c0760cab7aec4a8359010b0
77de68daecd823babbb58edb1c8e14d7106e83bb
1b6453892473a467d07372d45eb05abc2031647a
ac3478d69a3c81fa62e60f5c3696165a4e5e6ac4
c1dfd96eea8cc2b62785275bca38ac261256e278
902ba3cda1883801594b6e1b452790cc53948fda
fe5dbbcea5ce7e2988b8c69bcfdfde8904aabc1f
0ade7c2cf97f75d009975f4d720d1fa6c19f4897
b1d5781111d84f7b3fe45a0852e59758cd7a87e5
17ba0791499db908433b80f37c5fbc89b870084b
7b52009b64fd0a2a49e6d8a939753077792b0554
bd307a3ec329e10a2cff8fb87480823da114f8f4
fa35e192121eabf3dabf9f5ea6abdbcbc107ac3b
f1abd670358e036c31296e66b3b66c382ac00812
1574bddb75c78a6fd2251d61e2993b5146201319
0716d9708d321ffb6a00818614779e779925365c
9e6a55b6b4563e652a23be9d623ca5055c356940
b3f0c7f6bb763af1be91d9e74eabfeb199dc1f1f
91032ad7bbcb6cf72875e8e8207dcfba80173f7c
```

Problem: Speicherplatz (2^{80} Hashwerte)
2417851639229258 3 GB

Pollard-λ

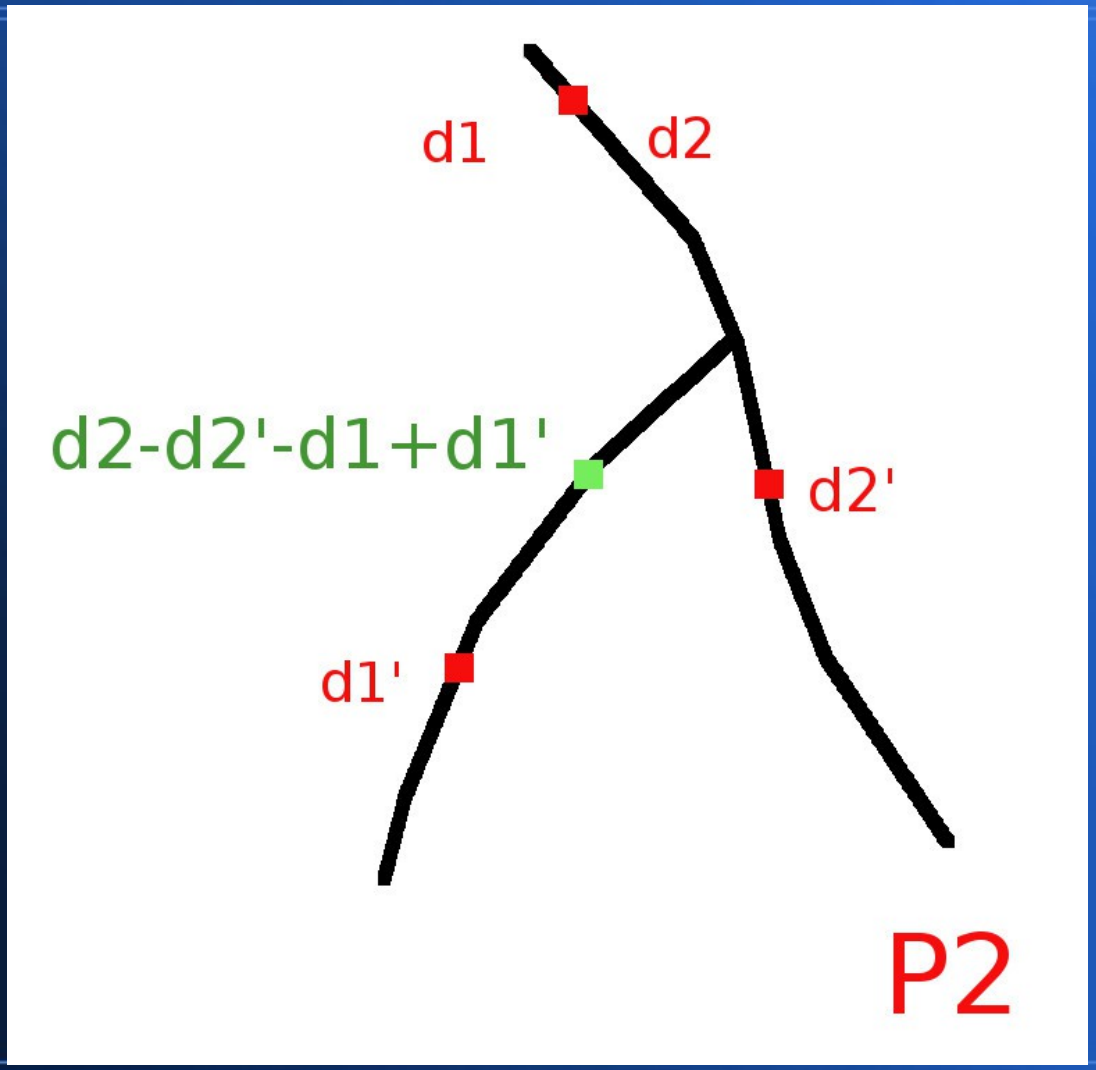

Kollision, erkannt
Kollision, nicht erkannt
Prozessor 2
Prozessor 1

Kollisionspunkt erkennen (1)

- Logarithmen.Lösung leicht: *x=(k2-k1)/(l1-l2)*
- Hashwerte kann man aber nicht dividieren ...
- Server meldet distinguished point *y* für P1, P2
  an Stellen d_1 (auf P_1) und d_2 (auf P_2)
- vorherige distinguished points
  an Stellen d_1' (auf P_1), d_2' (auf P_2)
- der Kollisionspunkt hat auf beiden Prozessoren den gleichen Abstand von d_1, d_2

ein Bild erklärt mehr als tausend Worte ...

Kollisionspunkt erkennen



Kollisionspunkt erkennen (2)

- wir setzen P_1, P_2 zurück auf ihren vorherigen DP
