Kapitel 9:
Kryptographische Hash-Funktionen

Meltdown und Spectre

- Woche ab 29.12.17: Hardware-Bug in Intel-Prozessoren
  - CVE-2017-5715: branch target injection (Spectre); CVSS3: 6,7
  - CVE-2017-5753: bounds check bypass (Spectre): CVSS3: 8,7
  - CVE-2017-5754: rogue data cache load (Meltdown): CVSS3: 7,9
- Weitere Prozessoren und Hersteller betroffen
  - Intel
  - ARM (Cortex-A75)
  - ARM Lizenz: z.B. Samsung, Qualcomm (Smartphones)
  - Power
  - S390
- Erlaubt das Auslesen von Hauptspeicherinhalten auch fremder Prozesse und aus dem Kernel-Space
  - Passwörter
  - Daten
  - Sensible Informationen

Meltdown Spectre: gunds. Funtkionsweise

- Moderne Prozessoren nutzen Pipelining, speculative execution und branch prediction um Performance zu verbessern
  - Prozessor überprüft Zugriffsrechte auf Daten (user, kernel space)
  - Speculative execution und branch prediction:
    - CPU führt Code aus, unter den Annahme das best. Bedingung wahr sind.
    - Falls sich diese später als falsch herausstellen -> Rollback
    - Richtiger Ausführungspfad wird ausgeführt
  - Daten werden „vorab“ in den CPU Cache geladen
  - Angriffe nutzen Race-Condition um Cache (ohne Rechte) auszulesen
  - Mehrfach hintereinander anwendbar, um Cache-Häppchen auszulesen, ca. 1.500 Byte pro Sekunde
  - Damit kann der gesamte Hauptspeicher ausgelesen werden

Meltdown: CVE-2017-5754

- Rogue data cache load
- User Mode Prozess kann Speicher auslesen, wie wenn er sich im Kernel Mode befinden würde

- Gegenmaßnahmen:
  - Patch des Betriebssystems
  - Linux: Kernel Page Table Isolation (KPTI) Patch
  - Apple MacOS und iOS: Mitigations in iOS 11.2, macOS 10.13.2 und tvOS 11.2. watchOS ist nicht betroffen
  - Windows: Patch verfügbar, allerdings Probleme mit Antivieren-Software (z.B. Sophos) -> Bluescreen
  - Android: Patch 2018-01-05

Spectre: CVE-2017-5753 und -5715

- CVE-2017-5753: Bounds check bypass
- Speculative code execution
  - Bereichsprüfung für Speicherbereiche (z.B. Arrays) kann umgangen werden
  - Damit Zugriff außerhalb der Speichergrenzen möglich
- Weiteres Problem: Just in time compiler (JIT) und in-kernel bytecode interpreter
  - z.B. extended Berkely Packet Filter (eBPF)
  - Packet-Filter Regeln aus dem User-Space
  - JIT Compiler erzeugt daraus Maschinencode der im Kernel-Mode läuft
- Mitigation:
  - Analyse und recompilation von code
  - Patch von Betriebssystemen
  - Patch von Anwendungen die nicht-vertrauenswürdigen Code ausführen (JIT, Laufzeitumgebungen, Skript-Interpreter, etc.)

Spectre: CVE-2017-5753 und -5715

- CVE-2017-5715: Branch target injection
  - Prozess kann speculative execution eines anderen Prozesses beeinflussen
  - Auf demselben CPU-Core
  - In einem anderen Sicherheitskontext
  - Tabellen für speculative execution werden zwischen allen Prozessen auf einem CPU-Core geteilt
- Damit Mapping von beliebigem Speicherbereich (Code) in anderen Prozess, in den Hypervisor oder in den Kernel möglich (!!!!!!!!!!)
- Schwer zu nutzen aber riesiges Potential da alle Sicherheitsbereiche überschritten werden können
- Problem für Cloud-Betreiber:
  - VM kann andere VMs oder Hypervisor beeinflussen

Spectre: CVE-2017-5715: Mitigation

- CVE-2017-5715: Branch target injection
- Mitigation
  - Neue Hardware (CPU)
  - CPU Microcode Update vom Hersteller (z.B. Intel IBRS Microcode)
  - Software-Mitigation (z.B. Google's Retpoline) für
    - Hypervisor
    - Betriebssystem-Kernel
    - Systemprogramme
    - Bibliotheken
    - Anwendungen im Userspace
- Hersteller:
  - Browser Updates für Chrome, Edge, Firefox, Safari
  - Virtualisierungsplattformen: Update, ggf. mit Patch der Gast-OS

Meltdown und Spectre: Quellen (2)

- https://meltdownattack.com/
- Meltdown Paper https://meltdownattack.com/meltdown.pdf
- Spectre Paper: https://spectreattack.com/spectre.pdf

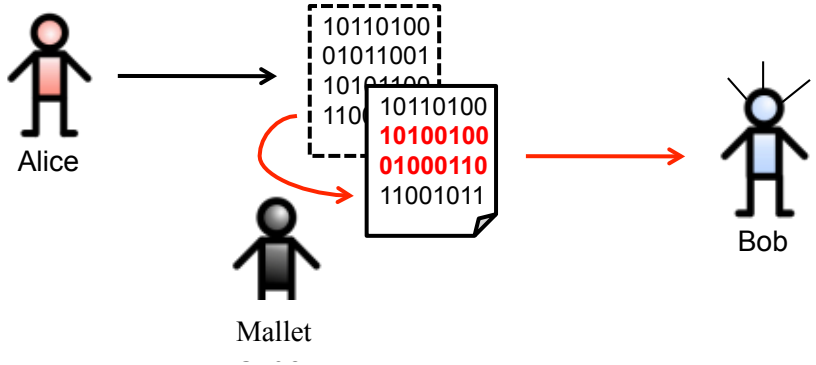
- Goole Project Zero
  - https://googleprojectzero.blogspot.de/2018/01/reading-privileged-memory-with-side.html
  - https://security.googleblog.com/2018/01/more-details-about-mitigations-for-cpu_4.html
  - https://support.google.com/faqs/answer/7625886

