

Practical Free-Start Collision Attacks on full SHA-1

Pierre Karpman

Inria and École polytechnique, France
Nanyang Technological University, Singapore

Joint work with Thomas Peyrin and Marc Stevens

Séminaire CCA, Paris
2016-04-01

Title deconstruction

Practical

We can compute it

Free-Start

Not unlike a false start

Collision

As in $f(x) = f(x')$

Attacks

We're the baddies

on full

The real thing this time!

SHA-1

Not a fish



Introduction

SHA-1 in brief

History of SHA-1 attacks

Our attack

Implementation

Results

Introduction

SHA-1 in brief

History of SHA-1 attacks

Our attack

Implementation

Results

Hash functions

Hash function

A (binary) hash function is a mapping $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^n$

- ▶ Many uses in **crypto**: hash n' sign, MAC constructions...
- ▶ It is a **keyless** primitive
- ▶ Sooo, what's a **good hash function**?

Three security notions (informal)

First preimage resistance

Given t , find m such that $\mathcal{H}(m) = t$

Best generic attack is in $\mathcal{O}(2^n)$

Second preimage resistance

Given m , find $m' \neq m$ such that $\mathcal{H}(m) = \mathcal{H}(m')$

Best generic attack is in $\mathcal{O}(2^n)$

Collision resistance

Find $m, m' \neq m$ such that $\mathcal{H}(m) = \mathcal{H}(m')$

Best generic attack is in $\mathcal{O}(2^{\frac{n}{2}})$

Merkle-Damgård construction

A domain of $\{0,1\}^*$ is annoying, so...

1 Start from a **compression function** $f: \{0,1\}^n \times \{0,1\}^b \rightarrow \{0,1\}^n$

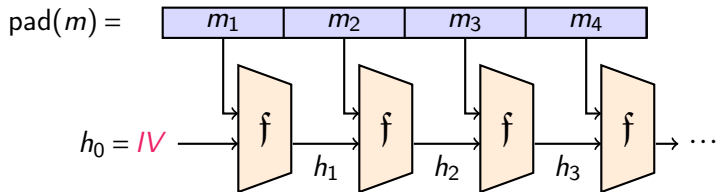
2 Use a **domain extender** \approx

$$\mathcal{H}(m_1 || m_2 || \dots || m_\ell) = f(f(\dots f(IV, m_1) \dots), m_\ell)$$

3 **Reduce the security** of \mathcal{H} to the one of f

- ▶ $A(\mathcal{H}) \Rightarrow A(f)$
- ▶ $\neg A(f) \Rightarrow \neg A(\mathcal{H})$
- ▶ $(A(f) \Rightarrow ???)$
 - ▶ Invalidates the security reduction, tho

MD in a picture



Additional security notions for MD

Semi-free-start collisions

The attacker may choose IV , but it must be the same for m and m'

Free-start preimages & collisions

No restrictions on IV whatsoever

Free-start preimages & collisions (variant)

Attack f instead of \mathcal{H}

What did we do?

- ▶ First try: collisions on 76/80 steps of the compression function of SHA-1 (95% of SHA-1)
- ▶ And it's practical
- ▶ Cost $\approx 2^{50.3}$ SHA-1, one inexpensive GPU is enough for fast results

- ▶ Second try: collisions on the full compression function of SHA-1 (100% of SHA-1)
- ▶ Still practical
- ▶ Cost $\approx 2^{57.5}$ SHA-1, 64 GPUs for a result in less than two weeks
- ▶ ?Not “the same attack as 1) with more computation power”

