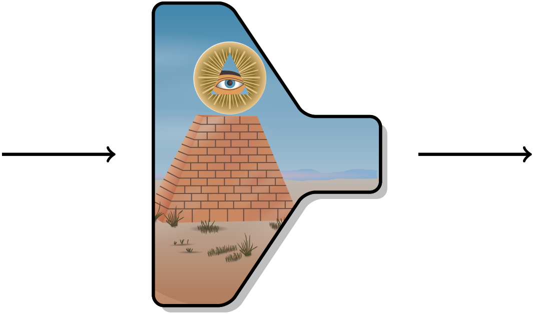
Time-memory Trade-offs for Near-collisions

Gaëtan Leurent

UCL Crypto Group

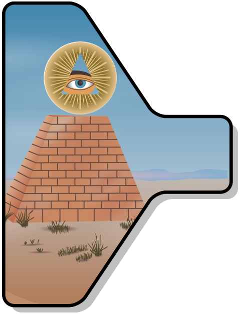
FSE 2013

An Ideal Hash Function: the Random Oracle



- Public Random Oracle
- The output can be used as a fingerprint of the document

An Ideal Hash Function: the Random Oracle



0x1d66ca77ab361c6f

- Public Random Oracle
- The output can be used as a fingerprint of the document

Concrete security goals

Preimage attack

Given F and $\overline{H}$, find M s.t. $F(M) = \overline{H}$.
Ideal security: 2^n.

Second-preimage attack

Given F and M_1, find $M_2 \neq M_1$ s.t. $F(M_1) = F(M_2)$.
Ideal security: 2^n.

Collision attack

Given F, find $M_1 \neq M_2$ s.t. $F(M_1) = F(M_2)$.
Ideal security: $2^{n/2}$.

Extra goals

Hash functions are used in many different contexts, with various assumptions:

- MAC security
- Multi-collision resistance
- Herding resistance
- Partial-collisions
- Random looking output
- Near-collisions
- ...

Near-collisions

Near-collision attack

Given F, w, find $M_1 \neq M_2$ s.t. $\|F(M_1) \oplus F(M_2)\| \leq w$.

- Relaxation of a collision attack
- Similar techniques than collision
  - Security margin
  - Turning near-collisions into collisions
- Many attack papers

Topic of this talk

What is the complexity of *generic* near-collision attacks?

State of the art

- Lower bound $2^{n/2}/\sqrt{\mathcal{B}_w(n)}$
- Memory-full algorithm $2^{n/2}/\sqrt{\mathcal{B}_w(n)}$

- Time-memory trade-off?
  - Truncate more, TMT for many collisions

    $2^\tau/\mathcal{B}_w(\tau) \approx M$ $\quad 2^{n/2}/\sqrt{\mathcal{B}_w(\tau)}$

- Memory-less algorithms
  - Truncation based $\tau \sim (2+\sqrt{2})(w-1)$ $\quad 2^{(n+\tau)/2}/\mathcal{B}_w(\tau)$
  - Covering codes based $2^{n/2}/\sqrt{\mathcal{B}_{w/2}(n)}$
  - Combine both?
  - Truncate and find truncated near-collisions with covering code

Lower bound

- After i hash evaluations, about i^2 pairs.
- Each pair is a w-near-collision with probability $\mathcal{B}_w(n)/2^n$
- Lower bound: $i^2 \approx 2^n/\mathcal{B}_w(n)$, *i.e.* $i \approx 2^{n/2}/\sqrt{\mathcal{B}_w(n)}$
  - Easier than collisions by a factor $\sqrt{\mathcal{B}_w(n)}$

Definition (size of a Hamming ball)

$$\mathcal{B}_w(n) = \#\left\{x \in \{0,1\}^n : \|x\| \le w\right\}.$$

Naive algorithm

Near-collision algorithm

for $0 \le a < i$ **do**
  $L[a] \leftarrow h(a)$ ▷ i computations
end for
for $0 \le a < b < i$ **do**
  if $\|L[a] \oplus L[b]\| \le w$ **then** ▷ i^2 comparisons
    return (a, b)
  end if
end for

- i hash computations
- i^2 comparisons, memory accesses
- i memory

Can we avoid this?