Inhalt

- Definition: Kryptographische Hash-Verfahren
- Angriffe gegen One-Way-Hash-Funktionen
- Konstruktion von Hash-Funktionen
- Algorithmen:
  - MD5
  - SHA-3 (Keccak)

Hash-Funktionen zur Integritätssicherung

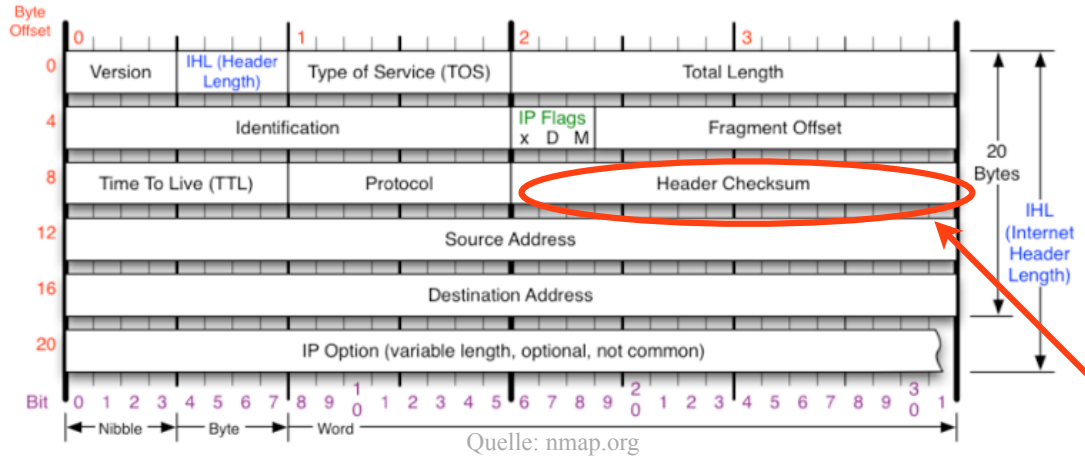
- Ziel: Sicherstellen, dass Manipulationen an einer übertragenen Nachricht erkannt werden.



- Beispiel Software-Distribution:

Herkömmliche vs. kryptographische Hash-Funktionen

- Prüfsummen dienen der Erkennung von (unbeabsichtigten) Übertragungsfehlern, z.B. beim IPv4-Header:



Quelle: nmap.org

Version

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.

Header Length

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

Protocol

IP Protocol ID. Including (but not limited to):

1 ICMP	17 UDP	57 SKIP
2 IGMP	47 GRE	88 EIGRP
6 TCP	50 ESP	89 OSPF
9 IGRP	51 AH	115 L2TP

Total Length

Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.

Fragment Offset

Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.

Header Checksum

Checksum of entire IP header

IP Flags

x D M

x 0x80 reserved (evil bit)
D 0x40 Do Not Fragment
M 0x20 More Fragments follow

RFC 791

Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.

16-bit Addition / Einerkomplement

- Kryptographische Prüfsummen sollen auch absichtliche Manipulationen erschweren

Kryptographische Hash-Funktionen: Grundlagen

- Hash-Funktionen
  - bilden „Universum“ auf endlichen Bildbereich ab
  - sind nicht injektiv
  - Bildbereich i.d.R. sehr viel kleiner als Universum
  - Kollisionen möglich:

$$\exists x, y \in U : x \neq y \quad \wedge \quad h(x) = h(y)$$

- Kryptographische Hash-Funktion H:
  - Eingabe: beliebig langes Wort m aus dem Universum U
  - Ausgabe: Hashwert H(m) mit fester Länge
  - H soll möglichst kollisionsresistent sein

Beispiel

- MD5-Hashwerte sind immer 128 Bits lang
  - egal, wie lange die Eingabe ist

```
829c11ba6dcdfe045dd1e5a77b34c05e  OOo-SDK_3.2.1_Linux_x86-64_install-deb_en-US.tar.gz
0f8abee370438e49e7ea0c2287589760  OOo-SDK_3.2.1_Linux_x86-64_install-rpm_en-US.tar.gz
35e8406c95c58b0087b9ad964faa13b8  OOo-SDK_3.2.1_Linux_x86_install-deb_en-US.tar.gz
ecc8271619ad788203cc61c7d9930522  OOo-SDK_3.2.1_Linux_x86_install-rpm_en-US.tar.gz
dddab486fd466bb1fc1a126d75919a3f  OOo-SDK_3.2.1_MacOS_x86_install_en-US.dmg
c4eb9f161736ba64933bdc81c274d8bc  OOo-SDK_3.2.1_Solaris_Sparc_install_en-US.tar.gz
3e8c88a645a8706e6c1bf7ef103bb993  OOo-SDK_3.2.1_Solaris_x86_install_en-US.tar.gz
7a7f4ea9173f9b8d9ae71cdf65c328f0  OOo-SDK_3.2.1_Win_x86_install_en-US.exe
```

- Weil es nur 2^{128} verschiedene MD5-Hashwerte gibt, existieren beliebig viele Dateien mit demselben MD5-Hashwert
  - = Kollision
- Zwei sehr ähnliche, aber nicht identische Eingaben sollen nicht denselben MD5-Hashwert haben
  - = Kollisionsresistenz
- Angreifer versucht, die Nachricht m „sinnvoll“ in m‘ abzuändern, so dass md5(m) = md5(m‘)

Def. Kryptographische Hashfunktion

Schwache Hash-Funktion H:

- H besitzt die Eigenschaften einer Einwegfunktion
- Hashwert H(m) = h mit |h|=k (z.B. k = 128 Bits) ist bei gegebener Nachricht m einfach zu berechnen
- Bei gegebenem h = H(m) für $m \in A_1^*$ ist es praktisch unmöglich, eine (sinnvolle) m‘ zu finden mit:

$$m' \neq m,\ m' \in A_1^* \quad \wedge \quad H(m') = h$$

Starke Hash-Funktion H:

- H hat alle Eigenschaften einer schwachen Hash-Funktion
- Es ist zusätzlich praktisch unmöglich, eine Kollision zu finden, d.h. ein Paar verschiedene Eingabewerte m und m‘ mit:

$$m' \neq m,\ m, m' \in A_1^* \quad \wedge \quad H(m) = H(m')$$

Birthday Attack auf One-Way-Hash-Funktionen

- Wie viele Personen brauchen Sie, damit mit Wahrscheinlichkeit P > 0,5 eine weitere Person mit Ihnen Geburtstag hat?
  - Antwort: 253 $P = 1 - \left(1 - \frac{1}{365}\right)^n$ (ab n=253 ist P > 0,5)
- Wie viele Personen brauchen Sie, damit mit Wahrscheinlichkeit P > 0,5 zwei Personen am selben Tag Geburtstag haben?
  - Antwort: 23 $P = 1 - \frac{365 \cdot 364 \cdots (365 - (n-1))}{365^n}$ (ab n=23 ist P > 0,5)
- Wie können Sie dieses Wissen für Angriffe gegen Hash-Funktionen nutzen?

Eine Kollision zu finden ist deutlich einfacher
als zu einem gegebenen Hash-Wert einen passenden Text!

Birthday Attack: Vorgehensweise

1. Alice sichert mit einem k Bits langen Hash eine Nachricht M
2. Mallet erzeugt $2^{k/2}$ Variationen der Nachricht M

- Die Wahrscheinlichkeit für eine Kollision ist größer 0,5.

- Wie können $2^{k/2}$ Variationen erzeugt werden?
  - Z.B. Einfügen von „Space – Backspace – Space“ Zeichen zwischen Wörtern
  - Wörter durch Synonyme ersetzen
  - .....

Beispiel für einen Brief mit 2^37 Variationen

- [Stal 98]

Dear Anthony,

{This letter is / I am writing} to introduce {you to / to you} {Mr. / --} Alfred {P. / --}

Barton, the {new / newly appointed} {chief / senior} jewellery buyer for {our / the}

Northern {European / Europe} {area / division}. He {will take / has taken} over {the / --}

responsibility for {all / the whole of} our interests in {watches and jewellery / jewellery and watches}

in the {area / region}. Please {afford / give} him {every / all the} help he {may need / needs}

to {seek out / find} the most {modern / up to date} lines for the {top / high} end of the

market. He is {empowered / authorized} to receive on our behalf {samples / specimens} of the

{latest / newest} {watch and jewellery / jewellery and watch} products, {up / subject} to a {limit / maximum}

of ten thousand dollars. He will {carry / hold} a signed copy of this {letter / document}

as proof of identity. An order with his signature, which is {appended / attached}

{authorizes / allows} you to charge the cost to this company at the {above / head office}

address. We {fully / --} expect that our {level / volume} of orders will increase in

the {following / next} year and {trust / hope} that the new appointment will {be / prove}

{advantageous / an advantage} to both our companies.

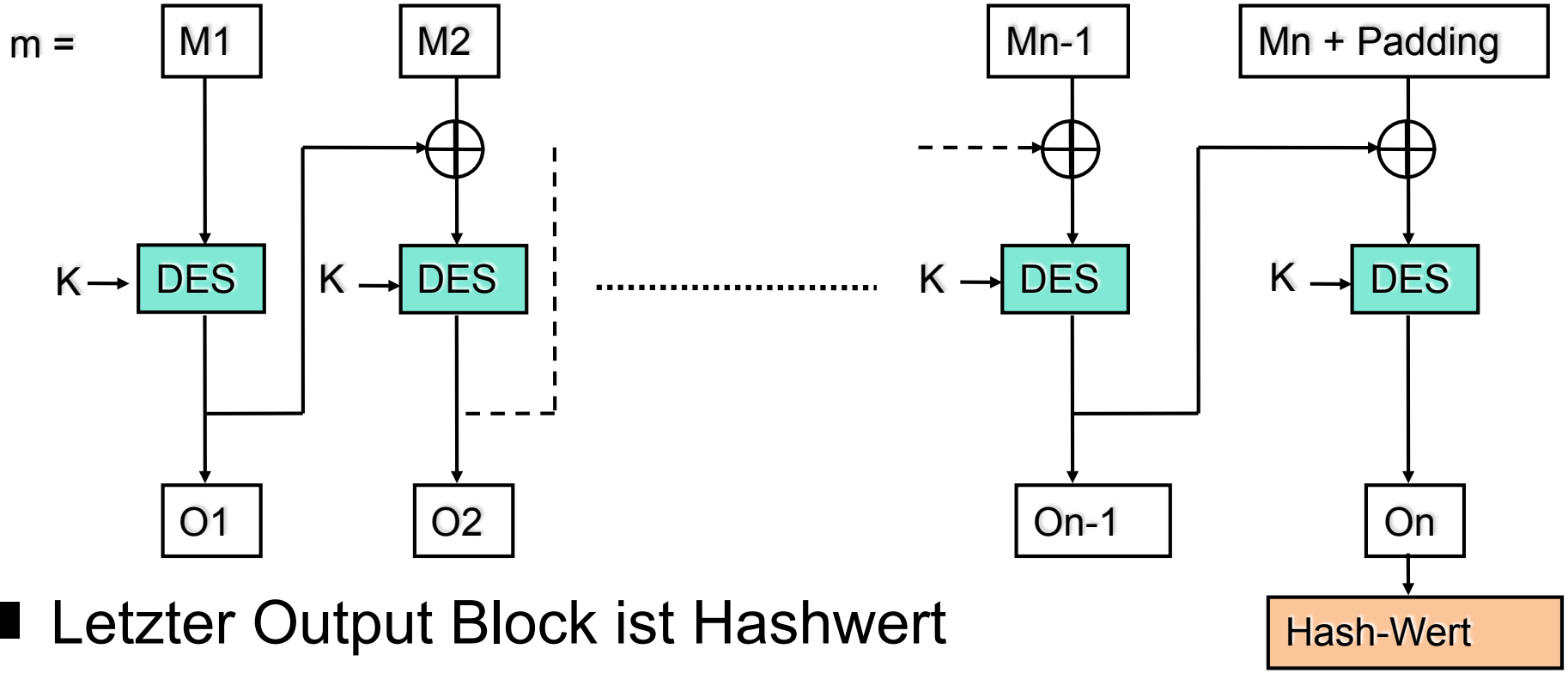
Konstruktion kryptographischer Hash-Funktionen

