Recent advances on SHA-1 collision attacks

Pierre Karpman

CWI, Amsterdam, The Netherlands

Séminaire MAD/CASYS — Grenoble
2017–04–06

Introduction

History of SHA-1 collision attacks

The full collision attack

Implementation

Results

Introduction

History of SHA-1 collision attacks

The full collision attack

Implementation

Results

Hash functions

Hash function

A hash function is a mapping $\mathcal{H} : \mathcal{M} \to \mathcal{D}$

In practice:

- $\mathcal{M} = \bigcup_{\ell<N}\{0,1\}^{\ell}$, $\mathcal{D} = \{0,1\}^{n}$, $N \gg n$
- Typically $N = 2^{64}$, $n \in \{128, 160, 224, 256, 384, 512\}$
- It is a keyless primitive
- 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 $\Theta(2^n)$

Second preimage resistance

Given m, find $m' \neq m$ such that $\mathcal{H}(m) = \mathcal{H}(m')$
Best generic attack is in $\Theta(2^n)$

Collision resistance

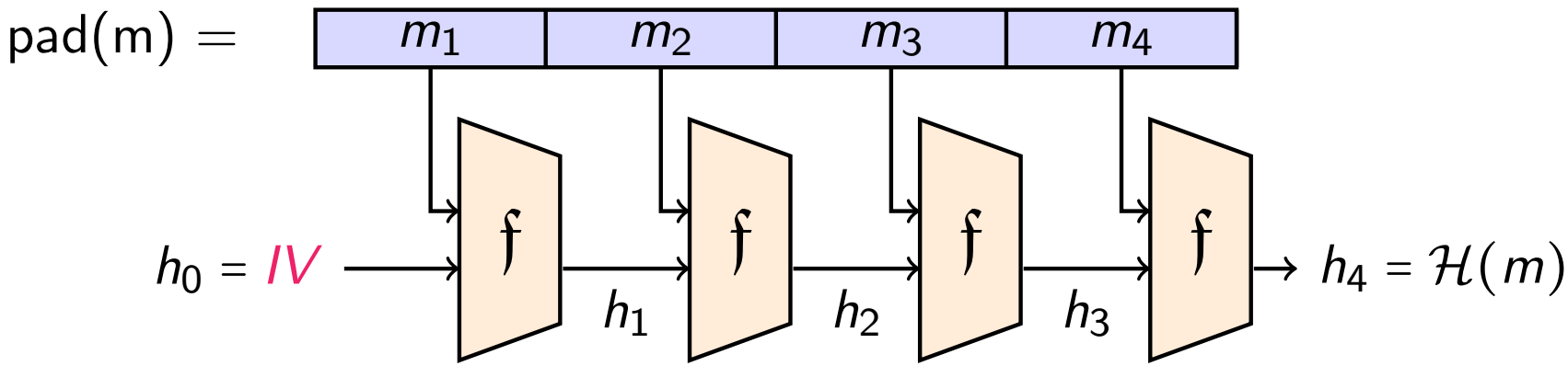
Find $m, m' \neq m$ such that $\mathcal{H}(m) = \mathcal{H}(m')$
Best generic attack is in $\Theta(2^{n/2})$