The collision on 80 steps

Message 1																				
IV_1	50	6b	01	78	ff	6d	18	90 20	22	91	fd	3a	de	38	71	b2	c6	65	ea	
M_1	9d	44	38	28 a5	ea	3d	f0 86	ea	a0	fa 77	83	a7	36							
	33	24	48	4d af	70	2a	aa a3	da	b6	79 d8	a6	9e	2d							
	54	38	20	ed a7	ff	fb	52 d3	ff	49	3f c3	ff	55	1e							
	fb	ff	d9	7f 55	fe	ee	f2 08	5a	f3	12 08	86	88	a9							
$\text{Compr}(IV_1, M_1)$	f0	20	48	6f	07	1b	f1	10	53	54	7a	86	f4	a7	15	3b	3c	95	0f	4b
Message 2																				
IV_2	50	6b	01	78	ff	6d	18	91 a0	22	91	fd	3a	de	38	71	b2	c6	65	ea	
M_2	3f	44	38	38 81	ea	3d	ec a0	ea	a0	ee 51	83	a7	2c							
	33	24	48	5d ab	70	2a	b6 6f	da	b6	6d d4	a6	9e	2f							
	94	38	20	fd 13	ff	fb	4e ef	ff	49	3b 7f	ff	55	04							
	db	ff	d9	6f 71	fe	ee	ee e4	5a	f3	06 04	86	88	ab							
$\text{Compr}(IV_2, M_2)$	f0	20	48	6f	07	1b	f1	10	53	54	7a	86	f4	a7	15	3b	3c	95	0f	4b

Introduction

SHA-1 in brief

History of SHA-1 attacks

Our attack

Implementation

Results

The SHA-1 hash function

- ▶ Designed by the NSA in 1995 as a quick fix to SHA-0
- ▶ Part of the MD4 family
- ▶ Hash size is 160 bits \Rightarrow collision security should be 80 bits
- ▶ Message blocks are 512-bit long
- ▶ Compression function in MD mode

SHA-1 compression function

Block cipher in Davies-Meyer mode

Block cipher: 5-branch ARX Feistel

$$A_{i+1} = A_i^{\circ 5} + \phi_{i \div 20}(A_{i-1}, A_{i-2}^{\circ 2}, A_{i-3}^{\circ 2}) + A_{i-4}^{\circ 2} + W_i + K_{i \div 20}$$

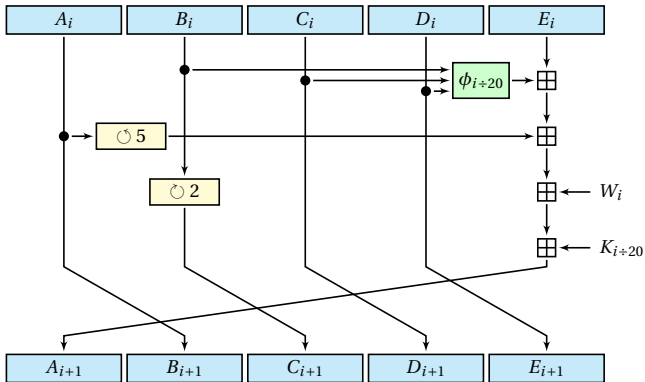
with a linear message (key) expansion:

$$W_{0 \dots 15} = M_{0 \dots 15}, \quad W_{i \geq 16} = (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus$$

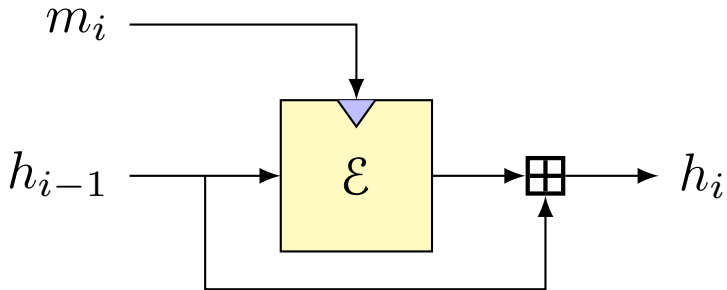
$$W_{i-16})^{\circ 1} \quad \leftarrow \text{The only difference between SHA-0 and SHA-1}$$

80 steps in total

Round function in a picture



Davies-Meyer construction in a picture



Introduction

SHA-1 in brief

History of SHA-1 attacks

Our attack

Implementation

Results

Wang collisions

SHA-1 is **not collision-resistant** (Wang, Yin, Yu, 2005)

