The most efficient SHA-1 attacks

Seminar Selected Topics in (Mobile) Security

Johannes Gilger

LuFG IT-Sicherheit

Aachen, March 18, 2010

Übersicht

Aufbau

1. Was ist eine Hashfunktion?
2. Wie funktionieren SHA-0 und SHA-1?
3. Was ist eine Kollision? Wie kann man Kollisionen nutzen?
4. Wie sind Kollisionsattacken auf SHA-0/SHA-1 aufgebaut?
5. Zeitlicher Verlauf bisheriger Ergebnisse

Definition Hashfunktion

Eine Funktion $h : D \longrightarrow B$, so dass $|D| > |B|$. Feste Ausgabelänge.

Sicherheitsanforderungen:

- *Preimage resistance:* Für $y = h(x)$ bekannt, x unbekannt soll es praktisch unmöglich ein beliebiges x' zu finden so dass $h(x') = y$. Komplexität $O(2^n)$.
- *Second preimage resistance:* Für x_1 und $h(x_1)$ gegeben soll es schwer einen Wert $x_2 \neq x_1$ zu finden, so dass $h(x_1) = h(x_2)$. Komplexität $O(2^n)$.
- *Collision resistance:* Es soll schwer sein Werte $x_1 \neq x_2$ zu finden, so dass $h(x_1) = h(x_2)$. Komplexität $O(2^{\frac{n}{2}})$ (Birthday Paradox).

SHA: Secure Hash Algorithm

- Entwickelt vom NIST
- SHA-0, SHA-1 und SHA-2 Familie
- SHA-3 ab 2012
- Eingabe: Bis zu 2^{64} Bits
- Ausgabe: 160 Bit (SHA-0 und SHA-1)

Aufbau des SHA

1. Padding / Aufteilung
2. Expansion
3. Rundenfunktion / Kompression

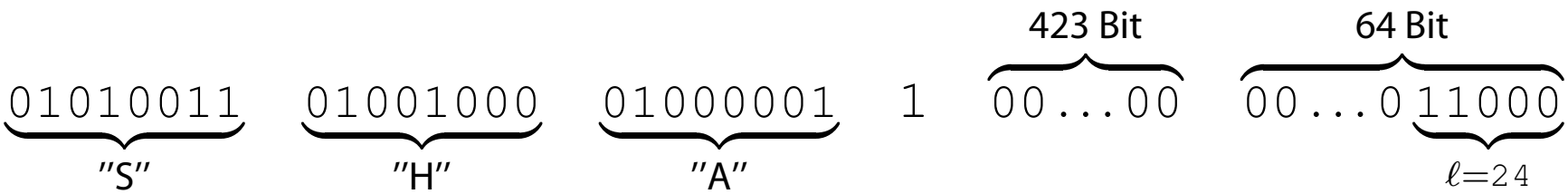
Notationen und Funktionen

- Bitweise Operatoren: $\wedge$, $\vee$, $\oplus$ und $\neg$.
- Addition modulo 2^{32}: $x + y = (x + y) \bmod 2^{32}$
- Zyklische Linksrotation um n Stellen: $ROTL^n(x)$
- Konkatenation: $\|$
- Wort: 32 Bit
- Bits: $0, \ldots, 31$

Padding

- Padding auf Länge $x \cdot 512$ Bit
- Einteilung in 512-Bit-Blöcke

Padding - Beispiel



Nachrichtenexpansion

Expansion von 512 Bit (Block) auf 2560 Bit (bzw. $80 \cdot 32$ Bit):

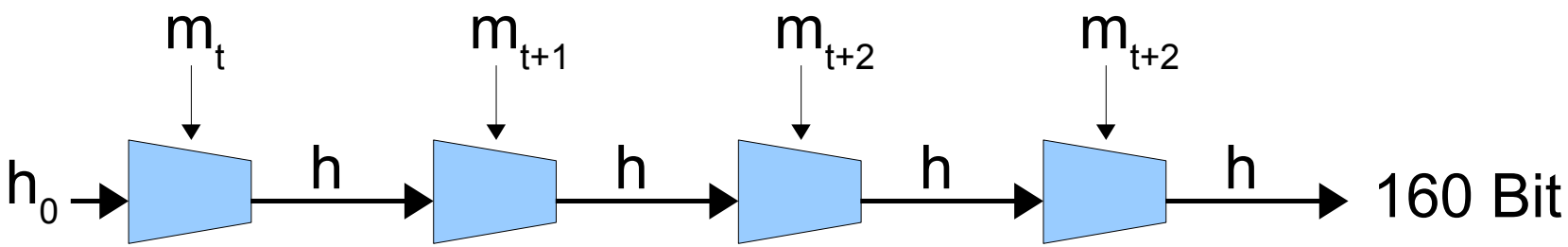
$$W_t = \begin{cases} M_t & 0 \leq t \leq 15 \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & 16 \leq t \leq 79 \end{cases}$$