Merkle-Damgård construction

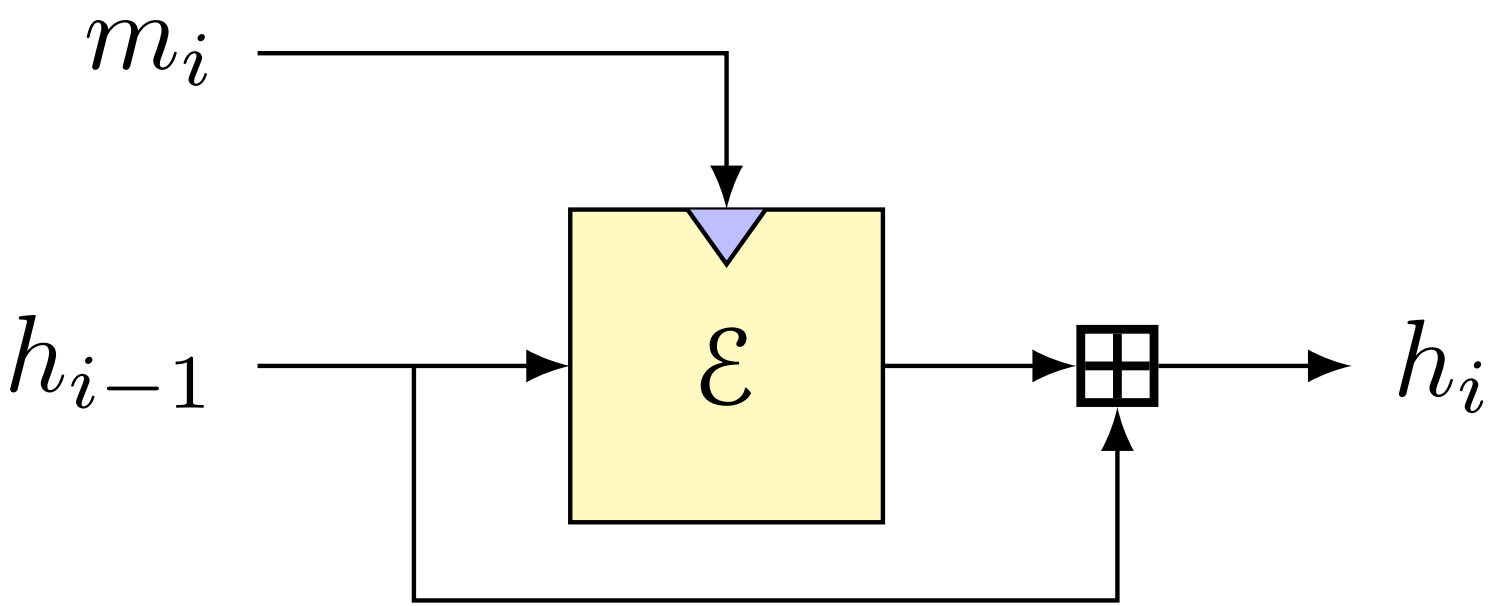
A domain of $\{0,1\}^*$ is annoying, so...

1. Start from a compression function
   $f : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$
2. Use a domain extender $\approx$
   $\mathcal{H}(m_1 \| m_2 \| \ldots \| m_\ell) = f(f(\ldots f(IV, m_1) \ldots), m_\ell)$
3. Reduce the security of $\mathcal{H}$ to the one of f
   - $\mathrm{A}(\mathcal{H}) \Rightarrow \mathrm{A}(f)$
   - $\neg\mathrm{A}(f) \Rightarrow \neg\mathrm{A}(\mathcal{H})$
   - $(\mathrm{A}(f) \Rightarrow ???)$
     - Invalidates the security reduction, tho

MD in a picture



The Davies-Meyer compression function construction



The SHA-1 hash function

- Designed by the NSA in 1995 (as an improvement of 1993's SHA-0)
- Hash size is 160 bits $\Rightarrow$ collision security should be 80 bits
- Compression function in Merkle-Damgård mode