- Folge von Kompressionsfunktionen G
- Nachricht m wird in Blöcke Mi mit fester Länge y zerlegt
- Hash-Verfahren wird mit Initialisierungswert IV vorbelegt

- Letzter Block Mn muss ggf. auf vorgegebene Länge y „aufgefüllt“ werden (Padding)
- Als Kompressionsfunktion G können verwendet werden:
  - Hash-Funktionen auf der Basis symmetrischer Blockchiffren
  - Dedizierte Hash-Funktionen

DES als Kompressionsfunktion

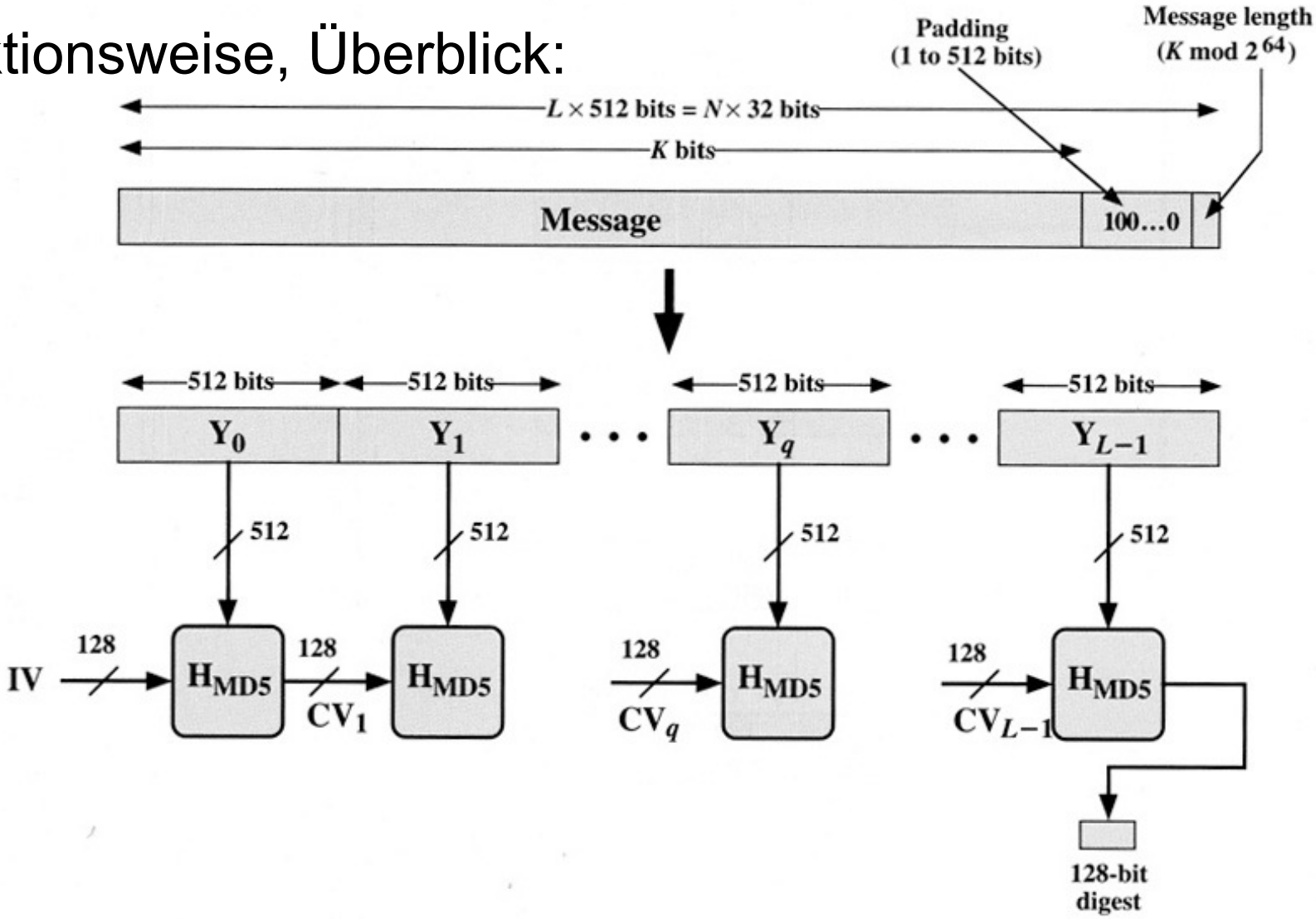
- DES im Cipher Block Chaining (CBC) Mode



- Letzter Output Block ist Hashwert
- Länge des Hashwerts? 64 Bits

Hash-Funktionen: MD5 (1991)

- Ausgabelänge 128 Bit, arbeitet auf 512-Bit-Eingabeblöcken
- Funktionsweise, Überblick:

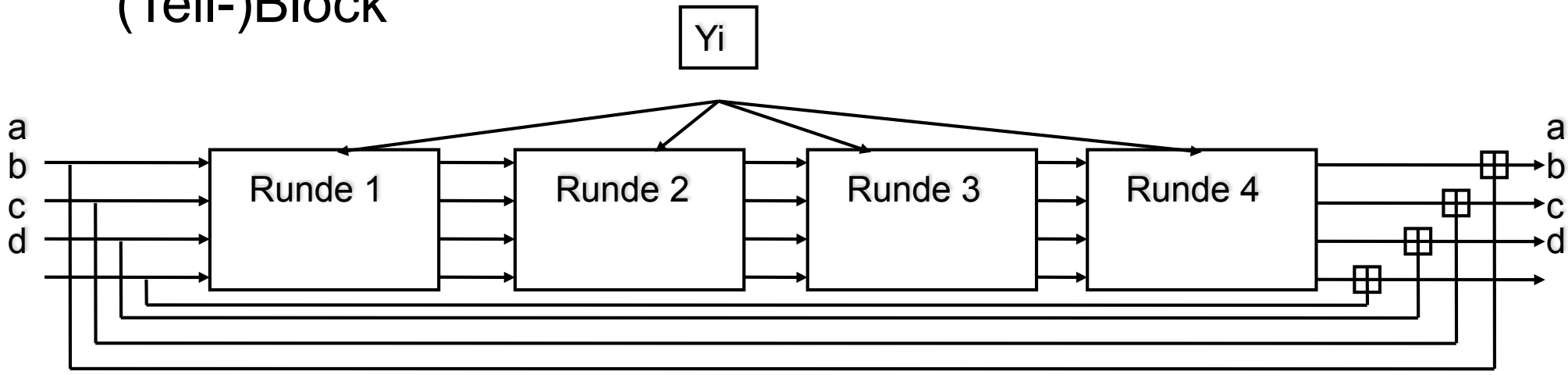


MD5: Ablauf

1. Padding Bits der Nachricht hinzufügen
2. Länge der Originalnachricht (mod 2^{64}) anfügen
3. Nachricht in 512-Bit-Blöcke aufteilen
4. Initialisierung von 32-Bit-Variablen:

   A = 0x01234567 C = 0xFEDCBA98

   B = 0x89ABCDEF D = 0x76543210
5. Zuweisung a=A, b=B, c=C, d=D
6. Kompressionsfunktion H_{MD5} angewendet auf jeden (Teil-)Block



Sicherheit von MD5

- Differentielle Kryptanalyse auf MD5 mit nur einer Runde [Bers 92]:
  - Für jede der 4 Runden einzeln möglich
  - Angriff auf alle 4 Runden konnte nicht gezeigt werden
- Pseudokollision [BoBo 93]:
  - Zwei verschiedene Variablenbelegungen von a,b,c,d führen für verschiedene Inputblöcke zum gleichen Outputblock
  - Damals schien eine Erweiterung des Ansatzes zu einem allgemeinen Angriff nicht möglich
- Erzeugung einer Kollision in der Kompressionsfunktion [Dobb 96]:
  - Zwei 512 Bit Blöcke produzieren den selben 128 Bit Output
  - Bis dahin gefährlichster bekannter Angriff

Sicherheit von MD5 (Forts.)

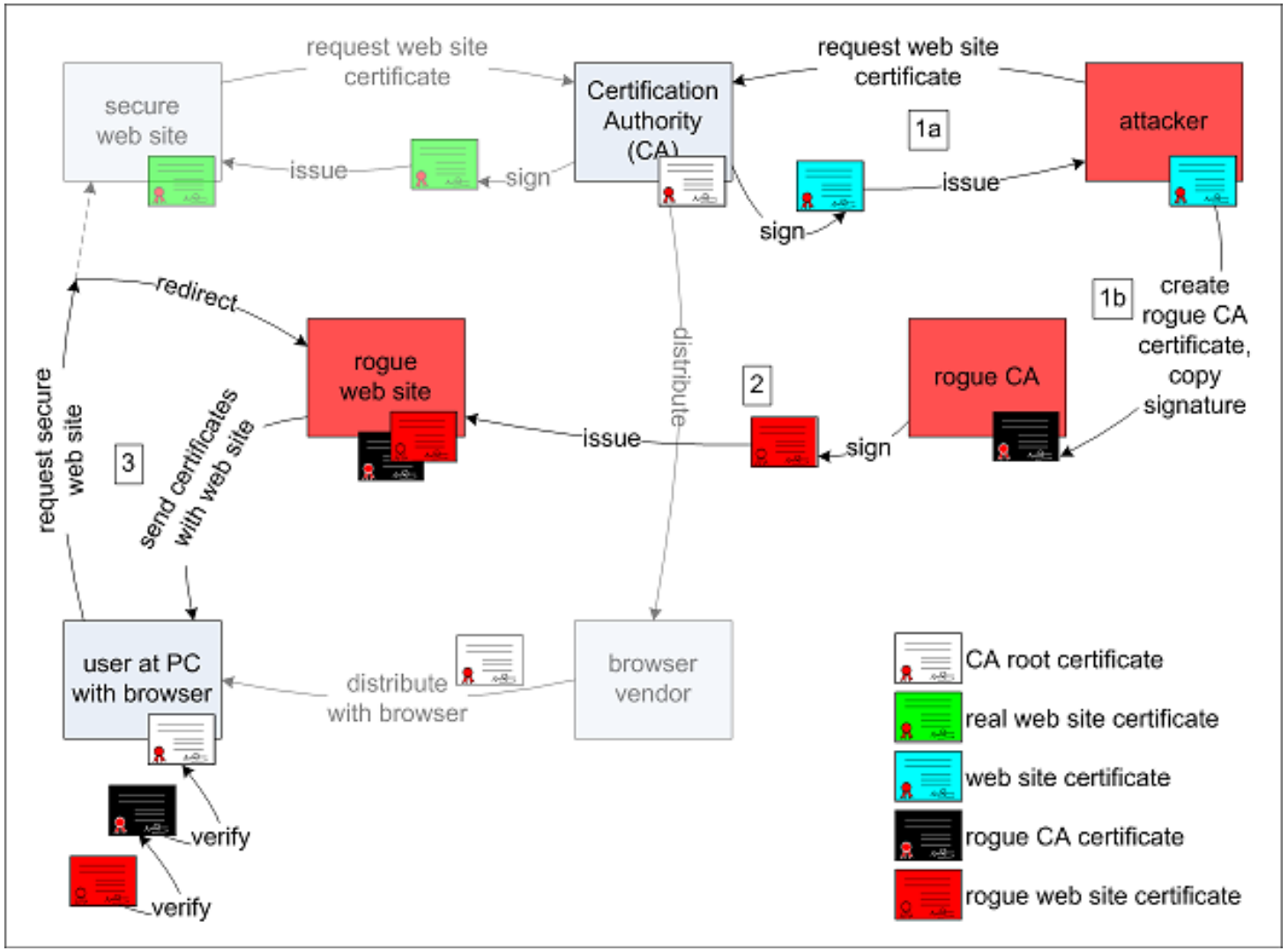
- Kollision gefunden [Wang,Feng,Lai,Yu 2004]:
  - MD5(M,Ni) = MD5(M',Ni')
  - M und M' zu finden dauert ca. eine Stunde (IBM P690 Cluster)
  - danach Ni und Ni' zu finden 15 Sek. bis 5 Minuten
  - funktioniert mit beliebigen Initialisierungsvektor IV
  - In der Arbeit werden auch Kollisionen für MD4, HAVAL-128 und RIPEMD-128 angegeben
  - Ende des MD5CRK-Projekts (distributed birthday attack)
- Kollision in X.509 Zertifikat gefunden (Kollision in den Schlüsseln) [de Weger 2005]
- Kollision in X.509 Zertifikat mit unterschiedlichen Identitäten
  [Stevens, Lenstra, de Weger 2006/2007]

➡ MD5 (und SHA-1) nicht mehr verwenden!