Unterschied zwischen SHA-0 und SHA-1:

$$W_t = \begin{cases} M_t & 0 \leq t \leq 15 \\ W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16} & 16 \leq t \leq 79 \end{cases}$$

$\Rightarrow$ Die ersten 16 Wörter sind direkt editierbar

Blockweise Bearbeitung (Merkle-Damgård Konstruktion)

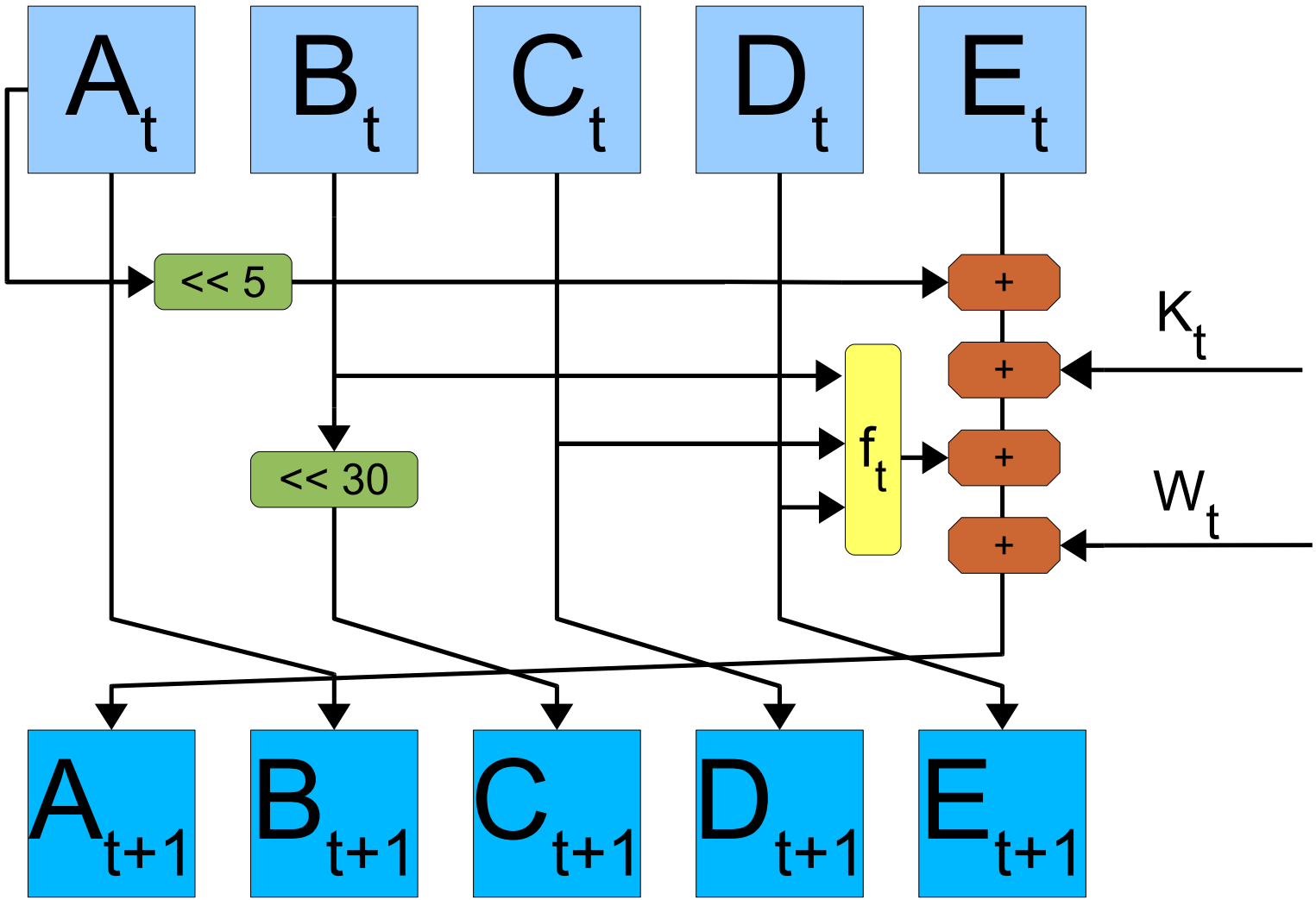


Schrittfunktion

Ein Schritt pro Wort (32 Bit). 80 Schritte, aufgeteilt in 4 Runden:

$$A_{t+1} = ROTL^5(A_t) + f_t(B_t, C_t, D_t) + E_t + K_t + W_t$$
$$B_{t+1} = A_t$$
$$C_{t+1} = ROTL^{30}(B_t)$$
$$D_{t+1} = C_t$$
$$E_{t+1} = D_t$$