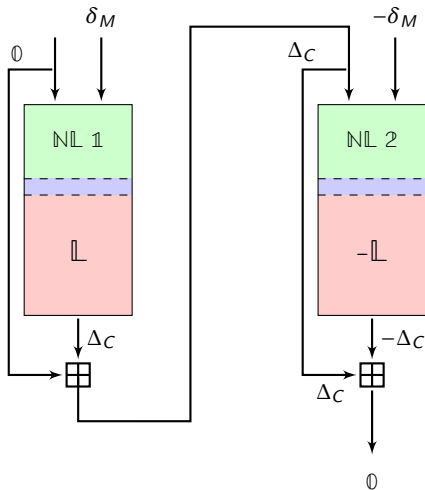
Differential collision attack

- ▶ Find a message difference that entails a good *linear* diff. path
- ▶ Construct a *non-linear* diff. path to bridge the *IV* with the linear path
- ▶ Use *message modification* to speed-up the attack
- ▶ Requires a pair of two-block messages

Attack complexity $\equiv 2^{69}$

Eventually improved to $\equiv 2^{61}$ (Stevens, 2013)

Two-block attack in a picture



SHA-1 is **much more** resistant to preimage attacks

- ▶ **No attack** on the **full function**
- ▶ Practical attacks up to $\lesssim 30$ steps ($\lesssim 37.5\%$ of SHA-1)
(De Cannière & Rechberger, 2008)
- ▶ Theoretical attacks up to **62** steps (77.5% of SHA-1)
(Espitau, Fouque, Karpman, 2015)

Introduction

SHA-1 in brief

History of SHA-1 attacks

Our attack

Implementation

Results

Let's break stuff!

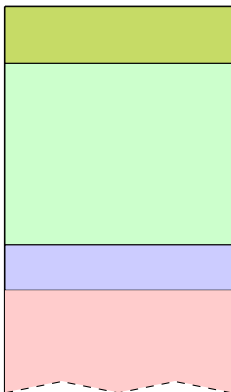


Why doing free-start again?

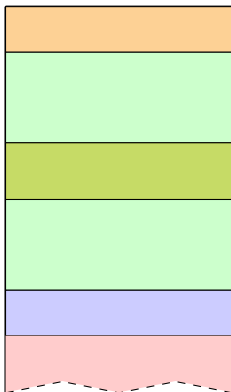
- ▶ Main reason is starting from a “middle” state + shift the message
- ▶ ⇒ Can use freedom in the message up to a later step
- ▶ ⇒ But no control on the IV value
- ▶ ⇒ Must ensure proper backward propagation

The point of free-start (in a picture)

Usual



Free-Start



But then we need to...

- 1 Find a good **linear part**
- 2 Construct a good shifted **non-linear part**
- 3 Find **accelerating techniques**

Let's do this for **80 steps!**

Linear part selection

Criteria:

- ▶ High overall probability
- ▶ No (or few) differences in last five steps (= differences in IV)
- ▶ Few differences in early message words

⇒ Not many candidates

We picked $II(59,0)$ (Manuel notation, 2011)

(This is just a shifted version of $II(55,0)$ used for 76 steps)

Linear path in a picture (part 1/2)

A

W

41: x
42:
43: x
44:
45: x
46:
47: .x
48:
49:
50: x
51:
52:
53:
54:
55: x
56:
57:
58:
59:
60:

.....x.....
.....
..x.....x.....
..x.....
.....x.....
..xx.....
.....xx.....
xxx.....
x..x.....
..xx.....x.....
x.xx.....
..x.....
..x.....
x.x.....
.....x.....
x.....
..x.....
..x.....
..x.....
x.....