➡ Algorithmen mit längeren Hash-Werten verwenden:
z.B. SHA-256, Whirlpool, SHA-3, o.ä.

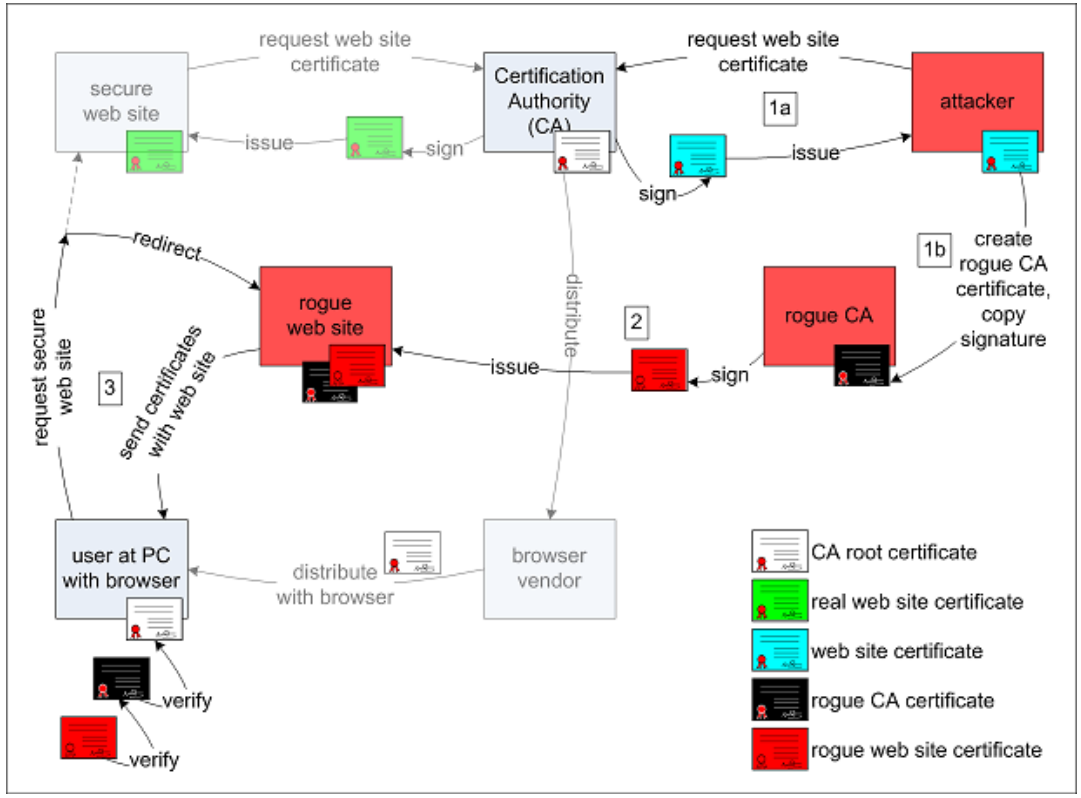
Sicherheit von MD5 (Forts.)

- Bislang umfangreichster, praktisch relevanter Angriff [SSALMOW08]: http://www.win.tue.nl/hashclash/rogue-ca/



Sicherheit von MD5 (Forts.)

- Bislang umfangreichster, praktisch relevanter Angriff [SSALMOW08]: http://www.win.tue.nl/hashclash/rogue-ca/



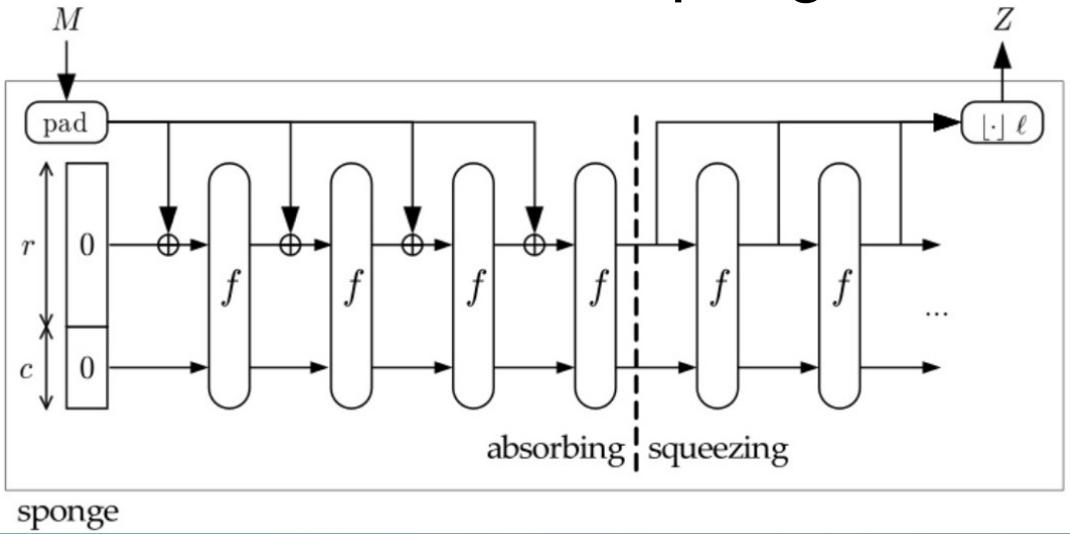
- Alle Browser, die RapidSSL-Zertifikaten vertrauten, vertrauten auch den mit dem „rogue CA certificate“ ausgestellten Zertifikaten
- Man-in-the-Middle Angriffe: Browser kann bei SSL-Zertifikaten, die MD5-Hashsummen verwenden, die Server-Authentizität nicht mehr zuverlässig prüfen