SHA-0 / SHA-1 erklärt



Rundenfunktionen

Die Funktion $f_t(x, y, z)$ ist von der Runde t abhängig:

$$
\begin{array}{rll}
0 \leq t \leq 19 & IF(x,y,z) & = (x \wedge y) \vee (\neg x \wedge z) \\
20 \leq t \leq 30 & XOR(x,y,z) & = x \oplus y \oplus z \\
40 \leq t \leq 59 & MAJ(x,y,z) & = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z) \\
60 \leq t \leq 79 & XOR(x,y,z) & = x \oplus y \oplus z
\end{array}
$$

IF und MAJ sind die nicht-linearen Funktionen.
Erinnerung: Alle Funktion operieren bitweise!

Rundenkonstanten

Berechnung mittels $2^{30} \cdot \sqrt{x}$ für $x \in \{2, 3, 5, 10\}$.

$$
\begin{aligned}
0 \leq t \leq 19 \quad & K_t = \texttt{0x5a827999} \\
20 \leq t \leq 30 \quad & K_t = \texttt{0x6ed9eba1} \\
40 \leq t \leq 59 \quad & K_t = \texttt{0x8f1bbcdc} \\
60 \leq t \leq 79 \quad & K_t = \texttt{0xca62c1d6}
\end{aligned}
$$

”Nothing-up-my-sleeve number”

Rundenvariablen

$$
\begin{aligned}
H_0^{(i)} &= A + H_0^{(i-1)} \\
H_1^{(i)} &= B + H_1^{(i-1)} \\
H_2^{(i)} &= C + H_2^{(i-1)} \\
H_3^{(i)} &= D + H_3^{(i-1)} \\
H_4^{(i)} &= E + H_4^{(i-1)}
\end{aligned}
$$

160 Bit Hash besteht aus der Ausgabe des letzten Blocks:

$$H_0^{(n)} \| H_1^{(n)} \| H_2^{(n)} \| H_3^{(n)} \| H_4^{(n)}$$

IV - Initiale Vektoren

$$H_0^{(0)} = \texttt{0x67452301}$$
$$H_1^{(0)} = \texttt{0xefcdab89}$$
$$H_2^{(0)} = \texttt{0x98badcfe}$$
$$H_3^{(0)} = \texttt{0x10325476}$$
$$H_4^{(0)} = \texttt{0xc3d2e1f0}$$

Hash-Kollisionen

Nachricht:

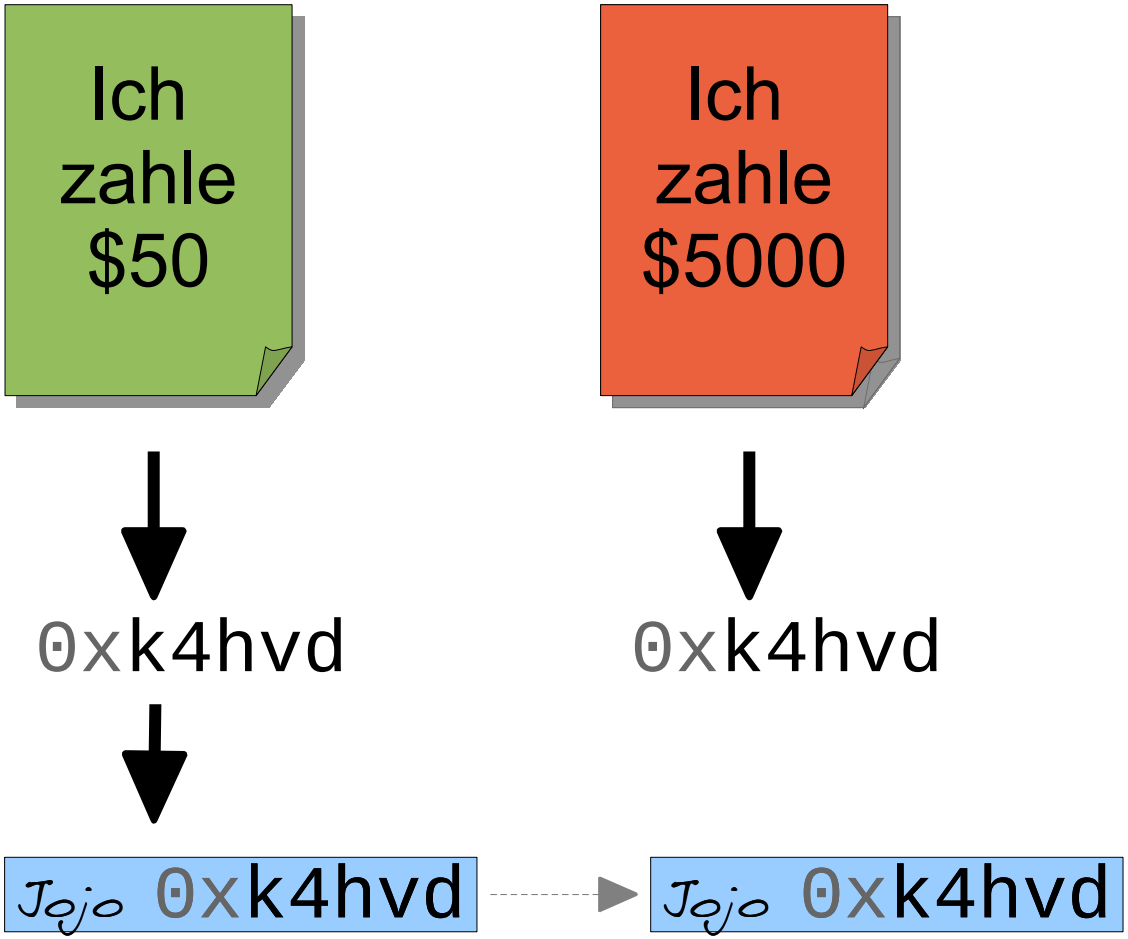
```
a766a602 b65cffe7 73bcf258 26b322b3 d01b1a97 2684ef53 3e3b4b7f 53fe3762
24c08e47 e959b2bc 3b519880 b9286568 247d110f 70f5c5e2 b4590ca3 f55f52fe
effd4c8f e68de835 329e603c c51e7f02 545410d1 671d108d f5a4000d cf20a439
4949d72c d14fbb03 45cf3a29 5dcda89f 998f8755 2c9a58b1 bdc38483 5e477185
f96e68be bb0025d2 d2b69edf 21724198 f688b41d eb9b4913 fbe696b5 457ab399
21e1d759 1f89de84 57e8613c 6c9e3b24 2879d4d8 783b2d9c a9935ea5 26a729c0
6edfc501 37e69330 be976012 cc5dfe1c 14c4c68b d1db3ecb 24438a59 a09b5db4
35563e0d 8bdf572f 77b53065 cef31f32 dc9dbaa0 4146261e 9994bd5c d0758e3d
```

Modifizierte Nachricht:

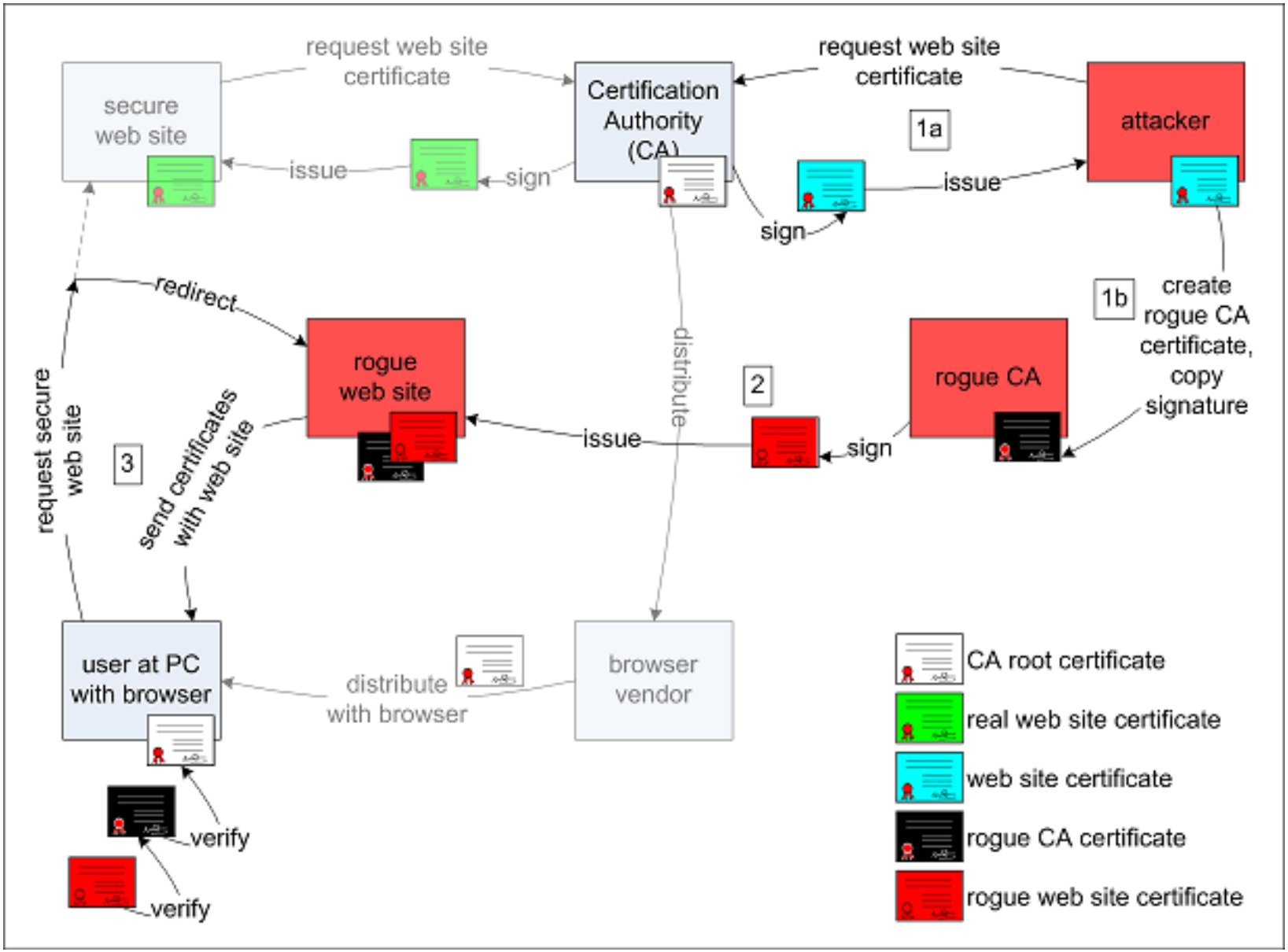
```
a766a602 b65cffe7 73bcf258 26b322b1 d01b1ad7 2684ef51 be3b4b7f d3fe3762
a4c08e45 e959b2fc 3b519880 39286528 a47d110d 70f5c5e0 34590ce3 755f52fc
6ffd4c8d 668de875 329e603e 451e7f02 d45410d1 e71d108d f5a4000d cf20a439
4949d72c d14fbb01 45cf3a69 5dcda89d 198f8755 ac9a58b1 3dc38481 5e4771c5
796e68fe bb0025d0 52b69edd a17241d8 7688b41f 6b9b4911 7be696f5 c57ab399
a1e1d719 9f89de86 57e8613c ec9e3b26 a879d498 783b2d9e 29935ea7 a6a72980
6edfc503 37e69330 3e976010 4c5dfe5c 14c4c689 51db3ecb a4438a59 209b5db4
35563e0d 8bdf572f 77b53065 cef31f30 dc9dbae0 4146261c 1994bd5c 50758e3d
```