Naive algorithm

Near-collision algorithm

for $0 \leq a < i$ **do**
    $L[a] \leftarrow h(a)$ ▷ i computations
end for
for $0 \leq a < b < i$ **do**
    if $\|L[a] \oplus L[b]\| \leq w$ **then** ▷ i^2 comparisons
        return (a, b)
    end if
end for

- i hash computations
- i^2 comparisons, memory accesses
- i memory

Can we avoid this?

Naive algorithm

Near-collision algorithm

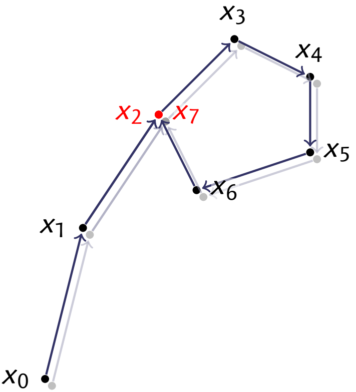
for $0 \leq a < i$ **do**
    $L[a] \leftarrow h(a)$ ▷ i computations
end for
for $0 \leq a < b < i$ **do**
    if $\|L[a] \oplus L[b]\| \leq w$ **then** ▷ i^2 comparisons
        return (a, b)
    end if
end for

- i hash computations
- i^2 comparisons, memory accesses
- i memory

Can we avoid this?

Memoryless collision finding

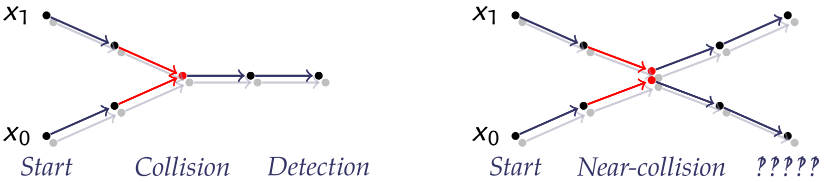
Memoryless algorithms are known for *full* collisions: Pollard's rho



- Iterate h: $x_i = f(x_{i-1})$
- Collision after $\approx 2^{n/2}$ iterations
  - Iteration cycles
- Memoryless cycle detection
  - Floyd (tortoise and hare)
  - Brent
  - Nivasch
  - Distinguished points
  - ...

Memoryless near-collisions algorithms

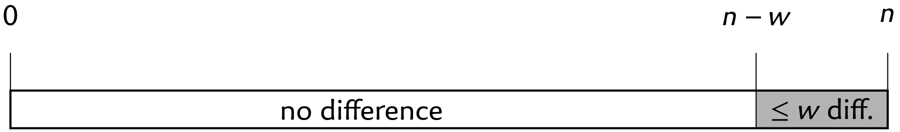
- Memoryless collision algorithms based on iterating chains
- Collisions can be detected later in the chain



- This doesn't work for near-collision
    - New approaches needed

Using truncation

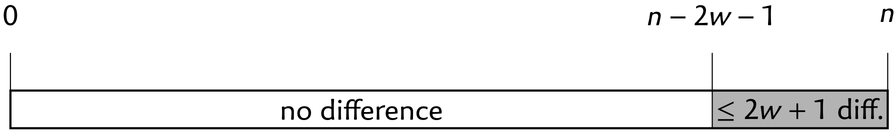
1. Truncate w bits
2. Find $n - w$-bit collision (memoryless)
3. Gives w-near-collision for the full output



