Quantum Collision-Finding in Non-Uniform Random Functions

Marko Balogh[1] and **Edward Eaton**[2,3] and Fang Song[1]

[1] Portland State University

[2] University of Waterloo

[3] ISARA Corporation

April 11, 2018

Motivation

Let

$$H : [M] \to [N]$$

be a hash function.

Motivation

Let

$$H : [M] \to [N]$$

be a hash function.

The *collision resistance* of H is a measure of how difficult it is to find $x, y \in [M]$ such that $H(x) = H(y)$.

Difficulty?

Difficulty?

- Time

Difficulty?

- Time
- Space

Difficulty?

- Time
- Space
- (Qu)bit operations

Difficulty?

- Time
- Space
- (Qu)bit operations (logical/physical)

Difficulty?

- Time
- Space
- (Qu)bit operations (logical/physical)
- Easy to parallelize

Difficulty?

- Time
- Space
- (Qu)bit operations (logical/physical)
- Easy to parallelize
- Hash function queries

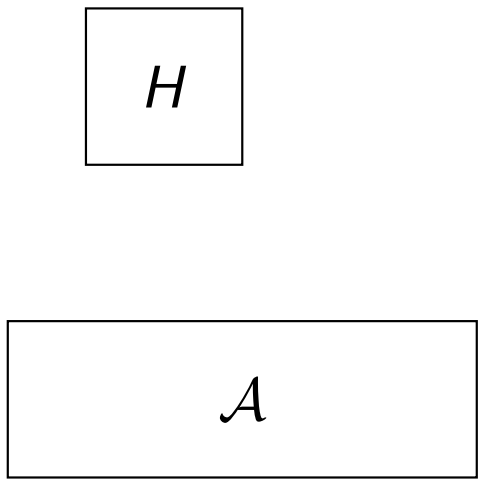
Difficulty?

- Time
- Space
- (Qu)bit operations (logical/physical)
- Easy to parallelize
- Hash function queries

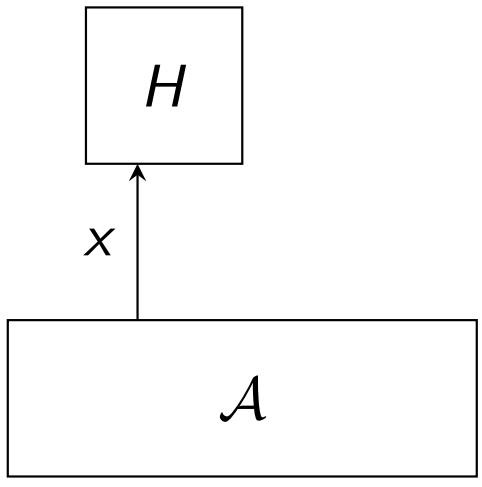
We can allow a *quantum* query to H by

$$U_H : \sum_{\substack{x\in[M]\\ y\in[N]}} \alpha_{x,y}|x\rangle|y\rangle \mapsto \sum_{\substack{x\in[M]\\ y\in[N]}} \alpha_{x,y}|x\rangle|y \oplus H(x)\rangle$$