Hashwert (SHA-0): `c9f16077 7d4086fe 8095fba5 8b7e20c2 28a4006b`

Einfaches Szenario

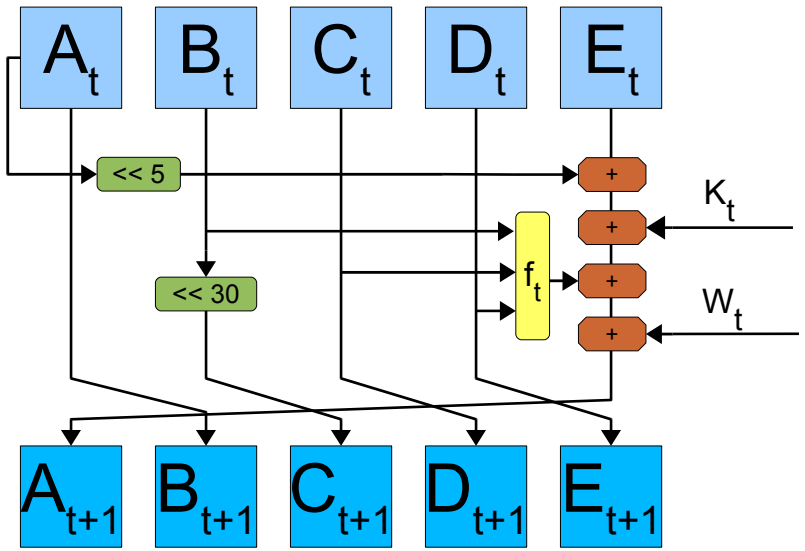


MD5 Rogue CA - 25C3

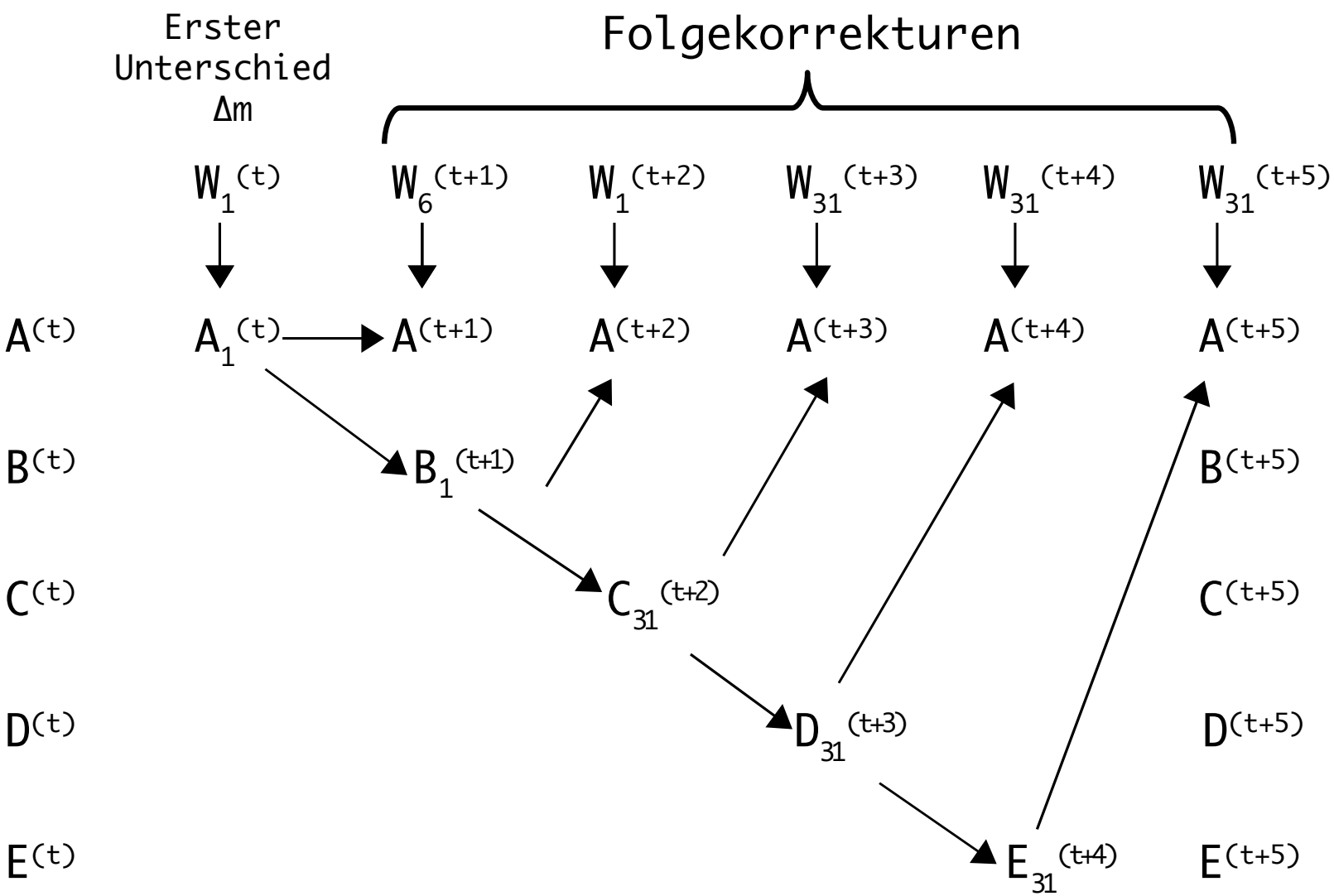


SHA - Rundenfunktion Erinnerung

$$A_{t+1} = ROTL^5(A_t) + f_t(B_t, C_t, D_t) + E_t + K_t + W_t$$
$$B_{t+1} = A_t$$
$$C_{t+1} = ROTL^{30}(B_t)$$
$$D_{t+1} = C_t$$
$$E_{t+1} = D_t$$



Die 6-Schritt Kollision (Chabaud / Joux)



Die 6-Schritt Kollision (Chabaud / Joux)

Allgemeines Vorgehen

1. Konstruktion einer linearen Approximation von SHA (z.B. XOR statt IF/MAJ und Addition)
2. Finden eines Kollisionspfads für lineare Approximation
3. Wiedereinsetzen der nicht-linearen Funktion und Untersuchung der Wahrscheinlichkeit für Verhalten wie lineare Variante

Beobachtung

"Einzelne" 6-Schritt Kollisionen nicht möglich

Ein Störvektor für SHA-0

```
00000  00100010000000101111 01100011100000010100
       01000100100100111011 00110000111110000000
```

Wahrscheinlichkeit: 2^{-69} bzw. 2^{-61}

Anforderungen an den Störvektor

1. $x_i = 0 : i \in \{75, \ldots, 79\}$. Kollision bis zum letzten Schritt fertig haben.
2. $x_i = 0 : i \in \{-5, \ldots, -1\}$. Teilkollisionen in den ersten Schritten vermeiden.
3. Keine benachbarten 1-Bits in den ersten 17 Schritten (wegen IF-Funktion).