- Complexity: $2^{(n-w)/2}$

Using truncation

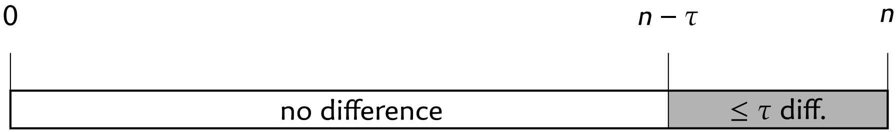
1. Truncate $2w + 1$ bits
2. Find $n - 2w - 1$-bit collisions (memoryless)
3. Gives w-near collision with probability ½



- Complexity: $2^{(n-2w-1)/2} \times 2$

Using truncation

1. Truncate τ bits
2. Find $n - \tau$-bit collisions (memoryless)
3. Gives w-near collision with probability $\mathcal{B}_w(\tau)/2^\tau$



- Complexity: $2^{(n+\tau)/2}/\mathcal{B}_w(\tau)$
- Optimal $\tau \sim (2 + \sqrt{2})(w - 1)$ [Lamberger & Teufl, IPL 2013]

Generalization

1. Build a function f so that

$$f(x) = f(y) \Rightarrow \|x \oplus y\| \leq w$$

2. Find collisions in $f \circ h$ (memoryless)
3. Gives a w-near-collision

$$f(h(x)) = f(h(y)) \Rightarrow \|h(x) \oplus h(y)\| \leq w$$

- Use a covering code [Lamberger & Rijmen]
  - Covering radius R, decoding function f:
    $$\|x \oplus f(x)\| \leq R$$
  - $f(x) = f(y) \Rightarrow$
    $$\|x \oplus y\| \leq \|x \oplus f(x)\| + \|y \oplus f(y)\| \leq 2R$$

Outline

- Lower bound $2^{n/2}/\sqrt{\mathcal{B}_w(n)}$
- Memory-full algorithm $2^{n/2}/\sqrt{\mathcal{B}_w(n)}$
- Time-memory trade-off?
  - Truncate more, TMT for many collisions
    $2^\tau/\mathcal{B}_w(\tau) \approx M$ $\quad 2^{n/2}/\sqrt{\mathcal{B}_w(\tau)}$
- Memory-less algorithms
  - Truncation based $\tau \sim (2+\sqrt{2})(w-1)$ $\quad 2^{(n+\tau)/2}/\mathcal{B}_w(\tau)$
  - Covering codes based $2^{n/2}/\sqrt{\mathcal{B}_{w/2}(n)}$
  - Combine both?
  - Truncate and find truncated near-collisions with covering code

Outline

- Lower bound $2^{n/2}/\sqrt{\mathcal{B}_w(n)}$
- Memory-full algorithm $2^{n/2}/\sqrt{\mathcal{B}_w(n)}$
- Time-memory trade-off?
  - Truncate more, TMT for many collisions
    $2^{\tau}/\mathcal{B}_w(\tau) \approx M$ $2^{n/2}/\sqrt{\mathcal{B}_w(\tau)}$
- Memory-less algorithms
  - Truncation based $\tau \sim (2+\sqrt{2})(w-1)$ $2^{(n+\tau)/2}/\mathcal{B}_w(\tau)$
  - Covering codes based $2^{n/2}/\sqrt{\mathcal{B}_{w/2}(n)}$
  - Combine both?
  - Truncate and find truncated near-collisions with covering code

Another look at truncation

Near-collision using truncation by τ bits

- $i(\tau) = 2^{\tau}/\mathcal{B}_w(\tau)$ collisions needed. Increase with τ
- One truncated collision costs $2^{n-\tau}$. Decrease with τ

Can we do better than $i \cdot 2^{(n-\tau)/2}$ to find i collisions?

- Memoryless: no
- With memory: yes, keep state after first collision

$\Rightarrow$ Improved near-collision algorithms

Another look at truncation

Near-collision using truncation by τ bits

- $i(\tau) = 2^{\tau}/\mathcal{B}_w(\tau)$ collisions needed. Increase with τ
- One truncated collision costs $2^{n-\tau}$. Decrease with τ

Can we do better than $i \cdot 2^{(n-\tau)/2}$ to find i collisions?