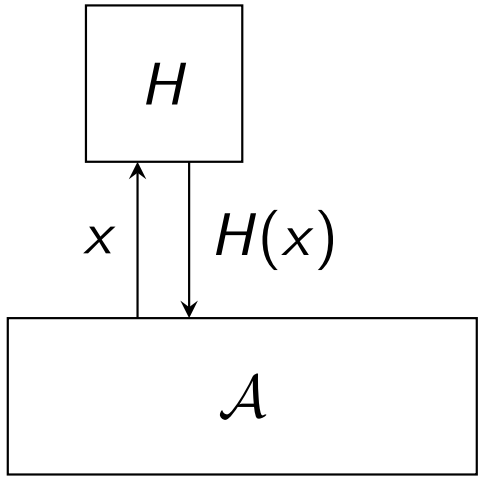
Generic Security



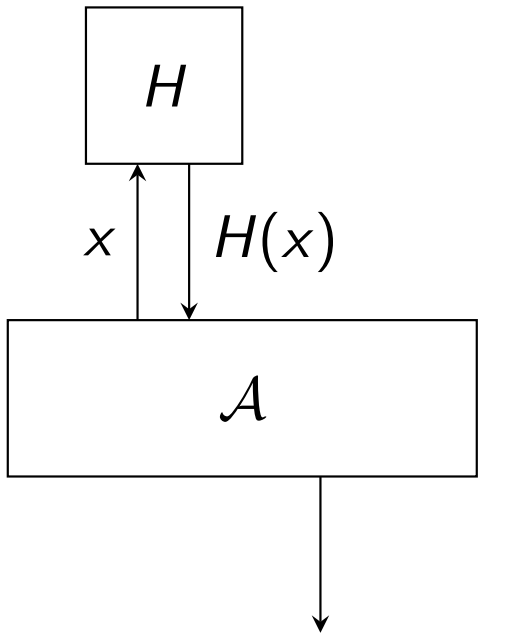
Generic Security



Generic Security



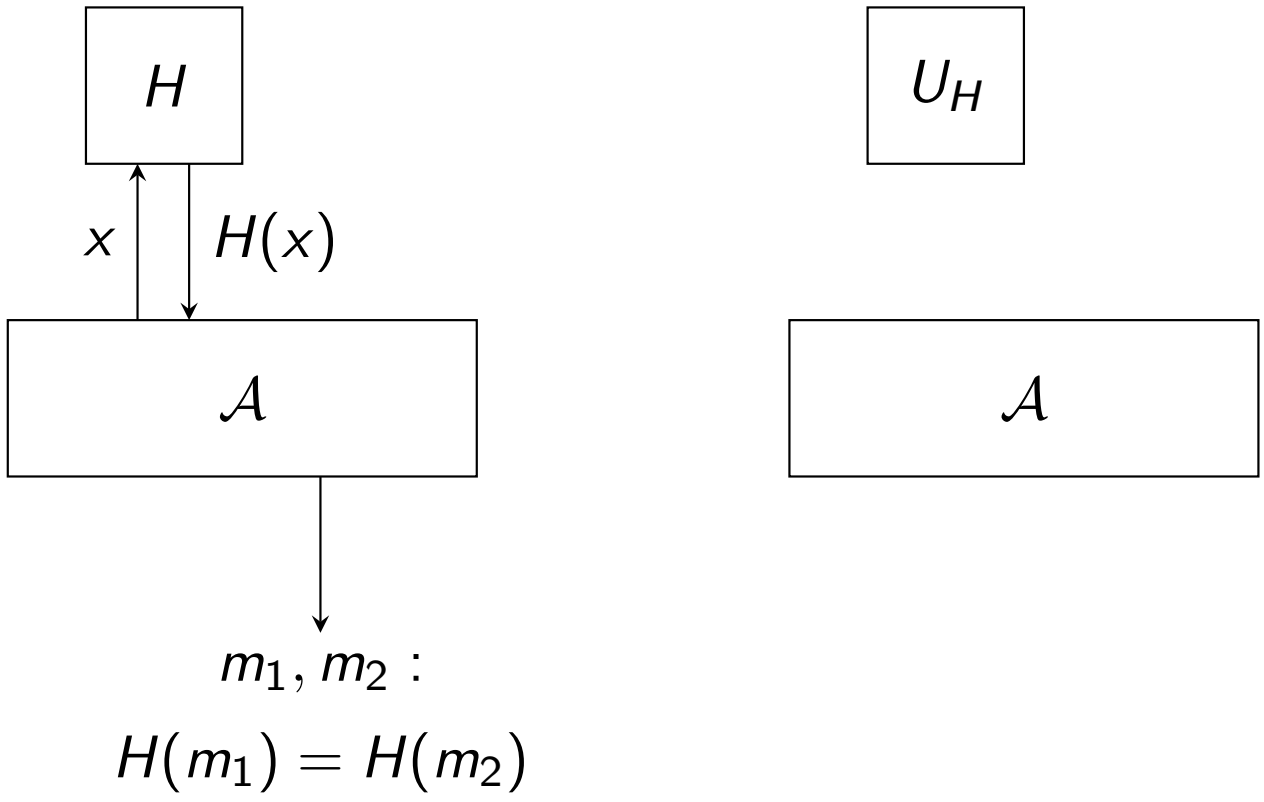
Generic Security



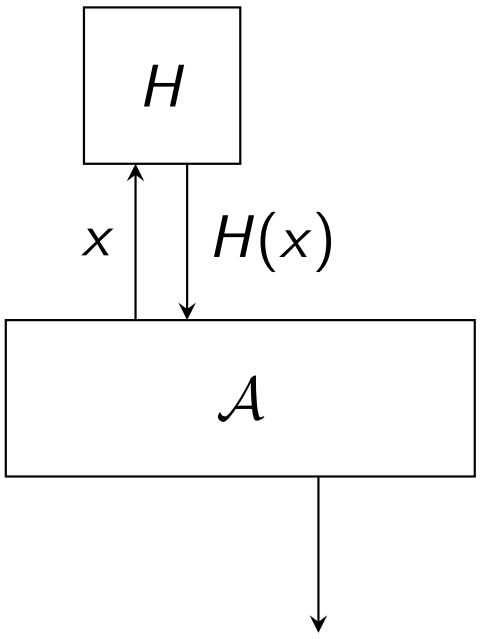
m_1, m_2 :