SHA-1 compression function

Block cipher in Davies-Meyer mode

Block cipher: 5-branch ARX Feistel

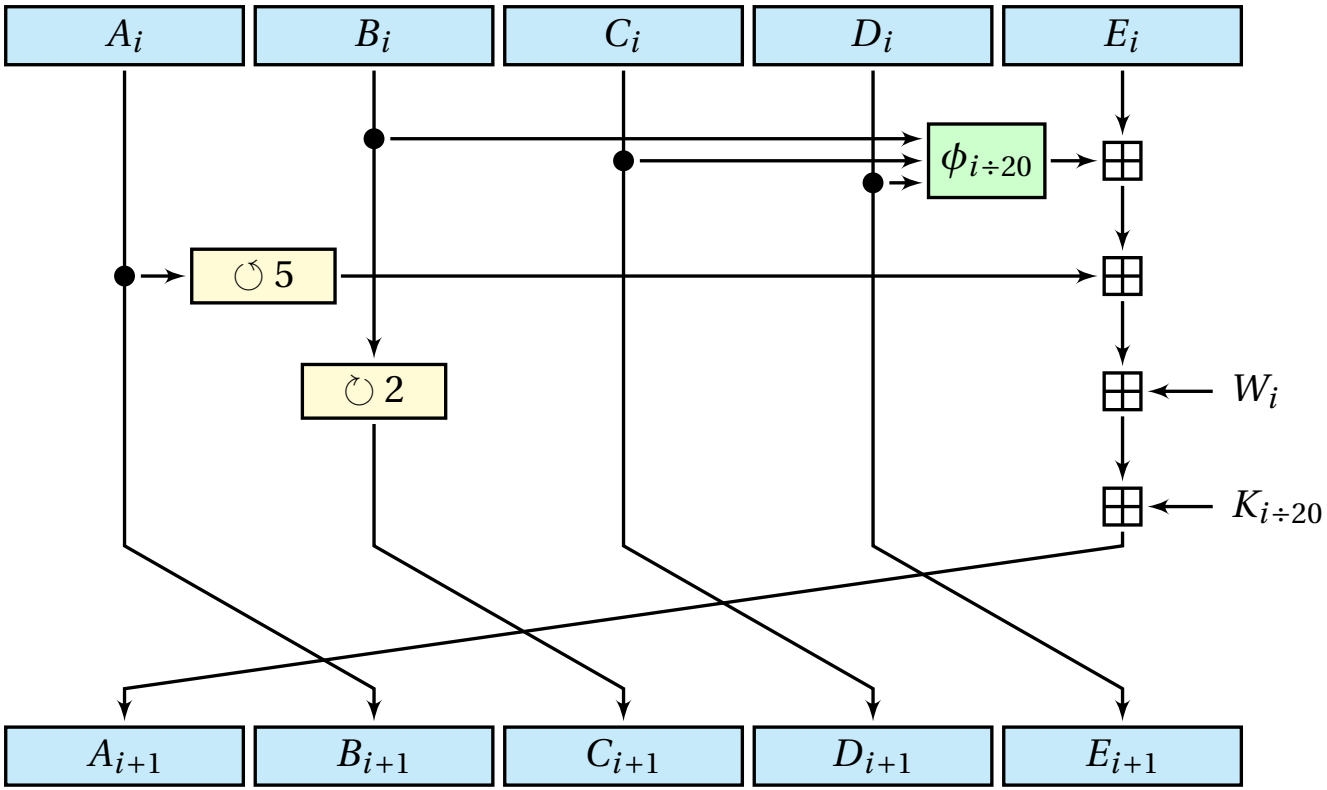
$$A_{i+1} = A_i^{\circlearrowleft 5} + \phi_{i \div 20}(A_{i-1}, A_{i-2}^{\circlearrowleft 2}, A_{i-3}^{\circlearrowleft 2}) + A_{i-4}^{\circlearrowleft 2} + W_i + K_{i \div 20}$$

with a linear message (key) expansion:

$$W_{0\ldots15} = M_{0\ldots15},\ W_{i \geq 16} = (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16})^{\circlearrowleft 1}$$

80 steps in total

SHA-1 step function in a picture



Our work: full practical attacks

2015–2016: collisions for the compression function of SHA-1 (Joint work with M. Stevens & T. Peyrin)

- 76-step, cost $\approx 2^{50.3}$ SHA-1, one inexpensive GPU is enough for fast results
- 80-step (full), cost $\approx 2^{57.5}$ SHA-1, 64 GPUs for a result in less than two weeks
  - ⁉Not "the same attack as 76-step with more computation power"

2017: collisions for the full hash function SHA-1 (Joint work with M. Stevens, E. Bursztein, A. Albertini & Y. Markov)

- Cost $\approx 2^{63}$ SHA-1, tons of GPUs needed!

Introduction

History of SHA-1 collision attacks

The full collision attack

Implementation

Results

The intuition

- We want $\mathcal{H}(m) = \mathcal{H}(m' \neq m)$
- Try a differential attack:
  - Fix the (XOR) difference $m \oplus m' = \delta$
  - Try many $\mathcal{H}(m)$, $\mathcal{H}(m \oplus \delta)$
  - $\Rightarrow$ an attack, if there is a good differential path on the state A implied by δ

First try: work with a single block

- Only consider the compression function (with fixed IV): we want $\mathfrak{f}(IV, m) = \mathfrak{f}(IV, m \oplus \delta)$
- Because of Davies-Meyer $\Rightarrow$ no state differences in $A_{76\ldots80}$
- Good enough for SHA-0 (Chabaud & Joux, 1998), but not for SHA-1

Second (er...) try: use two blocks

- Jointly exploit Merkle-Damgård and Davies-Meyer feedforward
- First block: target specific end difference Δ
  - $f(IV, m_0) = h_1$, $f(IV, m_0 \oplus \delta) = h_1 \oplus \Delta$
- Second block: cancel the difference
  - If $f(h_1, m_1) \oplus f(h_1 \oplus \Delta, m_1 \oplus \delta) = \Delta$
  - Then $\mathcal{H}(m_0 || m_1) = \mathcal{H}(m_0 \oplus \delta || m_1 \oplus \delta)$