SHA-3

- 10/2012 vom NIST als Nachfolger von SHA-2 standardisiert

- 2007: Wettbewerb ähnlich zu AES-Standardisierung:
  - motiviert durch erfolgreiche Angriffe auf MD5 und SHA-1
  - 64 Einreichungen, 14 Algorithmen in engerer Auswahl, 5 Finalisten
  - Gewinner: Keccak von Bertoni, Daemen, Peeters und van Assche

- Innovativer Ansatz: Sponge-Funktion



sponge

Zwei Phasen: absorbing/squeezing

Variable Output-Länge

Bildquelle: http://sponge.noekeon.org

Keccak: Parametrisierung und Keccak-f

- Als SHA-3 standardisierte Varianten umfassen u.a.
  - SHA3-256: r=1152, c=448, Ausgabe abgeschnitten nach 256 Bits
  - SHA3-512: r=576, c=1024, Ausgabe abgeschnitten nach 512 Bits
- f[b] Keccak Permutationsfunktion; Breite der Perumutation $b = c + r = 25*2^l$
- Funktion f betrachtet State als dreidimensionales Array von GF[2]

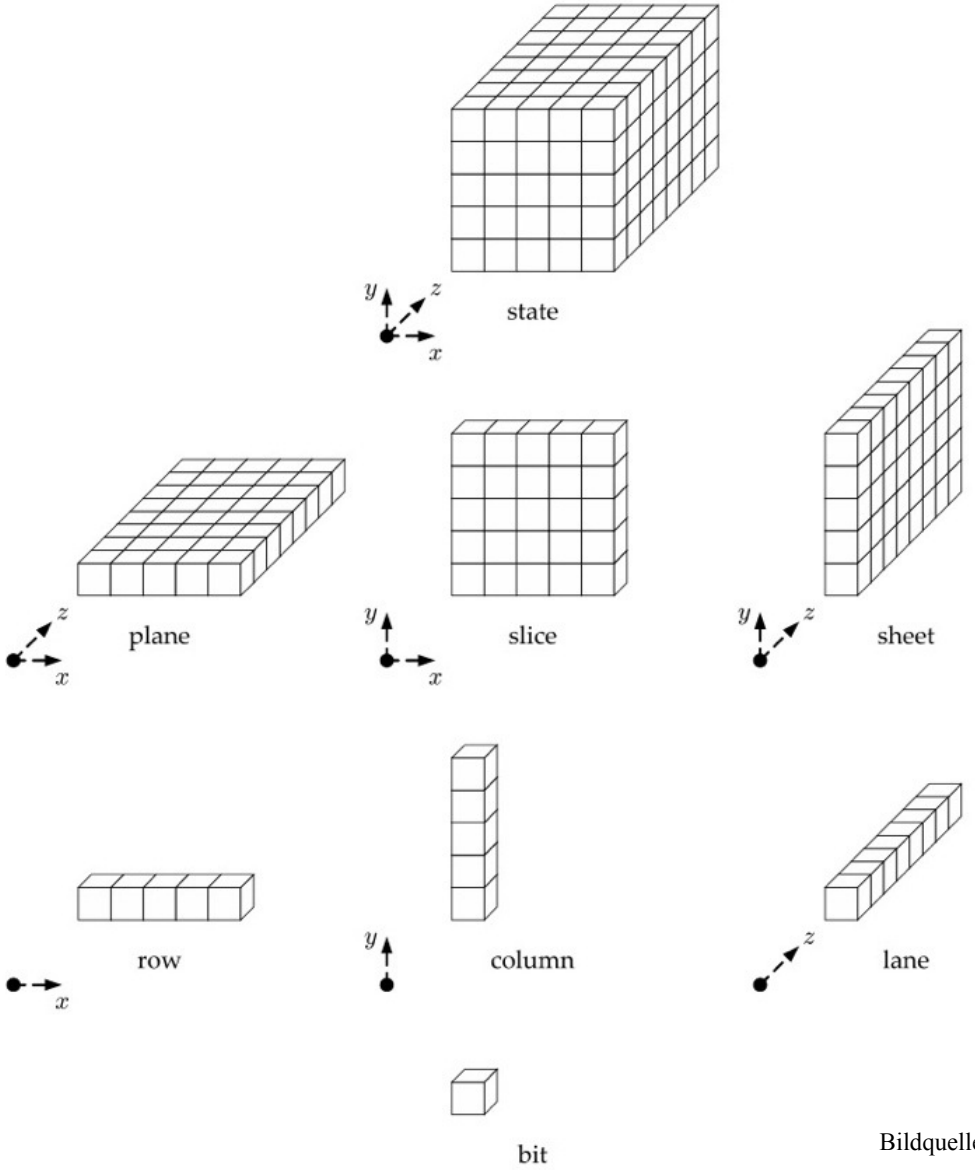
  $a[5][5][w]$ mit $w = 2^l$, $b = c + r = 25*2^l$

  Beispiel SHA3-256: b = 1152 + 448 = 1600,

  d.h. l = 6, w = 64

- Jede Anwendung von f besteht aus nr Runden:
  nr = 12 + 2*l, d.h. für SHA3-256: nr = 24

