

# Improved generic algorithms for 3-collisions

Antoine Joux<sup>1</sup>   Stefan Lucks<sup>2</sup>

DGA and Université de Versailles Saint-Quentin-en-Yvelines  
UVSQ PRISM, 45 avenue des États-Unis, F-78035, Versailles CEDEX, France  
[antoine.joux@m4x.org](mailto:antoine.joux@m4x.org)

BAUHAUS-UNIVERSITÄT WEIMAR, 99423 Weimar, Germany  
[Stefan.Lucks@uni-weimar.de](mailto:Stefan.Lucks@uni-weimar.de)

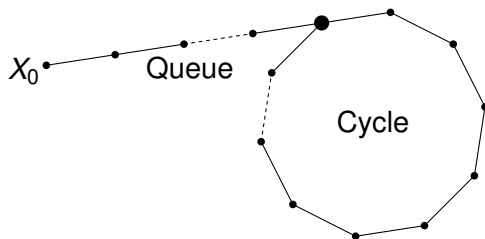
Asiacrypt, Tokyo, December 9<sup>th</sup>, 2009

# Collision resistance

- ▶ Essential property for hash functions
- ▶ Let  $H$  be a  $n$ -bit hash function
  - ▶ It should be hard to find  $M \neq M'$  such that  $H(M) = H(M')$ :
    - ▶ Best generic attack based on birthday paradox
    - ▶ Requires  $2^{n/2}$  calls to  $H$
- ▶ Elementary algorithm;
  - ▶ Evaluate  $H$  at  $2^{n/2}$  points
  - ▶ Sort and read list to find collision

# Reminder: Memoryless algorithms for collisions

- ▶ Based on cycle finding on sequence  $X_{n+1} = H(X_n)$ 
  - ▶ Brent
  - ▶ Floyd
  - ▶ Nivasch



# Nivasch's method

- ▶ Compute sequence  $X_{n+1} = F(X_n)$
- ▶ Keep list of “smallest” encountered values:

$$L_0 < L_1 < \dots < L_k$$

- ▶ Stored as pairs  $(L_i, N_i)$  where  $X_{N_i} = L_i$
- ▶ Insertion rule:
  - ▶ If  $L_i < X_n < L_{i+1}$ , let  $(L_{i+1}, N_{i+1}) \leftarrow (X_n, n)$   
Erase the end of list  $L$
  - ▶ If  $L_i = X_n$ , cycle of length  $n - n_i$  discovered

Entrance of the cycle  $\Rightarrow$  Collision

# What about Multicollisions?

- ▶ A  $k$ -collision for  $F$  is  $k$ -tuple of distinct values such that:

$$F(a_1) = F(a_2) = \dots = F(a_k)$$

- ▶ Hard to find in general. Well known that:

$$k! N^{(k-1)/k}$$

evaluations of  $F$  are required.

# Elementary algorithm for $k$ -collisions

- ▶ Compute  $F$  at  $k! N^{(k-1)/k}$  distinct random points
- ▶ Sort
- ▶ Search for  $k$  consecutive equal values

Expensive in terms of memory.  
Cannot be easily parallelized.

# The case of 3-collisions

- ▶ Elementary algorithm  $\Rightarrow$  time:  $N^{2/3}$ , memory:  $N^{2/3}$
- ▶ No useful parallelization
- ▶ Note that different time/memory tradeoff are possible
  - ▶ Typical tradeoff: Time  $N^{3/4}$ , memory  $N^{1/2}$

Can we do better ?

## A first improvement

- ▶ Use cycle finding collision algorithm on  $F \circ \pi$
- ▶ Obtain  $N^{1/4}$  different collisions.
- ▶ Compute  $F$  at  $N^{3/4}$  random points

Collision between the two phases  $\Rightarrow$  3-Collision.

- ▶ Time:  $N^{3/4}$ , memory:  $N^{1/4}$
- ▶ Can be parallelized, but requires  $N^{1/4}$  mem/proc

## Going further

- ▶ Goal: obtain  $N^{1/3}$  collisions in time  $N^{2/3}$
- ▶ Then compute  $F$  at  $N^{2/3}$  random points

Collision between the two phases  $\Rightarrow$  3-Collision.

- ▶ Time:  $N^{2/3}$ , memory:  $N^{1/3}$
- ▶ Can be parallelized, but would require  $N^{1/3}$  mem/proc

How to reach this goal?