Linear path in a picture (part 2/2)

A

W

61:	x	x
62:	
63:	x x	x
64:		x . x	
65:		
66:	 x	
67:	 x	
68:		
69:		
70:		
71:		
72:		
73:		
74:		x
75:		x
76:		x
77:	 x	x
78:	 x	x x
79:	 x	x x
80:	

Non-linear part construction

- ▶ Start with **prefix of high backward probability** for the first 4 steps
- ▶ Use **improved JLCA** for the rest
- ▶ ⇒ Good overall path with “few” conditions (**246** up to #30)

Non-linear path in a picture

A

```
-4: .....
-3: .....
-2: ..... ^ -
-1: 1...1.....0.....+
00: 01..0.....1.....
01: 11+^..+.....^.....+...
02: ..-11-1.1.....^.....1+110.1.0..
03: .0.0-0011.^..10...+01.01111^0.1.1
04: .111+-1+^^^+1^^^011^.-+++++-.+
05: .+.+++++.....+0-1111
06: .0.0.1.011.111.11110-0100-1.10-+
07: 1-.+.1.010100010000000111+.-.0.+
08: 0+.0.0.....0..+.-.0.1
09: .+.0.0.....0.+...^
10: .+.....+.0..
11: ...-.....
12: ...0.1.....1..
13: .1...0.....!^
14: +-.....
15: 1.1-.....!
16: +.10.1.....
```

W

```
x.+...+.....+...
..-.-.....-+...
..+.-.-.....-+...
..-.-.-.....-+.-
.....+...
.....+...
x+..++.....-+...
.....+...
x-.....+...
x.-+.-.....-+...
..-+++.....-
x++++.....-+...
..-.....-...
..+..+.....-+...
x+++.+.....-+...
.....+...
x+.....-
```

Accelerating techniques

- ▶ **Message modification**: correct bad instances
- ▶ **Neutral bits**: generate more good instances when one's found
- ▶ We choose NBs because:
 - ▶ **Easy** to find
 - ▶ **Easy** to implement
 - ▶ Good **parallelization potential** (more of that later)
 - ▶ Includes both “single” NBs and **boomerangs**

Neutral bits (with an offset)

- ▶ We start with an **offset** (remember?)
- ▶ \Rightarrow Use neutral bits with an offset too
- ▶ In our attack, **offset = 5**
 - ▶ free message words = **W5...20** instead of W0...15
- ▶ \Rightarrow Must also consider **backward propagation**

Our 60 “single” neutral bits

```
A18 :
W14  ..... xxxx .....
W15  ..... xxxx .....
A19 :
W14  ..... x . x .....
W15  ..... xxxxx .....
W16  ..... xxxxxx .....
A20 :
W15  ..... x . x .....
W16  ..... xxxxx .....
W17  ..... xxxxxxx .....
A21 :
W17  ..... xxxxx .....
W18  ..... x .....
A22 :
W18  ..... xxxxxxx .....
W19  ..... x . . x .....
A23 :
W18  ..... xxx . x .....
W19  ..... xx . x .....
W20  ..... x .....
A24 :
W19  ..... xxx .....
W20  ..... xxx .....
A25 :
W20  ..... x .....
```

Our 4 boomerangs

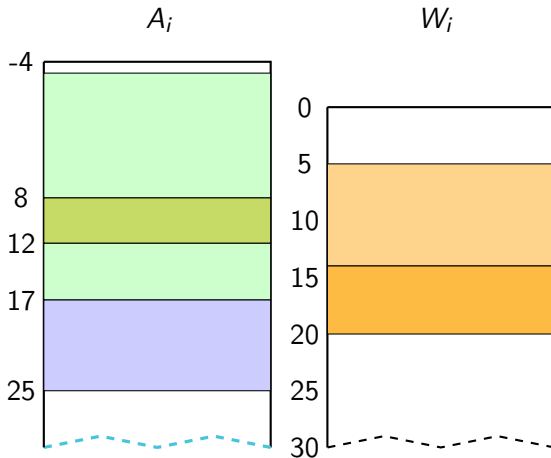
W10: BA
W11: ba DC
W12: dc
W13:
W14: a
W15: ba
W16: dc

Let's sum up

- ▶ Initialize the state with an offset
- ▶ Initialize message words with an offset
- ▶ Use neutral bits with an offset
- ▶ \Rightarrow many neutral bits up to late steps (yay)
- ▶ \Rightarrow don't know the IV in advance (duh)

- ▶ Linear path \Rightarrow differences in the IV
- ▶ Everything done in one block
- ▶ \Rightarrow Attack on the compression function

Same thing in a picture



Introduction

SHA-1 in brief

History of SHA-1 attacks

Our attack

Implementation

Results

If it's practical you must run it

- ▶ Attack expected to be practical, but still **expensive**
- ▶ Why not using **GPUs**?
- ▶ One main challenge: how to deal with the **branching**?