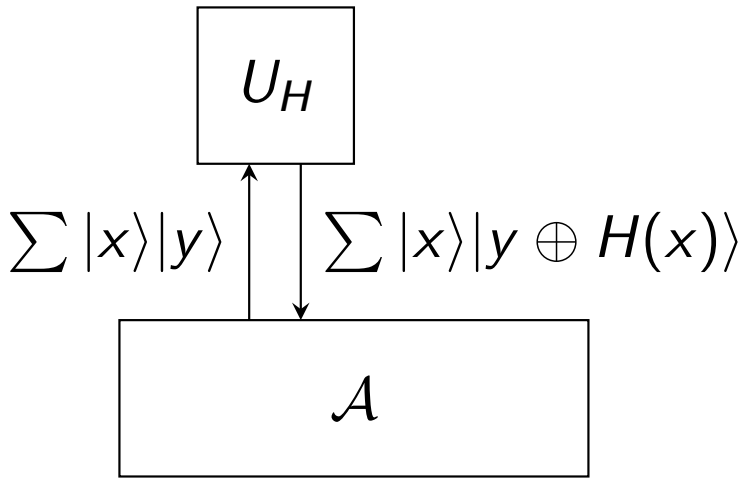
$H(m_1) = H(m_2)$

Generic Security

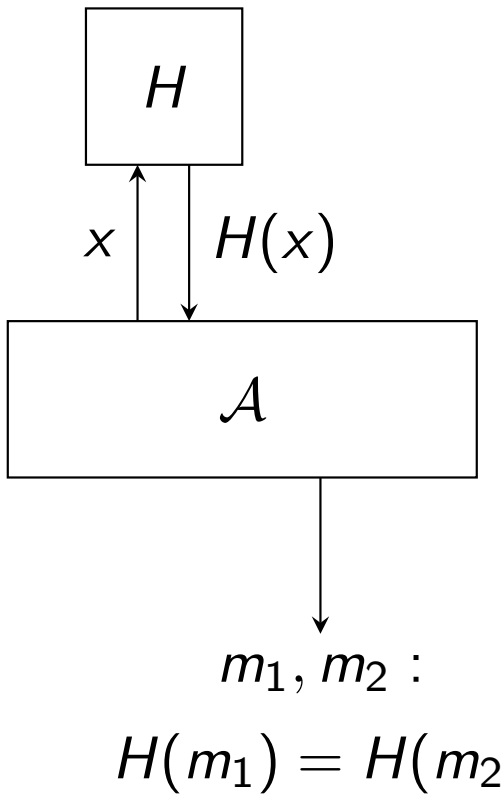


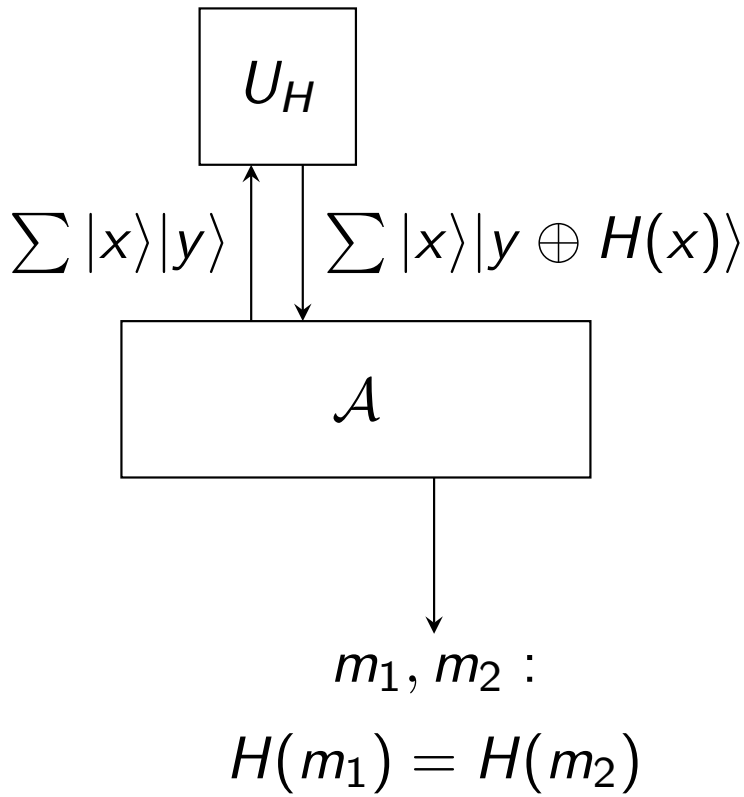
Generic Security





Generic Security





Collision Resistance

Let $\mathcal{H} := \{H : [M] \to [N]\}$, and $M = \Omega(N^{1/2})$.

Collision Resistance

Let $\mathcal{H} := \{H : [M] \to [N]\}$, and $M = \Omega(N^{1/2})$.

Then if we have $H \xleftarrow{\$} \mathcal{H}$ *uniformly*:

- Any algorithm finding a collision in H (with constant probability) must make $\Omega(N^{1/3})$ queries to U_H.

Collision Resistance

Let $\mathcal{H} := \{H : [M] \to [N]\}$, and $M = \Omega(N^{1/2})$.

Then if we have $H \xleftarrow{\$} \mathcal{H}$ *uniformly*:

- Any algorithm finding a collision in H (with constant probability) must make $\Omega(N^{1/3})$ queries to U_H.
- There is an algorithm that finds a collision in H (with constant probability) and makes $O(N^{1/3})$ queries to U_H.

Results from "A Note on the Quantum Collision and Set Equality Problems" by Mark Zhandry (2015).

Motivation

When H is uniform, the query complexity is $\Theta(N^{1/3})$.

Motivation

When H is uniform, the query complexity is $\Theta(N^{1/3})$.

Is only considering *uniform* functions *enough?*

Motivation

- Uniformity is a very strong condition on a function — considering non-uniform can relax our security assumptions.

Motivation

- Uniformity is a very strong condition on a function — considering non-uniform can relax our security assumptions.
- Some crypto functions are certainly *not* uniform, e.g., if H_1, H_2 are uniform then $H_1 \circ H_2$ is not.