# Cheaper collisions

- ▶ Phase 1:
  - ▶ Construct  $N^{1/3}$  random chains  $(s, f)$  where  $f = F^{(N^{1/3})}(s)$
  - ▶ Sort by  $f$  values
- ▶ Phase 2:
  - ▶ Construct new random chains
  - ▶ If  $F^{(i)}(x)$  is a stored  $f$  go to phase 3
- ▶ Phase 3:
  - ▶ We know that  $F^{(N^{1/3})}(s) = F^{(i)}(x)$
  - ▶ Thus  $F^{(i)}(F^{(N^{1/3}-i)}(s)) = F^{(i)}(x)$
  - ▶ Usually  $\Rightarrow$  Collision

Yields  $N^{1/3}$  collisions in time  $N^{2/3}$   
using memory  $N^{1/3}$

# Parallelizable approach

- ▶ Based on *Distinguished points*
- ▶ A point  $x$  is distinguished if  $x < M$  ( $M \approx N^{2/3}$ )
- ▶ Assume  $N_p \approx N^{1/3}$  processors
- ▶ In Phase 1, each processor:
  - ▶ Chooses a random  $x_0$
  - ▶ Computes  $x_{i+1} = F(x_i)$
  - ▶ Stops when  $x_i < M$  (or aborts if  $i > 20 N^{1/3}$ )
  - ▶ Sends  $(x_i, x_0, i)$  to processor numbered  $x_i \pmod{N_p}$
- ▶ In Phase 2, each processor:
  - ▶ Collects triples  $(D, X, \ell)$  with common  $D$  value
  - ▶ Recomputes synchronized chains
  - ▶ Wins if simultaneous collision of 3 chains

3-collisions in time  $N^{1/3}$   
using  $N^{1/3}$  processors  
(Constant memory/processor)

# With fewer processors

3-collisions in time  $N^{2/3-\theta}$   
using  $N^\theta$  processors

( $N^{1/3-\theta}$  memory/processor)

Is it optimal ?

# Example

- ▶ Function based on DES:

$$F(x) = \text{DES}_{K_1}(x) \oplus \text{DES}_{K_2}(x),$$

with  $K_1 = (3322110077665544)_{16}$  and  
 $K_2 = (3b2a19087f6e5d4c)_{16}$ .

- ▶ Example computed on a mix of CPUs and graphic cards <sup>1</sup>
- ▶ We found:

$$\begin{aligned} F(d332b9ba5e5a7d4e) &= F(51b8095db532afcc) \\ &= F(b084dc15dce042ab) \end{aligned}$$

---

<sup>1</sup>Thank to CEA/DAM for the computing power.

# More details

- ▶ Parameters:
  - ▶  $M = 2^{44} \Rightarrow$  average length of chains:  $2^{20}$
  - ▶ We found 35 447 322 chains  $\Rightarrow$   
3 078 699 groups of 3 or more chains (max. 36)
- ▶ First phase:
  - ▶ Mix of CPUs and graphic cards
    - ▶ 32 Intel Xeon at 2.8 GHz
    - ▶ 8 Nvidia CUDA cards (Tesla type)
  - ▶ Running time equivalent to:
    - ▶ 94 days on a single Xeon **or**
    - ▶ 11.5 days on a single Tesla
- ▶ Second phase:
  - ▶ Xeon CPUs only
  - ▶ Running time:
    - ▶ 18 days on a single Xeon
- ▶ All in all, 3 triple collisions found

# Generalization to k-collisions

- ▶ Elementary algorithm  $\Rightarrow$  time:  $N^{(k-1)/k}$ , memory:  $N^{(k-1)/k}$

# Generalization to $k$ -collisions

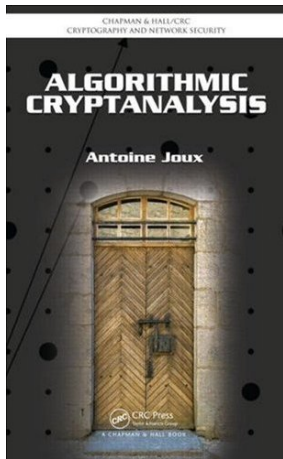
- ▶ Elementary algorithm  $\Rightarrow$  time:  $N^{(k-1)/k}$ , memory:  $N^{(k-1)/k}$
- ▶ Generalization of distinguished points approach  
( $\theta \leq (k-2)/k$ )

$k$ -collisions in time  $N^{(k-1)/k-\theta}$   
using  $N^\theta$  processors

( $N^{(k-2)/k-\theta}$  memory/processor)

# A possible application

- ▶ Improving Ferguson-Lucks attack on AURORA (eprint 2009/113)
- ▶ Attack remains unpractical



## Conclusion