(NB: Because SHA-1 uses modular addition in the DM feedforward, signed message differences are in fact needed)

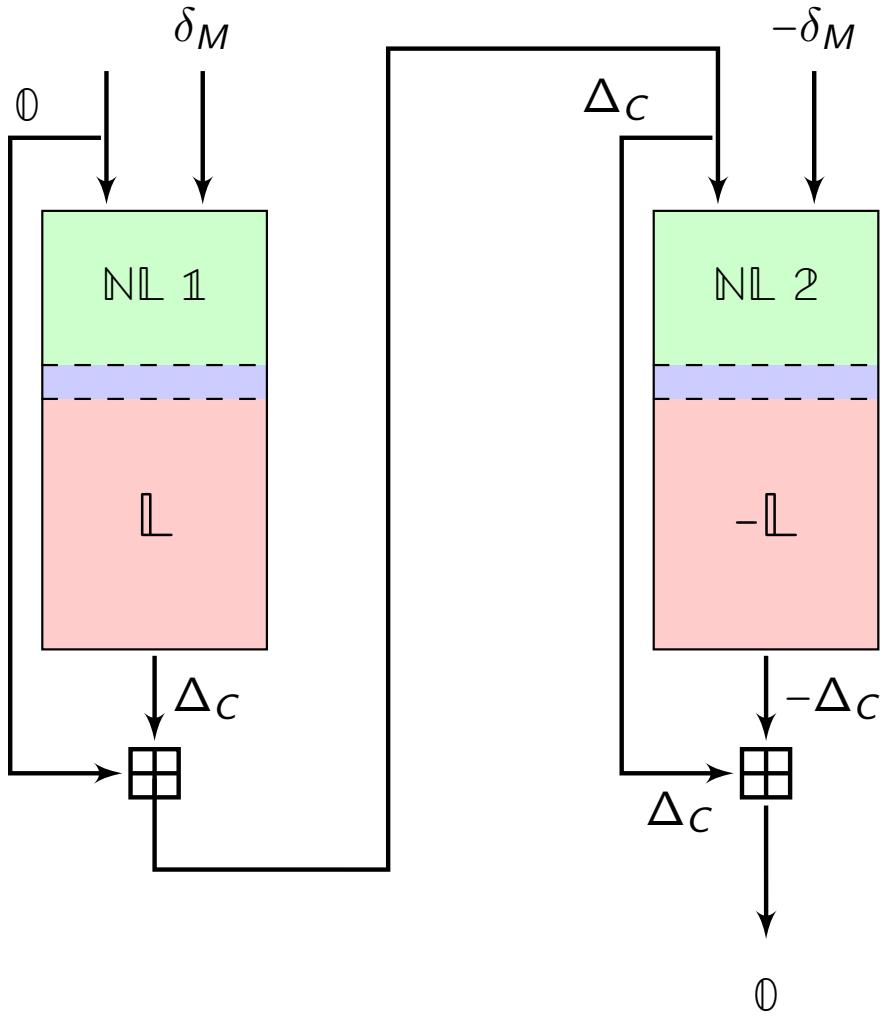
The result: Wang & al. collision attack

Major attack aspects:

- Use two-part differential paths
  - A bottom *non-linear* part
  - A top *linear* part
  - $\Rightarrow$ Allows to have two-block attacks
- Use *message modifications* to decrease complexity

$\Rightarrow$ SHA-1 is not collision-resistant (Wang, Yin & Yu, 2005)
Attack complexity $\equiv 2^{69}$ (theoretical)
Eventually improved to $\equiv 2^{61}$ (ditto, Stevens, 2013)

Two-block attack in a picture



Introduction

History of SHA-1 collision attacks

The full collision attack

Implementation

Results

Let's break stuff!

The objective

- Compute an *explicit* collision for SHA-1
- $\Rightarrow$ Definitely show that it is not secure
- Follows up explicit attacks on the compression function (less expensive, technically more challenging)
- Bonus: construct an *exploitable* collision

So we need to...

