A New Algorithm for the Unbalanced Meet-in-the-Middle Problem

Ivica Nikolić
(joint with Yu Sasaki)

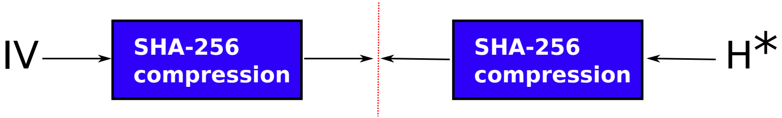
NTU, Singapore

Unbalanced Meet-in-the-Middle

Example: From pseudo-preimage to preimage attack on SHA-256

Let the compression function be invertible in 2^{64}

- Store 2^{96} preimages for the second compression function
- Generate 2^{160} images for the first
- Produce a collision in the middle

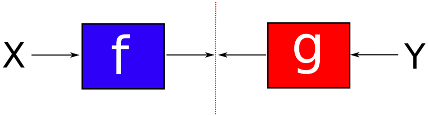


MITM $\neq$ meeting in the middle

- Diffie–Hellman introduced MITM to attack Double-DES. There, the two functions were indeed "meeting in the middle"
- However, today MITM has a different, more general meaning
- Example, MITM attacks on AES have nothing to do with "meeting in the middle".

MITM = Collision search

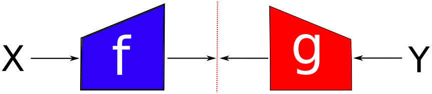
- MITM attack is synonym for collision search
- So, instead of MITM we can talk about collisions between two functions $f(x)$ and $g(y)$



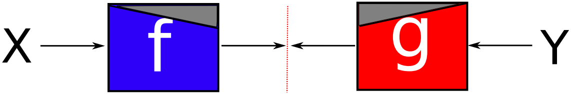
Collision types

We can differentiate two types of collisions between f and g

1. f, g have range **larger** than domain.



2. f, g have range **not larger** than domain.



Our target: Unbalanced Collisions

We deal only with the case 2. Furthermore, to simplify, we focus only on collision search between two n-bit functions f, g:

$$f : \{0,1\}^n \rightarrow \{0,1\}^n$$
$$g : \{0,1\}^n \rightarrow \{0,1\}^n$$

Unbalanced collisions – g is R times more "expensive" than f (in the previous example of SHA-256, $R = 2^{64}$)

The balanced case

When $R = 1$ (f, g have the same cost) then

- use Floyd's cycle finding algorithm
- it requires time $T = 2^{\frac{n}{2}} = \sqrt{N}$
- it requires **negligible** memory

The unbalanced case

When $R > 1$, then

- use MITM
- Store $\sqrt{\frac{N}{R}}$ images of g (in time $R\sqrt{\frac{N}{R}} = \sqrt{RN}$)
- Produce around $\sqrt{RN}$ images of f and check for collision
- Success because $\sqrt{\frac{N}{R}}\sqrt{RN} = N$

Time: $\sqrt{RN}$
Memory: $\sqrt{\frac{N}{R}}$

The unbalanced case - Tradeoff

The standard MITM algorithm allows a tradeoff

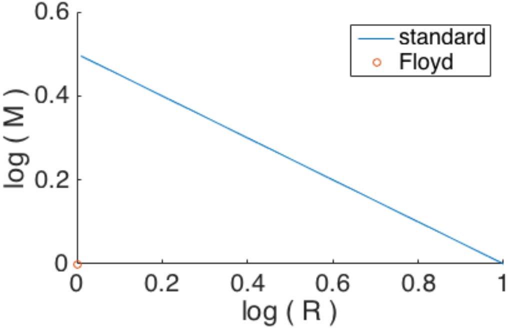
$$TM = N,$$

where $T \geq \sqrt{RN}$.

Why the standard MITM algorithm can be bad

Standard MITM :

- Huge jump of memory requirement when R goes beyond 1
- **Weird: the smaller the R, the larger the memory requirement**



Ideas

New algorithm combines 2 ideas:

1. Unbalanced interleaving
2. van Oorschot-Wiener parallel collision search

Unbalanced interleaving

Balanced interleaving

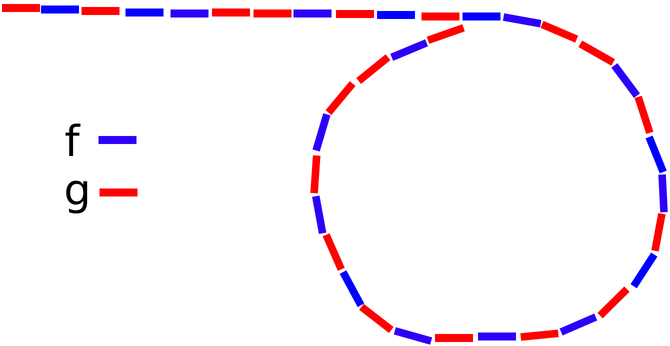
Floyd's algorithm used for collision search of 2 balanced functions selects the used function with equal probability. i.e. it finds a collision for $H(x)$ defined as

$$H(x) = \begin{cases} f(x) & \text{if } \sigma(x) = 0 \\ g(x) & \text{if } \sigma(x) = 1 \end{cases}$$