- Memoryless: no
- With memory: yes, keep state after first collision

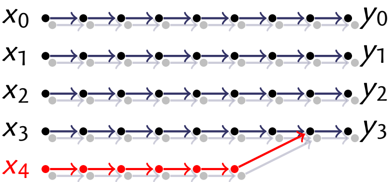
$\Rightarrow$ Improved near-collision algorithms

Finding several collisions

Parallel collision search [van Oorschot & Wiener, JoC 1999]

Definition (distinguished point)

y distinguished iff $y \bmod \theta^{-1} = 0$



M chains cover $\approx M/\theta$ points

1. Compute chains $x \rightsquigarrow y$
   Stop when y distinguished
2. If $y \in \{y_i\}$, new collision found
3. Store (x, y)

Finding several collisions

Complexity: [van Oorschot & Wiener, JoC 1999]

- Small number of collisions *i.e.* $i \ll M$

$$C_{small} = \sqrt{\pi/2} \cdot \sqrt{2^n i}$$

Speedup: $\sqrt{i}$ (optimal)

- Large number of collisions *i.e.* $i \gg M$.

$$C_{large} = 5\sqrt{2^n/M} \cdot i$$

Speedup: $\sqrt{M}/4$

- Combining:

$$C \approx C_{small} + C_{large} = \left(\sqrt{\frac{\pi}{2}} + 5\sqrt{\frac{i}{M}}\right)\sqrt{2^n i}$$

TM Trade-off for Near-collisions using Truncation

- Truncate τ bits.
- $i(\tau) = 2^\tau / \mathcal{B}_w(\tau)$ collisions needed.

Small τ, $i(\tau) \ll M$

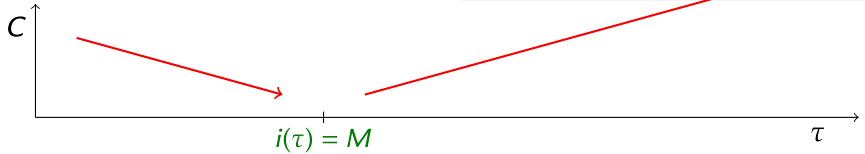
$C_{small} = \sqrt{\pi/2} \cdot 2^{n/2} / \sqrt{\mathcal{B}_w(\tau)}$

Decreasing

Large τ, $i(\tau) \gg M$

$C_{large} = 5 \cdot 2^{n/2+\tau/2} / \mathcal{B}_w(\tau) \sqrt{M}$

Increasing



- Optimum for $i(\tau) \approx M$

$C \approx 2^{n/2} / \sqrt{\mathcal{B}_w(\tau)}$

Comparison: $n = 128, w = 10$

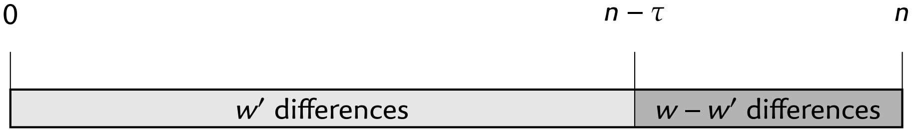
- Lower bounds
  - $C \geq 2^{n/2}/\sqrt{\mathcal{B}_w(n)}$ *(memory-full)* $C \geq 2^{40.1}$
- Covering codes
  - $C \geq 2^{n/2}/\sqrt{\mathcal{B}_{w/2}(n)}$ *for code-based* $C \geq 2^{50}$
  - Best code known $C = 2^{52.5}$
- Truncation, memoryless, $\tau = 2w + 1$ — $\tau = 21$
  - $C \approx 2^{(n-\tau)/2} \times 2$ $C = 2^{54.5}$
- Truncation, memoryless, optimal
  - $\tau \sim (2 + \sqrt{2})(w - 1)$ $\tau = 32$
  - $C \approx 2^{(n+\tau)/2}/\mathcal{B}_w(\tau)$ $C = 2^{53.3}$
- Truncation, with 1GB memory
  - $2^\tau/\mathcal{B}_w(\tau) \approx M$ $\tau = 56$
  - $C \approx 2^{n/2}/\sqrt{\mathcal{B}_w(\tau)}$ $C = 2^{47}$