Motivation

- Uniformity is a very strong condition on a function — considering non-uniform can relax our security assumptions.
- Some crypto functions are certainly *not* uniform, e.g., if H_1, H_2 are uniform then $H_1 \circ H_2$ is not.
- Proofs of Fujisaki-Okamoto require collision resistance of non-uniform functions ($f \circ H$).

Definitions

Let D be a distribution on $[N]$. Then we say that D has min-entropy k if

$$-\log_2 \max_{y \in [N]} \Pr[y \leftarrow D] = k.$$

Definitions

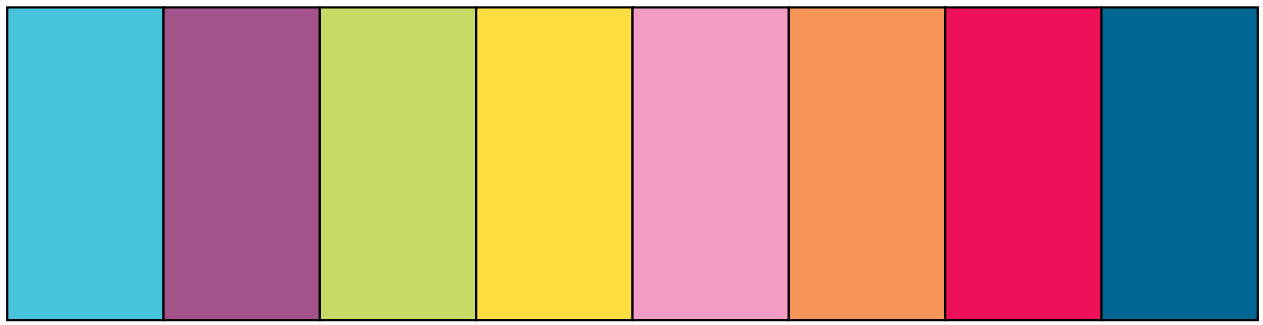
Let D be a distribution on $[N]$. Then we say that D has min-entropy k if

$$-\log_2 \max_{y \in [N]} \Pr[y \leftarrow D] = k.$$

We say that a function H has distribution D if $H(x)$ has distribution D for all $x \in [M]$, and all are independent.

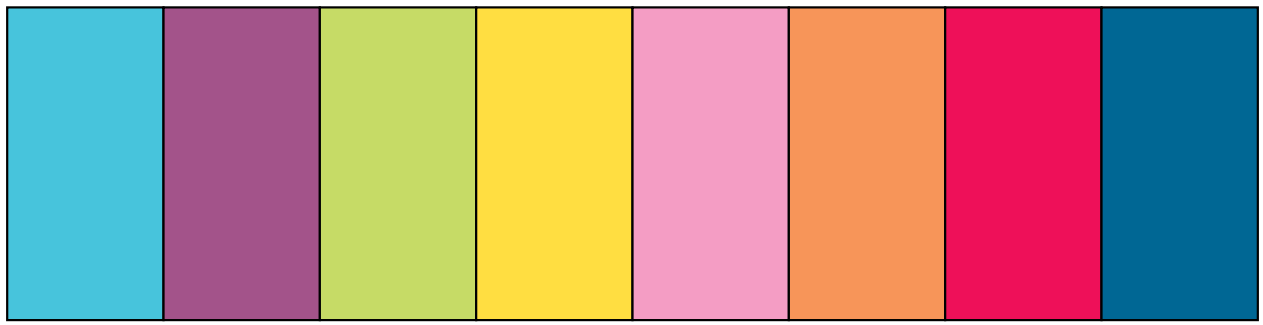
Examples

$N = 8$, $D_1 =$ uniform / flat:

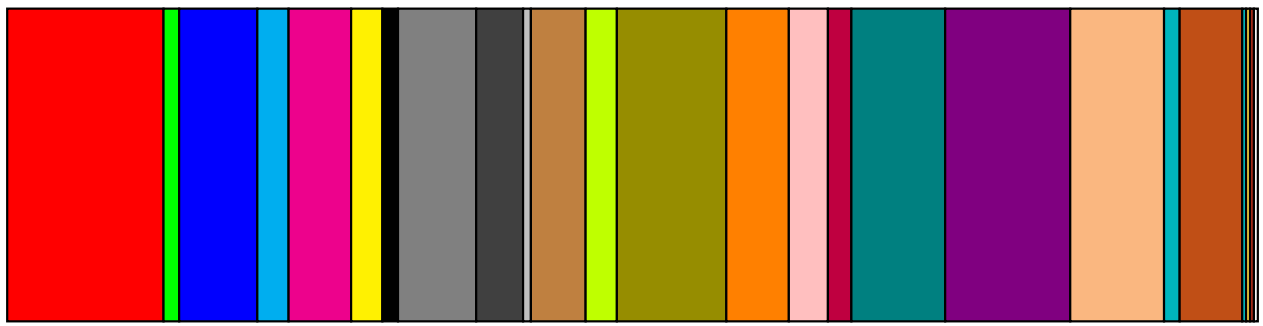


Examples

$N = 8$, $D_1 =$ uniform / flat:

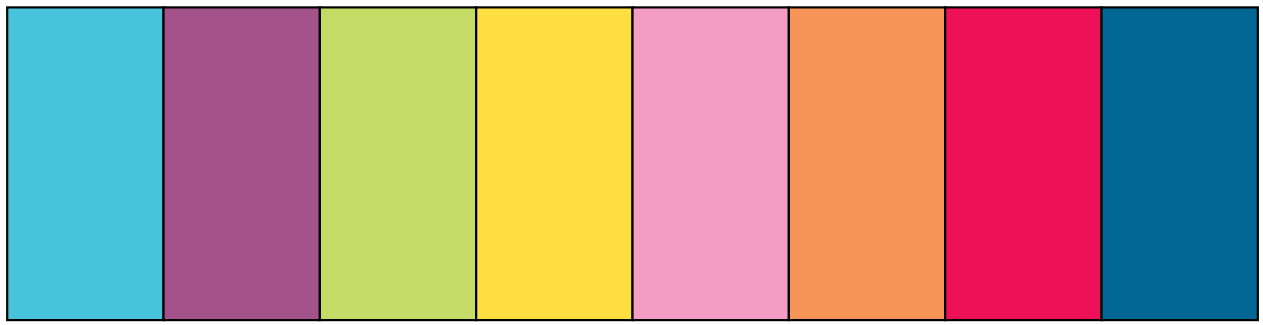


$N = 25$, $D_2 =$ *generic* :

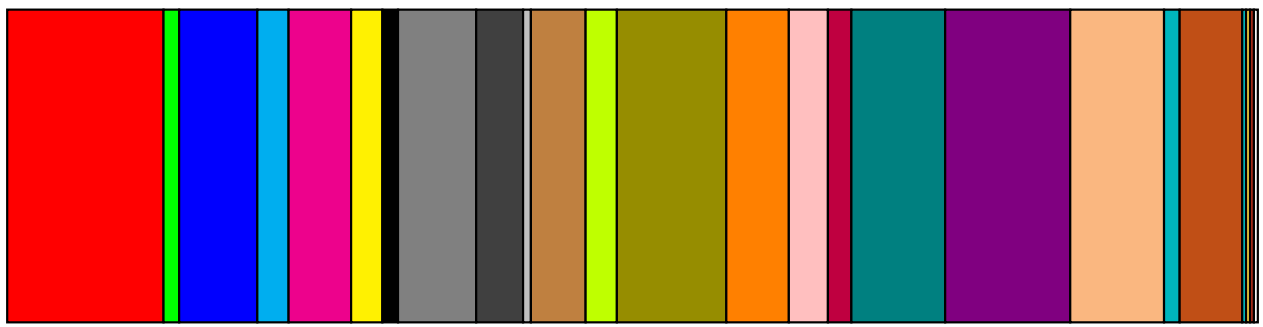


Examples

$N = 8$, $D_1 =$ uniform / flat:



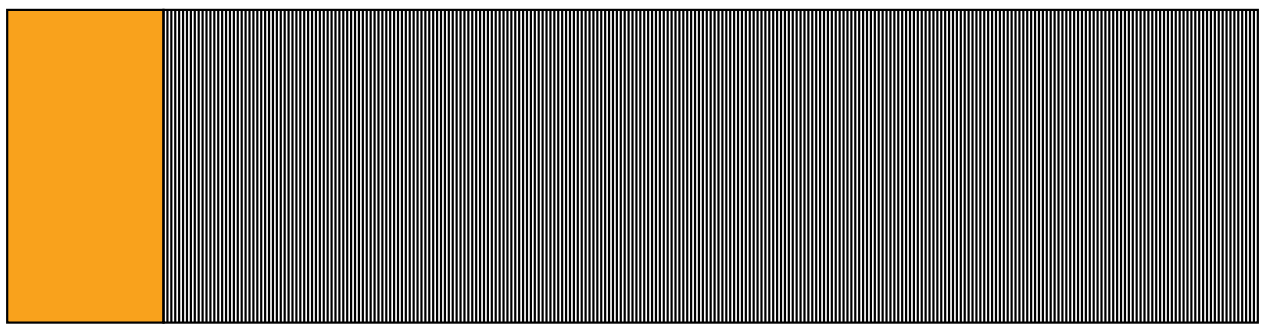
$N = 25$, $D_2 =$ *generic* :



Both have min-entropy 3

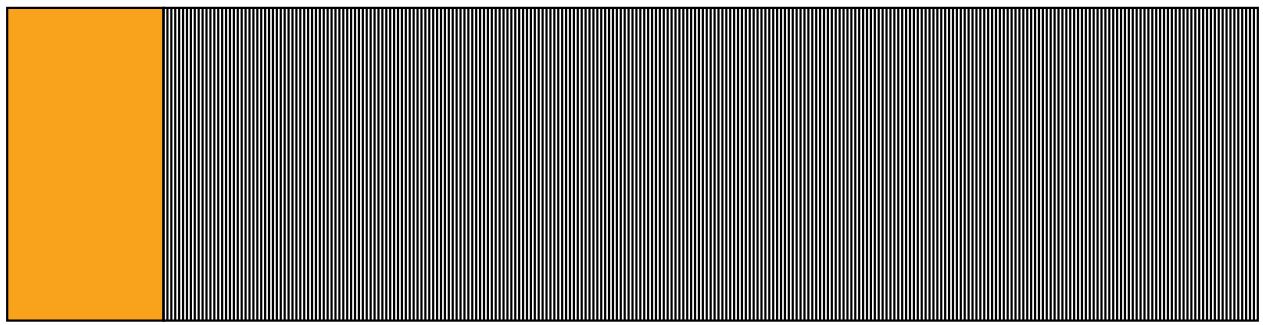
Examples

$N = \Omega(M)$, $D_3 =$ delta:



Examples

$N = \Omega(M)$, $D_3 =$ delta:



Still min-entropy 3

Definitions

Useful tool: The *collision probability*

$$\beta(D) := \frac{1}{\Pr[x = y : x, y \leftarrow D]}.$$

($-\log \beta$ is the *collision entropy*)

Definitions

Useful tool: The *collision probability*

$$\beta(D) := \frac{1}{\Pr[x = y : x, y \leftarrow D]}.$$

($-\log \beta$ is the *collision entropy*)

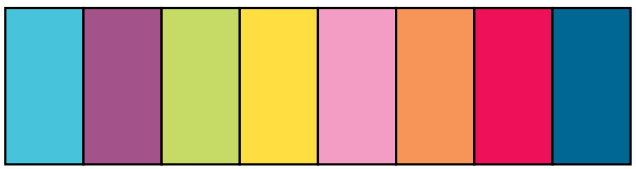
$\beta\Big($  $\Big) = 2^k$

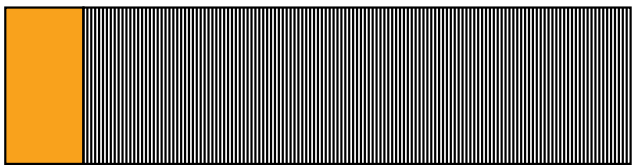
Definitions

Useful tool: The *collision probability*

$$\beta(D) := \frac{1}{\Pr[x = y : x, y \leftarrow D]}.$$

($-\log \beta$ is the *collision entropy*)

$\beta\Big($  $\Big) = 2^k$

$\beta\Big($  $\Big) \approx 2^{2k}$

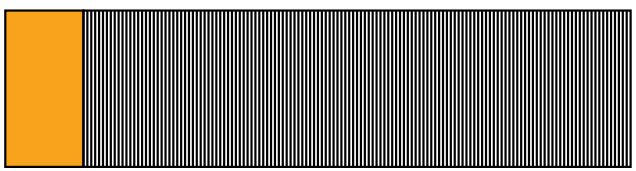
Definitions

Useful tool: The *collision probability*

$$\beta(D) := \frac{1}{\Pr[x = y : x, y \leftarrow D]}.$$

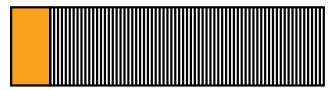
($-\log \beta$ is the *collision entropy*)

$\beta\Big($  $\Big) = 2^k$

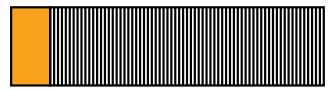
$\beta\Big($  $\Big) \approx 2^{2k}$

$\beta\Big($  $\Big) \in [2^k, 2^{2k})$

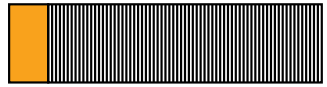
Previous Work

To find a collision with constant probability...			
it takes at **least** this many queries	$2^{k/3}$	?	$2^{k/9}$
it **can** be done in this many queries	$2^{k/3}$	?	?

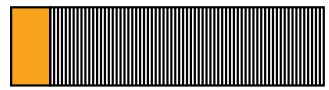
Independent Work — Ebrahimi & Unruh

To find a collision with constant probability...			
it takes at **least** this many queries	$2^{k/3}$	$2^{k/2}$	$2^{k/5}$
it **can** be done in this many queries	$2^{k/3}$	$2^{k/2}$	$\beta^{1/3}$

Our Work

To find a collision with constant probability...			
it takes at **least** this many queries	$2^{k/3}$	$\min\{N^{1/3}, 2^{k/2}\}$	$2^{k/3}$
it **can** be done in this many queries	$2^{k/3}$	$\min\{N^{1/3}, 2^{k/2}\}$	$\beta^{1/3}$

Our Work

To find a collision with constant probability...			
it takes at **least** this many queries	$2^{k/3}$	$\min\{N^{1/3}, 2^{k/2}\}$	$2^{k/3}$
it **can** be done in this many queries	$2^{k/3}$	$\min\{N^{1/3}, 2^{k/2}\}$	$\beta^{1/3}$