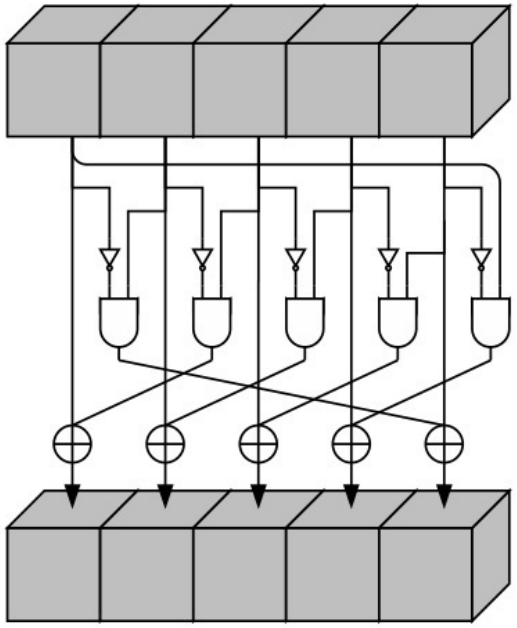
Keccak: Terminologie zum State



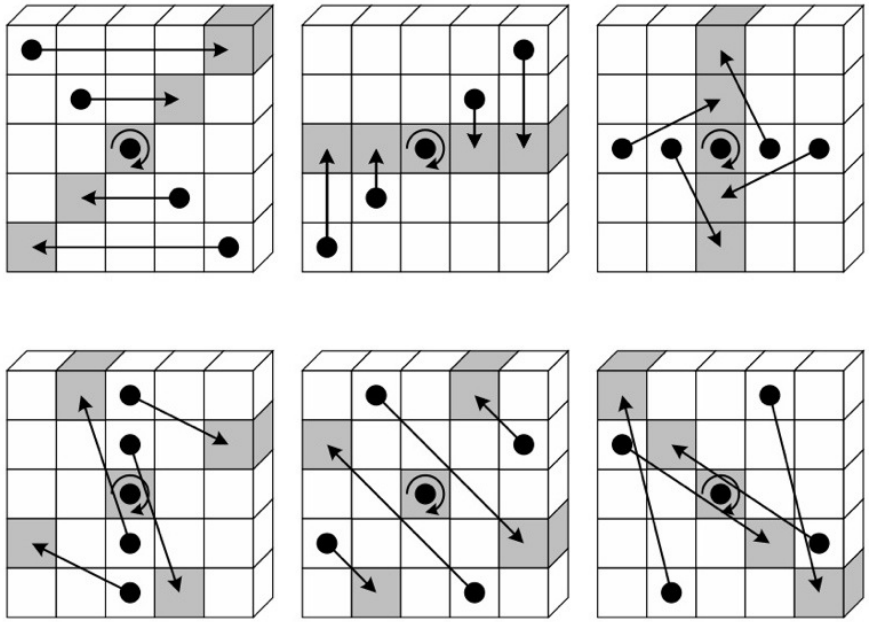
Bildquelle: http://keccak.noekeon.org/Keccak-reference-3.0.pdf

Keccak-f: Runden

- Jede Runde besteht aus fünf Schritten:
  - $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$
    - Addition von Rundenkonstanten
    - Nichtlinearität
    - Erhöhung der Diffusion in allen drei Dimensionen



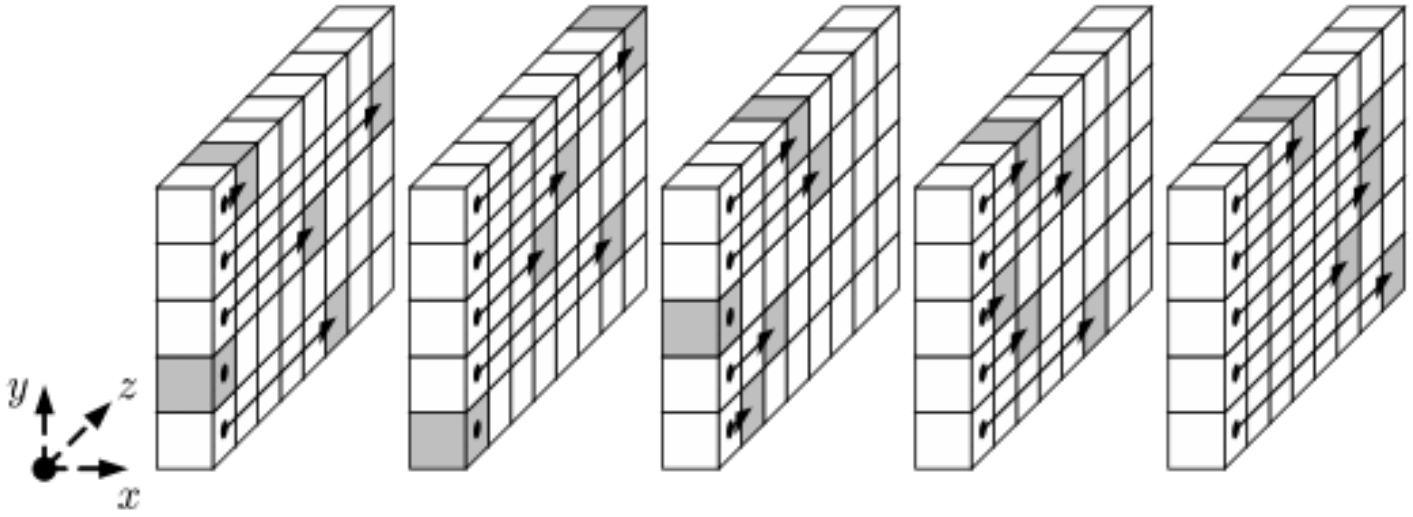
χ



π

Keccak-f: Runden

- Jede Runde besteht aus fünf Schritten:
  - $\mathrm{R} = \iota \circ \chi \circ \pi \circ \rho \circ \theta$
    - Addition von Rundenkonstanten
    - Nichtlinearität
    - Erhöhung der Diffusion in allen drei Dimensionen



ρ

Keccak: Bewertung

- Innovativer Ansatz:
  - Vermeidet Probleme klassischer Merkle-Damgard-Konstrukte wie MD5;
  - ist entsprechend aber noch weniger von Kryptanalytikern untersucht.
  - Komplementär zu SHA-2 verwendbar.
- Variable Output-Länge
  - ermöglicht flexible Anpassung an jeweiligen Bedarf
  - Gute Eignung als PRNG für Stream Ciphers
- Effiziente Implementierung in Hard- und Software möglich
- Konservative Sicherheitsreserve durch große Rundenzahl