Outline

- Lower bound $2^{n/2}/\sqrt{\mathcal{B}_w(n)}$
- Memory-full algorithm $2^{n/2}/\sqrt{\mathcal{B}_w(n)}$
- Time-memory trade-off?
  - Truncate more, TMT for many collisions
    $2^\tau/\mathcal{B}_w(\tau) \approx M$ $\quad$ $2^{n/2}/\sqrt{\mathcal{B}_w(\tau)}$
- Memory-less algorithms
  - Truncation based $\tau \sim (2+\sqrt{2})(w-1)$ $\quad$ $2^{(n+\tau)/2}/\mathcal{B}_w(\tau)$
  - Covering codes based $2^{n/2}/\sqrt{\mathcal{B}_{w/2}(n)}$
  - Combine both?
  - Truncate and find truncated near-collisions with covering code

New approach

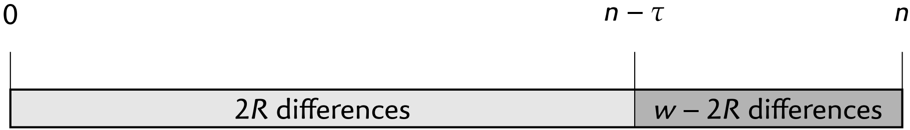
1. Truncate τ bits
2. Find $n-\tau$-bit w'-near-collisions
3. Gives w-near collision with some probability



- Large parameter space w, τ
- Special cases:
  - $\tau = 0$: coding based algorithm
  - $w' = 0$: truncation based algorithm
- Use a covering code to find near-collisions in the truncation

New approach

1. Truncate τ bits
2. Find $n - \tau$-bit w'-near-collisions
3. Gives w-near collision with some probability



- Large parameter space (R, τ)
- Special cases:
  - $\tau = 0$: coding based algorithm
  - $R = 0$: truncation based algorithm
- Use a covering code to find near-collisions in the truncation

Complexity

Analysis:

- No closed formula for parameter choice ☹
- Exhaustive search over τ and R, compute complexity

128 bits	M-Full*	Time-memory trade-off (τ, R)			Covr. codes		Trunc.
		2^{16} (1MB)	2^{26} (1GB)	2^{36} (1TB)	bnd	best	τ=2w−1
$w = 2$	57.5	60.5 (1,1)	60.0 (25,0)	59.5 (35,0)	60.5	60.5	62.0
$w = 4$	52.3	57.6 (17,1)	56.5 (27,1)	55.6 (44,0)	57.5	58.0	60.0
$w = 6$	47.8	54.5 (19,2)	53.1 (35,1)	52.0 (46,1)	54.8	56.0	58.0
$w = 8$	43.8	51.6 (26,2)	49.8 (43,1)	48.5 (54,1)	52.3	54.0	56.0
$w = 10$	40.1	48.7 (33,2)	46.7 (50,1)	45.2 (62,1)	50.0	52.5	54.0

* Number of hash function evaluation. More than $2^{n/2}$ memory accesses.

Summary

1. Time-memory trade-off
   - Finding i collisions costs less than $i \cdot 2^{n/2}$
   - Use larger τ

2. Combine truncation and covering codes
   - Find near-collisions in truncated function

$\Rightarrow$ Significant improvement for practical parameters

10-near-collision for a 128-bit hash

Complexity in $2^{45.2}$ using 1TB, versus $2^{52.5}$ memoryless.
Lower bound: $2^{40.1}$; reduce the gap for practical attacks.

Thanks

Questions?

With the support of ERC project CRASH

European Research Council
Established by the European Commission
Supporting top researchers from anywhere in the world