We prove

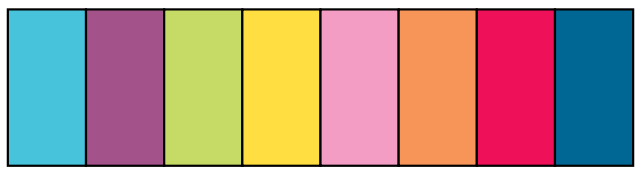
Any adversary that can find a collision in a hash function H' with outputs distributed by

in q queries to $U_{H'}$, with probability p,

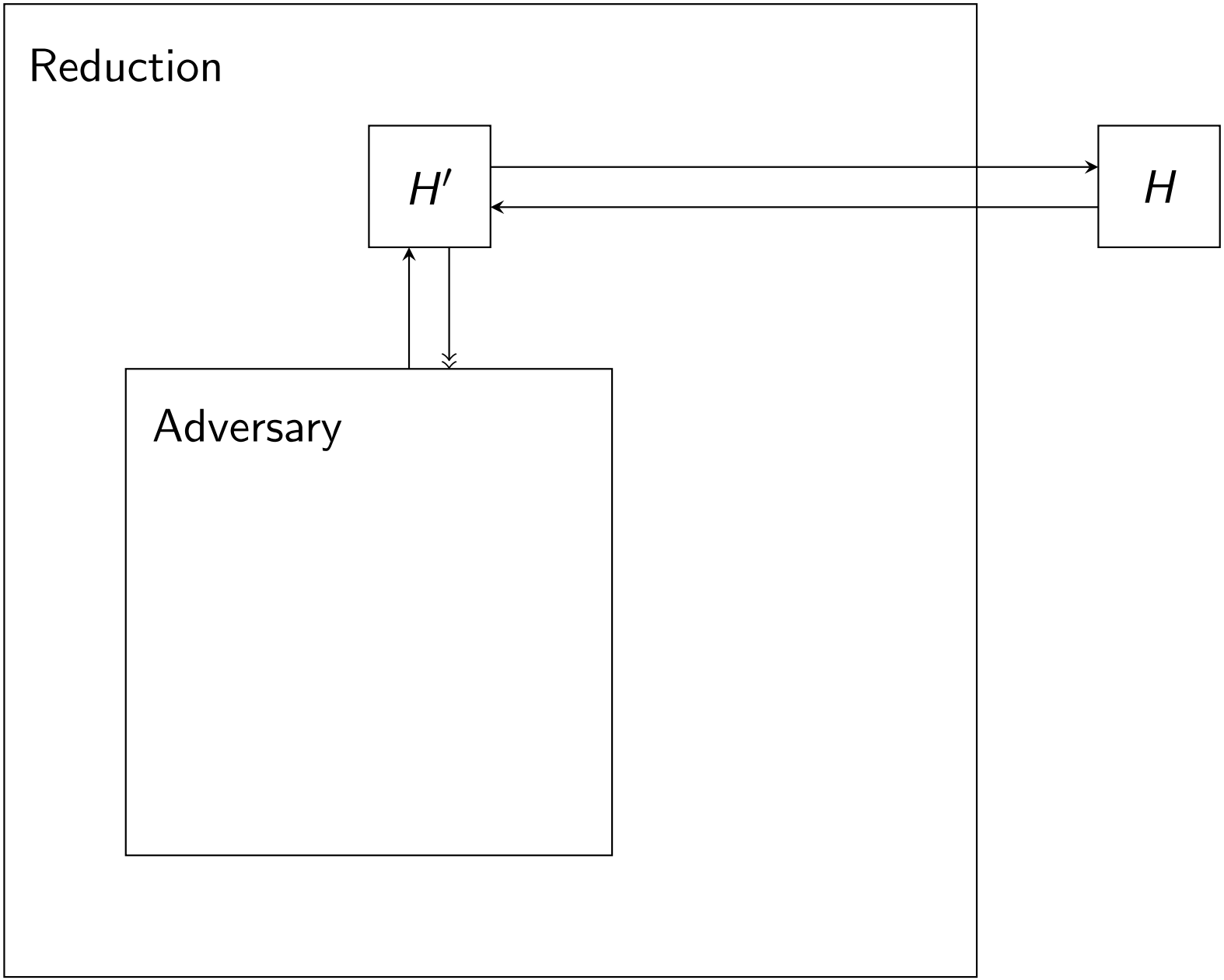
We prove

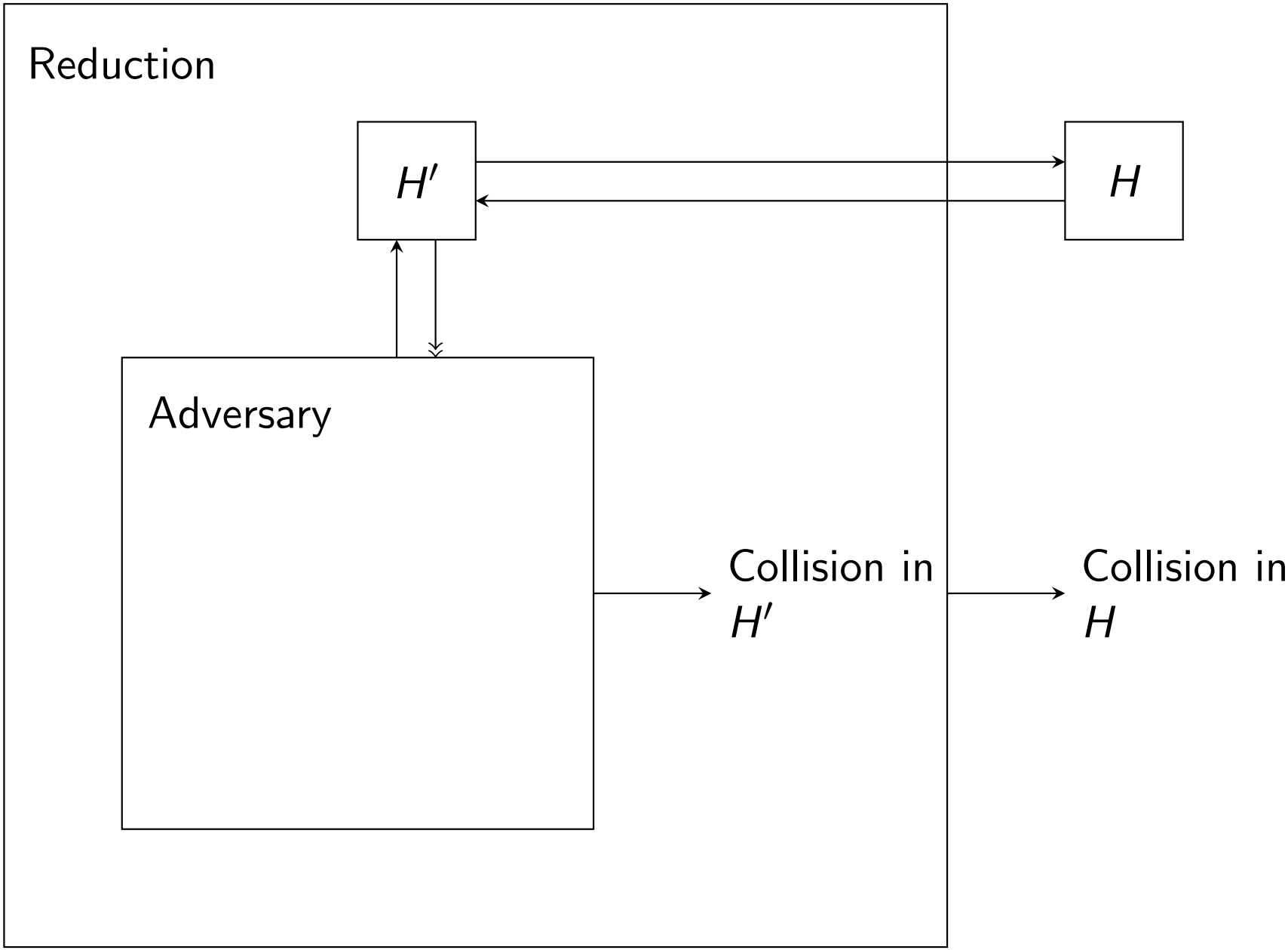
Any adversary that can find a collision in a hash function H' with outputs distributed by

in q queries to $U_{H'}$, with probability p, can be used to find a collision in a hash function H with outputs distributed by



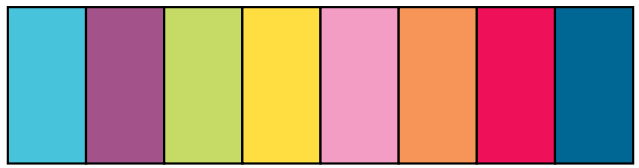
in $2q$ queries to U_H, with probability $p/2$.


Reduction
H'
H
Adversary


Reduction
H'
H
Adversary
Collision in H'
Collision in H

Proof outline

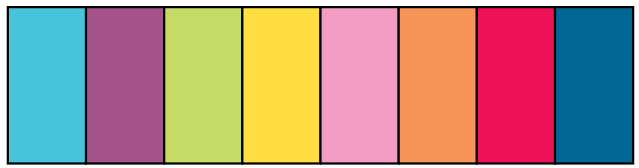