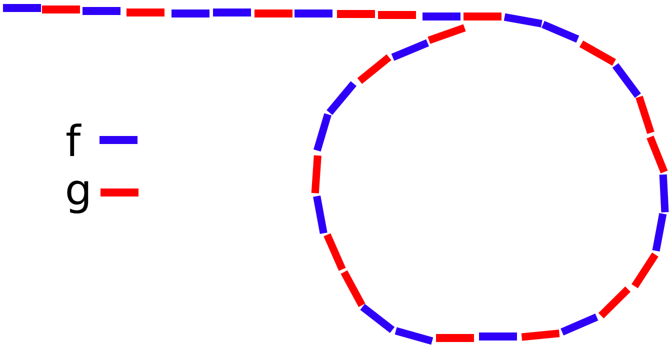
$\sigma(x)$ outputs 0 or 1, with equal probability

Collisions for $H(x)$ is collision between f, g with probability $\frac{1}{2}$
$\implies$ repeat the search 2 times

Balanced interleaving - Floyd's cycle finding algorithm



Balanced interleaving - Floyd's cycle finding algorithm



Unbalanced interleaving

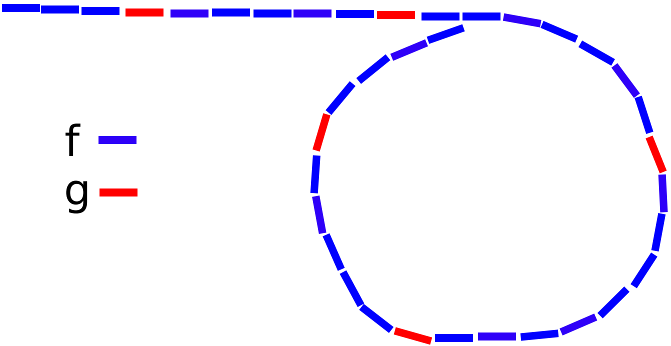
Unbalanced interleaving
Define $H(x)$ as

$$H(x) = \begin{cases} f(x) & \text{if } \sigma(x) = 0 \\ g(x) & \text{if } \sigma(x) = 1 \end{cases}$$

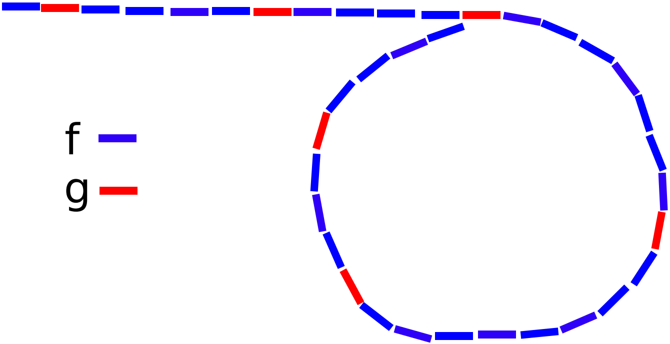
$\sigma(x)$ outputs 0 around R times more often than 1

Collisions for $H(x)$ is collision between f, g with probability $\frac{1}{R}$
$\implies$ repeat the search R times

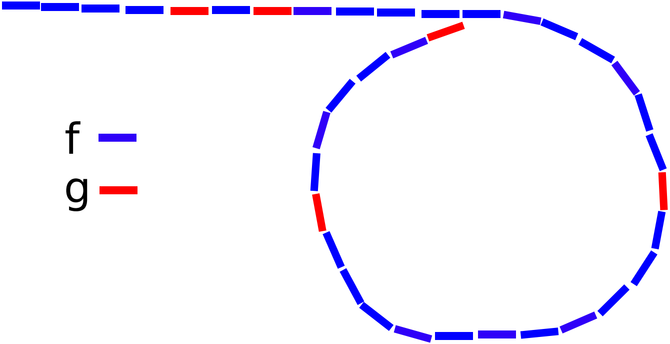
Unbalanced interleaving - Floyd's cycle finding algorithm