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

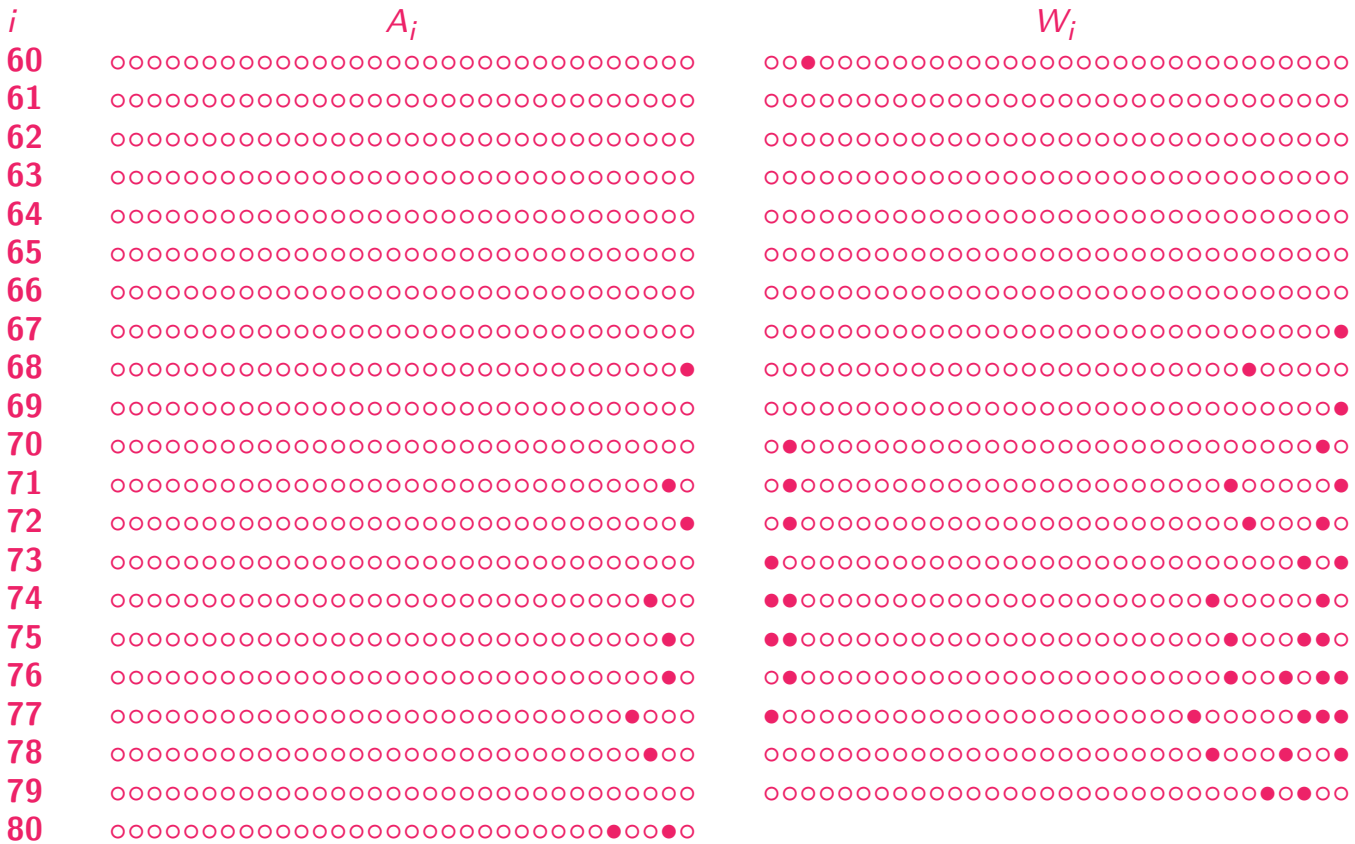
Let's do this for 80 steps!

Linear part selection

- Idea: chain *local collisions* along a low-weight codeword for the message expansion (Chabaud & Joux, 1998)
- Exploit interactions of local collisions through carry propagation and Boolean functions to increase probability (Wang & al., 2005; ...)
- Eventually, exact computation over a series of steps (Stevens, 2013)

We picked II(52,0) (Jutla & Patthak, 2005, Manuel notation, 2011)