Target platform

- ▶ Nvidia GTX-970
- ▶ Recent, high-end, good price/performance
- ▶ $13 \times 128 = 1664$ cores @ ≈ 1 GHz
- ▶ High-level programming with CUDA
- ▶ Throughput for 32-bit arithmetic: all $1/\text{cycle/core}$ except \odot
- ▶ \approx SGD 500

Architecture imperatives

- ▶ Execution is bundled in **warps** of 32 threads
- ▶ **Single Instruction Multiple Threads:**
Control-flow **divergence is serialized** \Rightarrow **minimize branching**
- ▶ Hide latency by grouping threads into larger **blocks**
- ▶ But careful about register / memory usage

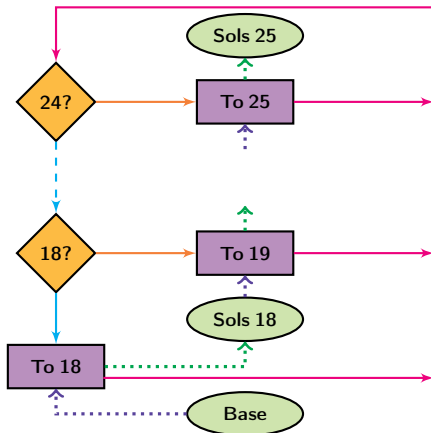
Our snippet-based approach

- 1 Store **partial solutions** up to some step in **shared buffers**
- 2 Every thread of a block loads one solution
- 3 ... tries **all neutral bits** for this step
- 4 ... stores **successful candidates** in next step buffer

Our snippet-based approach (cont.)

- 1 Base solutions up to #17 generated on CPU
- 2 Use single neutral bits up to #25 on GPU
- 3 Use boomerangs on #28 and #30 on GPU
- 4 Further checks up to #60 on GPU
- 5 Final collision check on CPU

Snippets in a picture (w/o boomerangs)



Introduction

SHA-1 in brief

History of SHA-1 attacks

Our attack

Implementation

Results

GPU results (76 steps)

- ▶ Hardware: one GTX-970
- ▶ One partial solution up to #56 per minute on average
- ▶ \Rightarrow Expected time to find a collision \approx 5 days
- ▶ Complexity $\equiv 2^{50.3}$ SHA-1 compression function

GPU v. CPU

- ▶ On one CPU core @ 3.2 GHz, the attack takes ≈ 606 days
- ▶ \Rightarrow One GPU \equiv 140 cores
- ▶ (To compare with $\equiv 40$ (Grechnikov & Adinetz, 2011))
- ▶ For raw SHA-1 computations, ratio is 320
- ▶ \Rightarrow Lose only $\times 2.3$ from the branching (not bad)

GPU results (80 steps)

- ▶ Hardware: 64 GTX-970
- ▶ \Rightarrow Expected time to find a collision \approx 10 days
- ▶ Complexity $\equiv 2^{57.5}$ SHA-1 compression function
- ▶ On Amazon Elastic C2 cost \approx USD 2K (with older GPUs)

What about a full *hash function* collision?

- ▶ Estimated complexity: $\approx 2^{61}$ (on CPU)
- ▶ GPU framework translates *swimmingly* to this case
- ▶ 512-“GPU” cluster $\Rightarrow \approx$ 50-80 days
- ▶ On Amazon Elastic C2 $\Rightarrow \approx$ USD 80-125K

For more details

Pierre Karpman, Thomas Peyrin, and Marc Stevens:
Practical Free-Start Collision Attacks on 76-step SHA-1,
CRYPTO 2015
Eprint 2015/530

Marc Stevens, Pierre Karpman, and Thomas Peyrin:
Freestart collision for full SHA-1,
EUROCRYPT 2016
Eprint 2015/967

C'est fini !