$\Rightarrow$ Suchraum von 2^{16} durch Expansion. Gefunden: 3 Vektoren

Verhalten von IF bei Änderung von B

B	C	D	IF
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1

B′	C	D	IF
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

IF verhält sich in der Hälfte der Fälle ”richtig” (wie die lineare Approximation) wenn B negiert wird

Zusätzlich Wahrscheinlichkeit dass kein Übertrag auftritt
$\Rightarrow$ Gesamtwahrscheinlichkeit für IF bei Änderung von B: $\frac{1}{4}$

Wahrscheinlichkeiten für lineares Verhalten

	Wahrscheinlichkeit	
Unterschied	*IF*	*MAJ*
$b'^{(t)} \oplus b^{(t)} = 2^1$	$1/4$	$1/2$
$c'^{(t)} \oplus c^{(t)} = 2^{31}$ oder $d'^{(t)} \oplus d^{(t)} = 2^{31}$	$1/2$	$1/2$
$c'^{(t)} \oplus c^{(t)} = 2^{31}$ und $d'^{(t)} \oplus d^{(t)} = 2^{31}$	0	1

Zusätzlich: Wahrscheinlichkeit dass Addition in Schritt $i + 1$ und $i + 2$ nicht zum Übertrag führt

Gesamtwahrscheinlichkeit

Perturbation in round i	$f^{(i+2)}$	case	$f^{(i+3)}$	case	$f^{(i+4)}$	case	overall probability	probability logarithm
2	IF	2	IF	3	IF	3	1/32	5
6	IF	2	IF	3	IF	3	1/32	5
14	IF	2	IF	3	IF	3	1/32	4 + 1
16	IF	2	IF	3	XOR	–	1/16	4
17	IF	2	XOR	–	XOR	–	1/8	3
18, 19, 21								
22, 26, 27	XOR	–	XOR	–	XOR	–	1/4	2
28, 35								
37	XOR	–	MAJ	3	MAJ	3	1/16	4
41	MAJ	2	MAJ	3	MAJ	3	1/16	4
45	MAJ	2	MAJ	3	MAJ	3	1/16	4
48	MAJ	2	MAJ	3	MAJ	3	1/16	4
51	MAJ	2	MAJ	3	MAJ	3	1/16	4
54	MAJ	2	MAJ	3	MAJ	4	1/8	3
55	MAJ	2	MAJ	4	MAJ	4	1/4	2
56	MAJ	2	MAJ	4	XOR	–	1/4	2
58, 59, 62								
63, 68, 69	XOR	–	XOR	–	XOR	–	1/4	2
70, 71, 72								

```
00000   00100010000000101111 01100011100000010100
        01000100100100111011 00110000111110000000
```

Beinahe-Kollisionen und Multiblock-Kollisionen

Beinahekollision

- Bedeutet Kollision in großer Anzahl von Bits, z.B. 142 von 160 Bit
- Leichter zu berechnen, weniger Anforderungen an Störvektor

Multiblock-Kollisionen

- Zwei Blöcke mit Hashwerten H'_1 und H'_2 die sich von H_1, H_2 so unterscheiden dass $H'_1 + H'_2 = H_1 + H_2$
- Suche nach zweitem Block für hat gleiche Komplexität wie für den ersten, bedeutet also ''nur'' Faktor 2

$\Rightarrow$ Gute Alternative zu den Ein-Block-Kollisionen

Hash-Kollisionen

Nachricht:

```
a766a602 b65cffe7 73bcf258 26b322b3 d01b1a97 2684ef53 3e3b4b7f 53fe3762
24c08e47 e959b2bc 3b519880 b9286568 247d110f 70f5c5e2 b4590ca3 f55f52fe
effd4c8f e68de835 329e603c c51e7f02 545410d1 671d108d f5a4000d cf20a439
4949d72c d14fbb03 45cf3a29 5dcda89f 998f8755 2c9a58b1 bdc38483 5e477185
f96e68be bb0025d2 d2b69edf 21724198 f688b41d eb9b4913 fbe696b5 457ab399
21e1d759 1f89de84 57e8613c 6c9e3b24 2879d4d8 783b2d9c a9935ea5 26a729c0
6edfc501 37e69330 be976012 cc5dfe1c 14c4c68b d1db3ecb 24438a59 a09b5db4
35563e0d 8bdf572f 77b53065 cef31f32 dc9dbaa0 4146261e 9994bd5c d0758e3d
```