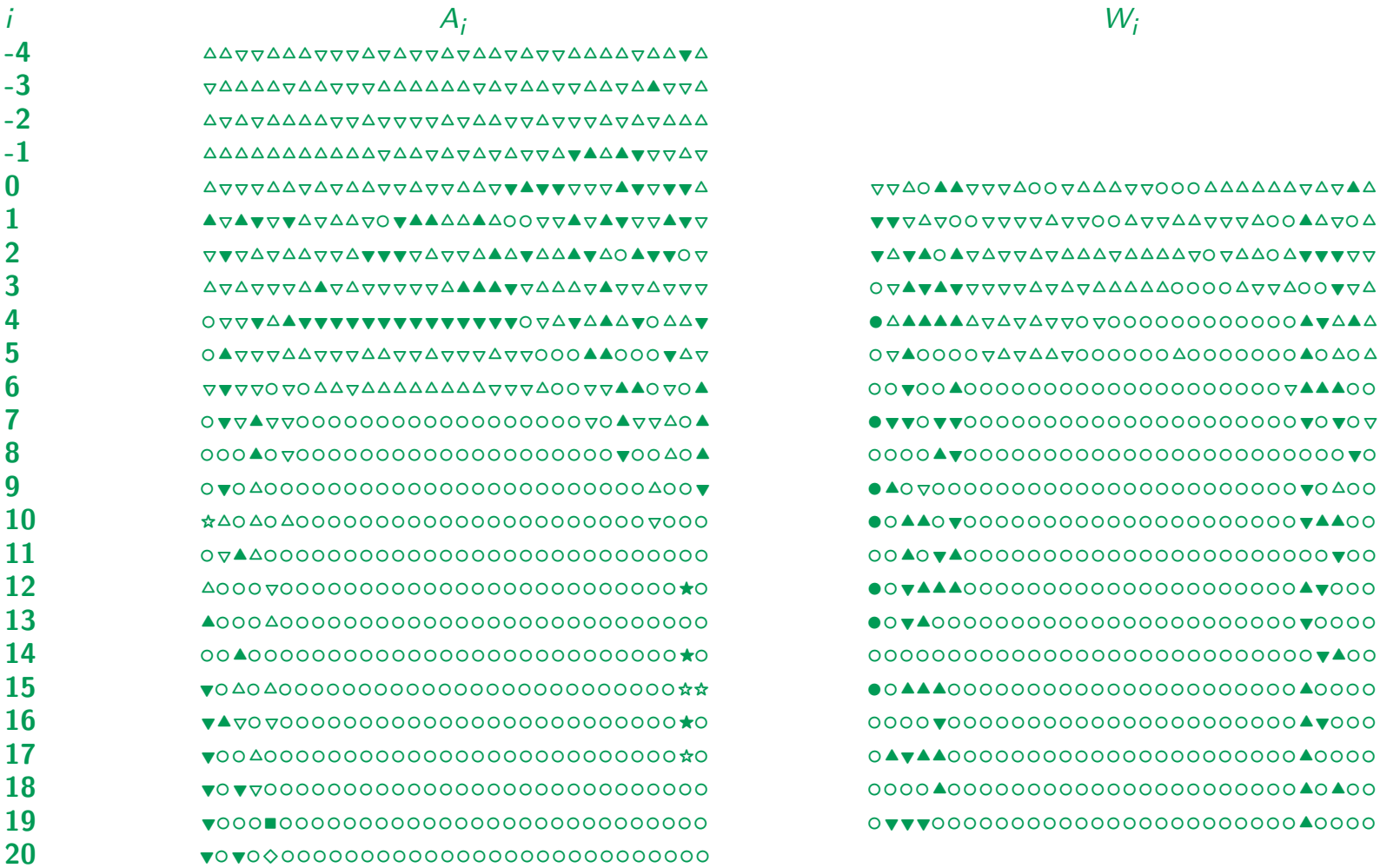
Linear path in a picture (last 20 steps)



Non-linear part construction

- Two main approaches available: guess-and-determine (De Cannière & Rechberger, 2006) and Meet-in-the-middle (Yajima & al., 2007)
- We used improved JLCA/MiTM (Stevens, 2013...)
- Used SAT to patch the second-block path
- $\Rightarrow$ Good overall path with many room for accelerating techniques

Non-linear path in a picture



Full solution example!!

Figure: Placeholder cat 1

Accelerating techniques

Attack process:

- Generate many "partial solutions": message pairs following the diff. path up to some step
- Hope that one yields a collision

To make this efficient, use:

- Message modification (Wang & al., 2005)
- Neutral bits (Biham & Chen, 2004):
  Generate more good instances when one's found
- We choose neutral bits because:
  - Easy to find
  - Easy to implement

Introduction

History of SHA-1 collision attacks

The full collision attack

Implementation

Results

If it's practical you must run it

- Attack's gonna be expensive
- Why not using GPUs?
- One main challenge: how to deal with the branching?