Unbalanced interleaving - Floyd's cycle finding algorithm



Unbalanced interleaving - Floyd's cycle finding algorithm



van Oorschot-Wiener Parallel Collision Search

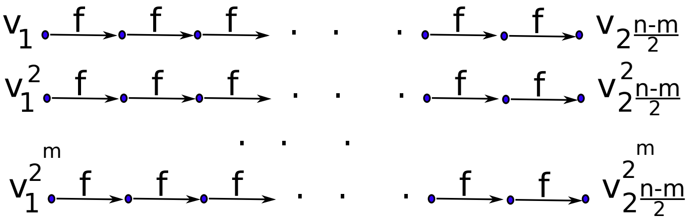
van Oorschot-Wiener algorithm can be used to find multiple collisions faster than Floyd's algorithm:

- Useful when many collisions are required
- It requires memory

van Oorschot-Wiener Algorithm: Hash Table

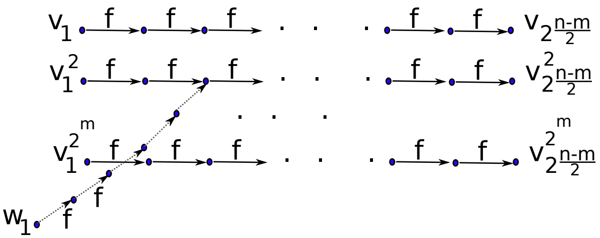
First, construct a hash table:

- Take a random point v_1 and produce a chain of values $v_i = f(v_{i-1}), i = 2, \ldots, 2^{\frac{n-m}{2}}$
- Store $(v_{2^{\frac{n-m}{2}}}, v_1)$ in hash table L
- Repeat for 2^m different points



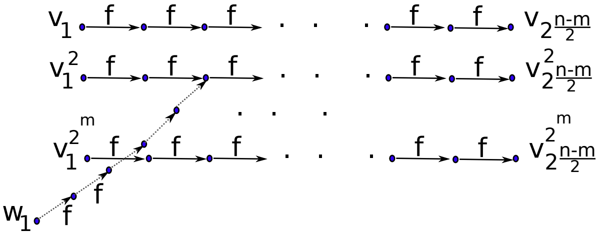
van Oorschot-Wiener Algorithm: Collision Search

1. Pick a random value w_1
2. Produce $w_i = f(w_{i-1})$
3. Check if w_i is in L. If not go to 2
4. By backtracking find the colliding values



van Oorschot-Wiener Algorithm: Collision Search

- During construction of L passed $2^{\frac{n+m}{2}}$ values
- If chain of w_i's is of length around $2^n/2^{\frac{n+m}{2}} = 2^{\frac{n-m}{2}}$ a collision will occur
- Time complexity of one collision: $2^{\frac{n-m}{2}}$



van Oorschot-Wiener Algorithm: Summary

- Initial cost for L: time $2^{\frac{n+m}{2}}$, memory 2^m
- Subsequent s collisions cost: $s \cdot 2^{\frac{n-m}{2}}$

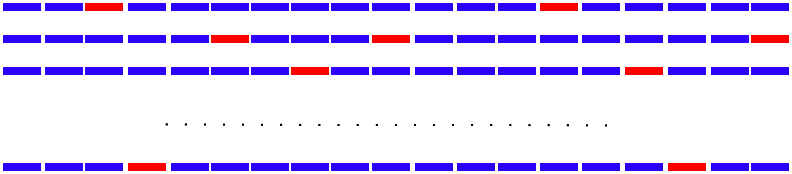
New Algorithm for Unbalanced Collision Search