Modifizierte Nachricht:

```
a766a602 b65cffe7 73bcf258 26b322b1 d01b1ad7 2684ef51 be3b4b7f d3fe3762
a4c08e45 e959b2fc 3b519880 39286528 a47d110d 70f5c5e0 34590ce3 755f52fc
6ffd4c8d 668de875 329e603e 451e7f02 d45410d1 e71d108d f5a4000d cf20a439
4949d72c d14fbb01 45cf3a69 5dcda89d 198f8755 ac9a58b1 3dc38481 5e4771c5
796e68fe bb0025d0 52b69edd a17241d8 7688b41f 6b9b4911 7be696f5 c57ab399
a1e1d719 9f89de86 57e8613c ec9e3b26 a879d498 783b2d9e 29935ea7 a6a72980
6edfc503 37e69330 3e976010 4c5dfe5c 14c4c689 51db3ecb a4438a59 209b5db4
35563e0d 8bdf572f 77b53065 cef31f30 dc9dbae0 4146261c 1994bd5c 50758e3d
```

Hashwert (SHA-0): `c9f16077 7d4086fe 8095fba5 8b7e20c2 28a4006b`

SHA-0 Angriffe: Allgemein

Fazit

- Begriffe:
  - **Diff. Pfad** ist anderer Weg durch die Hashfunktion mit gleichem Ziel
  - **Störvektor** wird benutzt um differentiellen Pfad zu konstruieren
- Hamminggewicht des Störvektors wichtig
- Beinahe-Kollisionen zusammen mit Multi-Block-Kollisionen sind einfacher

Probleme / Unterschiede

Problem: Die Linksrotation während der Expansion

- Suchraum für Störvektoren nun 2^{512} (16 32Bit-Wörter) statt 2^{16}.
- Betrachten von Bits ist nicht mehr einzeln möglich.
- Überträge können überall auftreten.

Aber: 6-Schritt-Kollision funktioniert nach wie vor.

SHA-1 Angriff: Methode nach Wang (2005)

Ideen

Ziel: Geringes Hamming-Gewicht im Störvektor in Runden 2-4

- Suche nur in 2 Bits und 16 konsekutiven Schritten (Expansion)
  - $\Rightarrow$ Suchraum nur $2^{32} \cdot 64 = 2^{38}$
- Störvektoren für 1-Block-Kollisionen haben zu großes Hamminggewicht
  - $\Rightarrow$ Benutzung von Beinahe-Kollisionen
  - $\Rightarrow$ Kollisionen müssen nicht bis Schritt 80 fertig sein
  - $\Rightarrow$ Suche nach zweitem Beinahe-Block genauso schwer, Faktor 2
- Nachrichtenmodifikation für Runde 1
  - $\Rightarrow$ Wahrscheinlichkeit für richtiges Verhalten von IF ist 1
  - Erweiterte Nachrichtenmodifikation: Berechnung ab Runde 22
- Frühes Stoppen beim Implementieren

Wahrscheinlichkeit von 2^{-69} Kollisionen zu finden.

Zeitleiste der Angriffe - SHA-0

SHA-0

- *1997*: Wang: 6-Schritt Kollision (nur Chinesisch)
- *1998*: Chabaud/Joux: 6-Schritt Kollision, Komplexität 2^{61}
- *2004*: Biham/Chen: Beinahe-Kollision auf 142 von 160 Bit. 2^{40}
- *2005*: Biham/Chen: Kollisionen mit 2^{51}
- *2005*: Wang: Kollisionen mit 2^{39}
- *2006*: Naito: Kollisionen mit 2^{36}
- *2008*: Manuel/Peyrin: Kollisionen mit 2^{33}

SHA-0 Kollisionen in einer Stunde möglich!

Zeitleiste der Angriffe - SHA-1

SHA-1

- *2005*: Wang: Kollisionen mit 2^{69}
- *2006*: Wang: Kollisionen mit 2^{63} (kein Paper bisher)
- *2009*: McDonald/Hawkes/Pieprzyk: Kollisionen mit 2^{52} (zurückgezogen)

SHA-1-Kollisionen noch nicht praktikabel (auch mit 2^{52} noch nicht)

Literatur

Chabaud und Joux

Differential collisions in SHA-0

Lecture Notes in Computer Science, 56--71, 1998.

Biham und Chen

Near-collisions of SHA-0

Lecture Notes in Computer Science, 290--305, 2004.

Biham, Chen, Joux, Carribault, Lemuet und Jalby

Collisions of SHA-0 und Reduced SHA-1

Advances in Cryptology--EUROCRYPT, 3494, 36--57, 2005.

Wang, Yin und Yu

Efficient collision search attacks on SHA-0

Lecture Notes in Computer Science, 3621, 1--16, 2005.

Finding collisions in the full SHA-1

Lecture Notes in Computer Science, 3621, 17--36, 2005.

Vielen Dank für's Zuhören.

Fragen?