(NB: The first (cheaper) block of the hash function attack was computed w. CPUs)

Target platform

- Nvidia GTX-970
- Quite 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/cycle/core except $\circlearrowleft$
- $\approx$ SGD 500 (2014)

(NB: For the hash function attack, only used for prototyping. The attack was run on K20/K40/K80)

Architecture imperatives

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

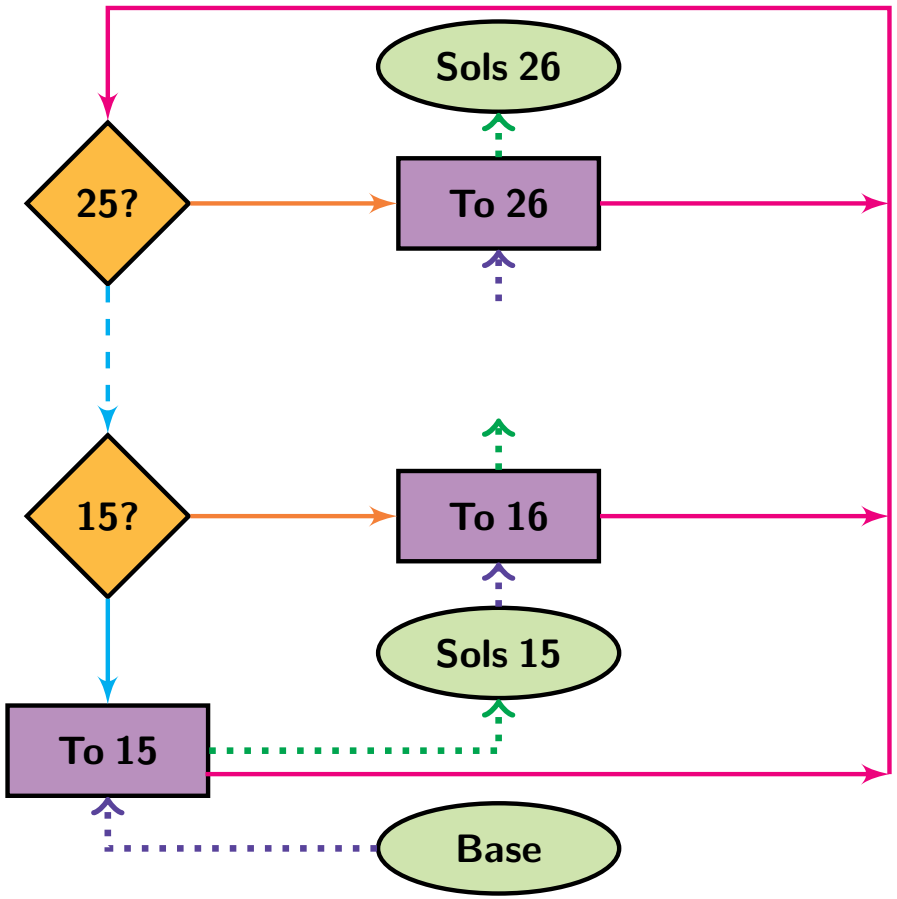
Our snippet-based approach

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 #14 generated on CPU
2. Use neutral bits up to #26 on GPU
3. Further checks up to #53 on GPU
4. Final collision check on CPU

Snippets in a picture



Introduction

History of SHA-1 collision attacks

The full collision attack

Implementation

Results

First message block

- A first block with suitable difference was found after c. 180 000 A_{61} partial solutions, taking c. 3600 core years
- Another one required c. 150 000 partial solutions (c. 3000 core years)
- Total effort: 6500 core years $\equiv 2^{60.9}$ SHA-1 compression function calls

Second message block (more expensive)

- Used GPUs for better efficiency
- A collision was found after c. 370 000 A_{61} partial solutions, taking c. 70–115 device years (depending on the GPUs)
- Complexity ≡ 2^{62}–2^{63} SHA-1 compression function calls
- Total for both blocks: ≡ 2^{63} SHA-1 calls
- (If done again, could be anywhere from 2^{62} to 2^{65} without changing the crypto part)

Comparison with other expensive attacks

- 768-bit RSA modulus factorization (Kleinjung & al., 2010): c. 2000 core years
- 768-bit Finite-Field discrete logarithm computation (Kleinjung & al., 2017): c. 5300 core years
- $\Rightarrow$ a bit less expensive, but done *only* on CPUs
- GPUs are $\approx 10\times$ more energy-efficient than CPUs (for SHA-1 attacks)
- Our (second-block) attack required less hardware and was faster (in calendar time) for less energy

Exploitable collisions (briefly)

- The idea: use byte difference in the colliding messages to point to two different payloads in a JPEG
- Embed the JPEG in a PDF

$\Rightarrow$

- Two files with the same SHA-1 hash
- Both contain the same (useful) data
- Decide which data to show depending on the colliding blocks

Demo time!!