- der mit dem größeren Abstand (z.B. P_1)

  verringert seine Distanz zu *y*, bis Distanz gleich mit P_2
- also *d1-d1'+i = d2-d2'*, daher hat P_1

  i = d2-d2'-d1+d1' Schritte zu tun
- danach beide gleichzeitig solange 1 Schritt, bis Gleichheit

Message Authentication Code

MAC

=

- Hashfunktion

+

- geheimer Schlüssel K

$H_K(x)$

Erster Versuch

- $H_K(x) = H(K,x)$
- Attacke: K raten bis gleiche Hashwerte gefunden
- Laufzeit: sqrt(Schlüsselraum)

Zweiter Versuch

- $H_K(x) = H(x,K)$
- Attacke: Kollisionen für x, x'
- dann kann man jeden Key dahinterhängen
- Problematisch: kollisionsanfällige Hashfunktionen

Allgemein akzeptiert: HMAC

- unabhängig vom Aufbau der Hashfunktion
- unabhängig von der Kollisionsresistenz
- Idee: Hash zweimal ausführen
- RFC 2104

$$HMAC(K, m) = H\Big((K \oplus opad) \mid\mid H\big((K \oplus ipad) \mid\mid m\big)\Big)$$

Analogie zu Signaturen

- MAC erzeugt eine Signatur mit Mitteln der symmetrischen Kryptographie

- zwei Partner haben authentisierte Messages , falls vorher geheimer Schlüssel ausgetauscht

Authentisieren und Verschlüsseln

- Authenticate then Encrypt (AtE) – SSL
- Encrypt then Authenticate (EtA) – IPSec
- Encrypt and Authenticate (E&A) – SSH
- allgemein empfohlen: EtA

(weniger Attacken, weniger Implementierungsfallen)

Neue Verschlüsselungsmodi „Authenticated Encryption“

- GCM: Galois Counter Mode
  (McGrew, Vieda, 2004)
- CCM: Counter with CBC-MAC
  (Housley/Whiting/Ferguson, 2002)
- EAX:
  (Bellare/Rogaway/Wagner, 2003)

Nutzung z.Zt. noch experimentell