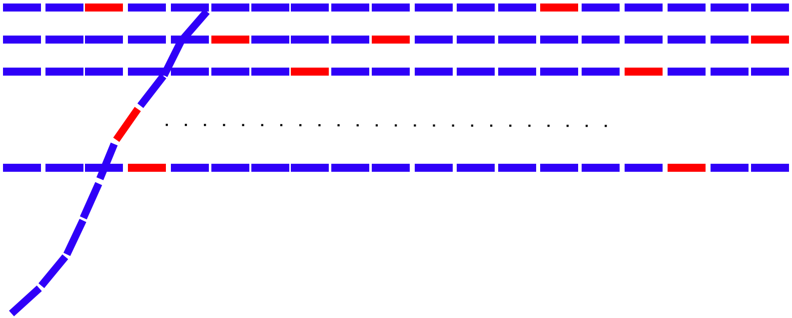
1. Define $H(x)$ as

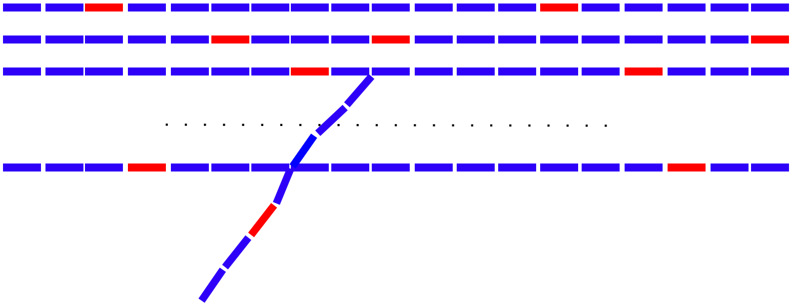
$$H(x) = \begin{cases} f(x) & \text{if } \sigma(x) = 0 \\ g(x) & \text{if } \sigma(x) = 1 \end{cases}$$

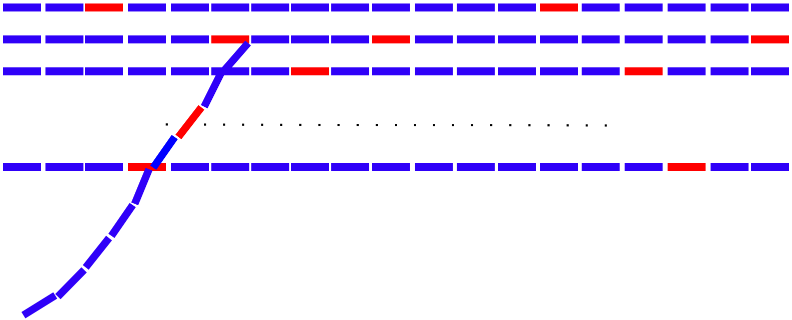
   $\sigma(x)$ outputs 0 around R times more often than 1
2. Construct hash table L for $H(x)$ with $M = 2^m$ entries
3. Find collision for $H(x)$. If not a collision for f, g repeat step 3

After repeating 3. around R times, collision for f, g will appear









New Algorithm: Complexity

$M = 2^m$
$T = 2^{\frac{n+m}{2}} + R \cdot 2^{\frac{n-m}{2}}$

When $2^{\frac{n+m}{2}} \leq R \cdot 2^{\frac{n-m}{2}}$, i.e. when $M \leq R$, then $T \approx R \cdot 2^{\frac{n-m}{2}}$, thus

$$T^2 M = R^2 \cdot 2^{(n-m)+m} = R^2 \cdot 2^n$$

Tradeoff

$$T^2 M = R^2 N,$$

where $M \leq R$.

New Algorithm: Misc

The new $T^2M = R^2N$ against the standard $TM = N$

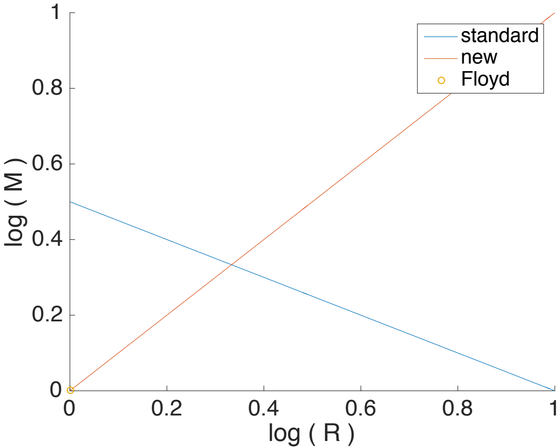
- **Better time** when $M < \frac{N}{R^2}$ (and $M \leq R$)
- **Better memory** when $T > R^2$ (and $T > \sqrt{RN}$)

About memory

Unbalanced collision search can be solved in optimal time with not more than $M = R$ memory.

New Algorithm: The Missing Link

Smaller the ratio R, less memory is required by the new algorithm.



Cons

The new algorithm may not always work as expected

- If R depends on the memory, then $T^2M = (R(M))^2N$
- If set(s) instead of function(s)
- If known plaintext (basically reduces to the above case)

Also, **if the user does not care about the memory complexity of his/her attack, then the new algorithm can be ignored.**

Applications

The new algorithm may replace the standard MITM algorithm in attacks resulting in the same time complexity but lower memory complexity

Certain balanced collision search problems can be reduced to unbalanced:

- Reduce the # calls of one of the functions to reduce data
- One of the functions has a reduced domain size

Conclusions and Open Problems

Conclusions:

- Consider using the new algorithm when dealing with unbalanced MITM problems
- Rule of thumb: if $R \leq 2^{\frac{n}{3}}$ then most likely the memory complexity of your attack can be reduced with the new algorithm (without increasing the time)

Problems:

- Find tricky use cases
- Find new algorithm when one side is given as a set