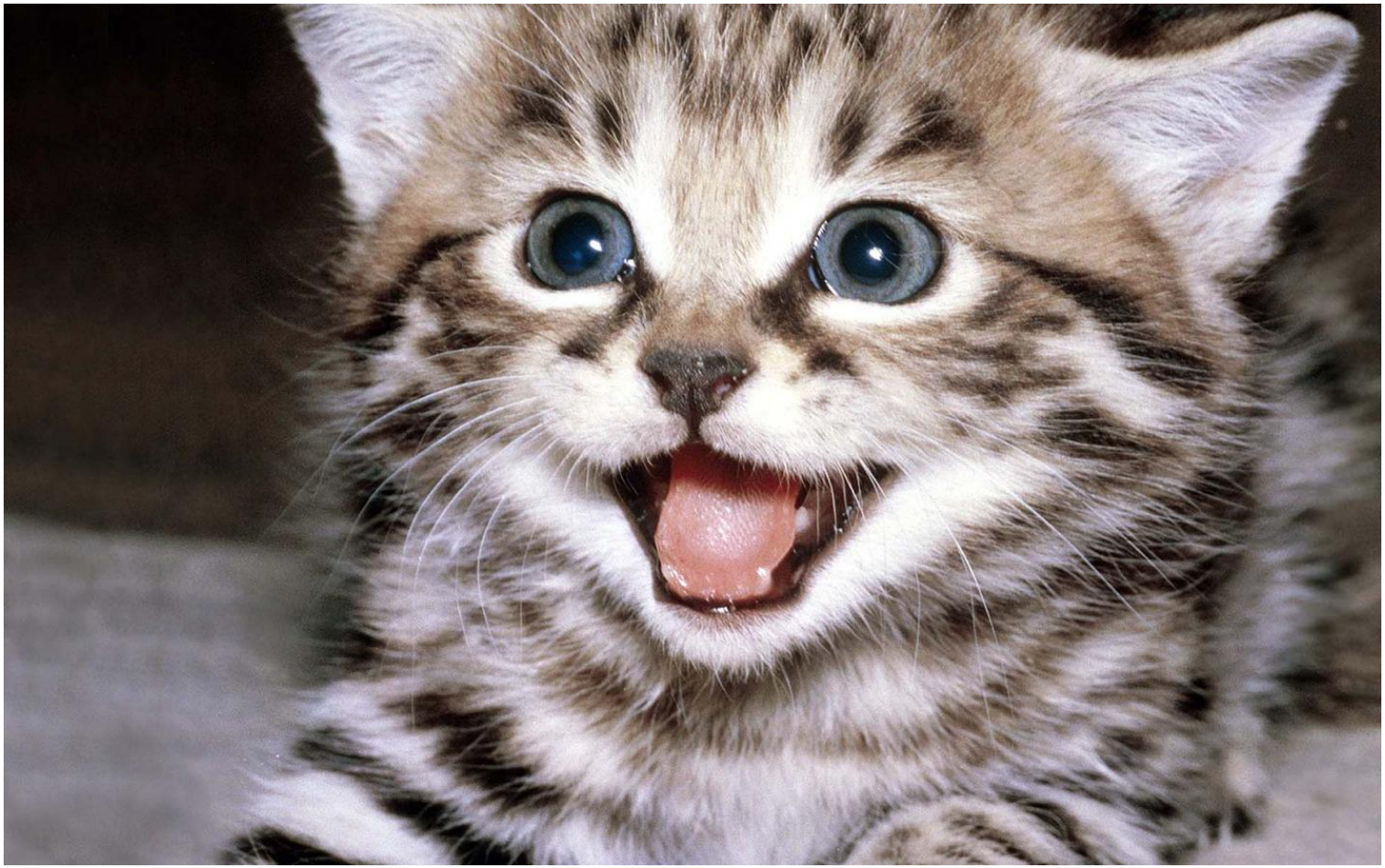


Figure: Placeholder cat 2

Some real-world impact

- SHA-1—based hash & sign of PDFs is (kind of) over
- Set-up malicious GIT servers (would require a new collision)
- Corrupt SVN repositories for free

For more details

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

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

Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, & Yarik Markov:
The first collision for full SHA-1,
Eprint 2017/190

C'est fini !