Idea: Use a distribution conversion to ‘chop up’



and turn it into

Proof outline

Idea: Use a distribution conversion to ‘chop up’

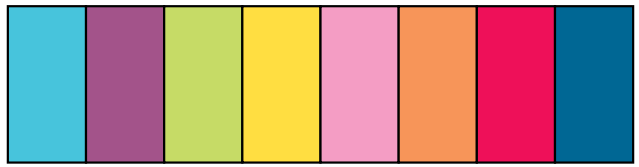


and turn it into

Then show that a collision in the generic distribution should imply a collision in the uniform!

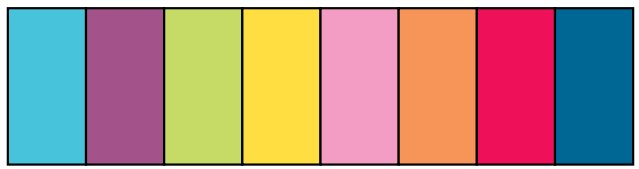
Simulating H'

Say we have H with output distribution

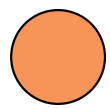


Simulating H'

Say we have H with output distribution

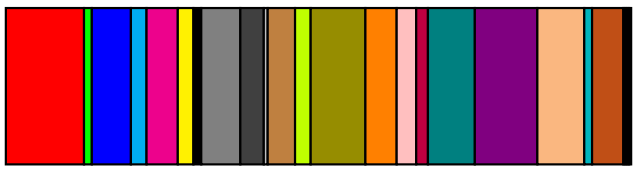


We pick $m \xleftarrow{\$} [M]$ and compute

$$H(m) = \quad$$ 

Simulating H'

We want to provide the adversary with access to a hash function H' with distribution

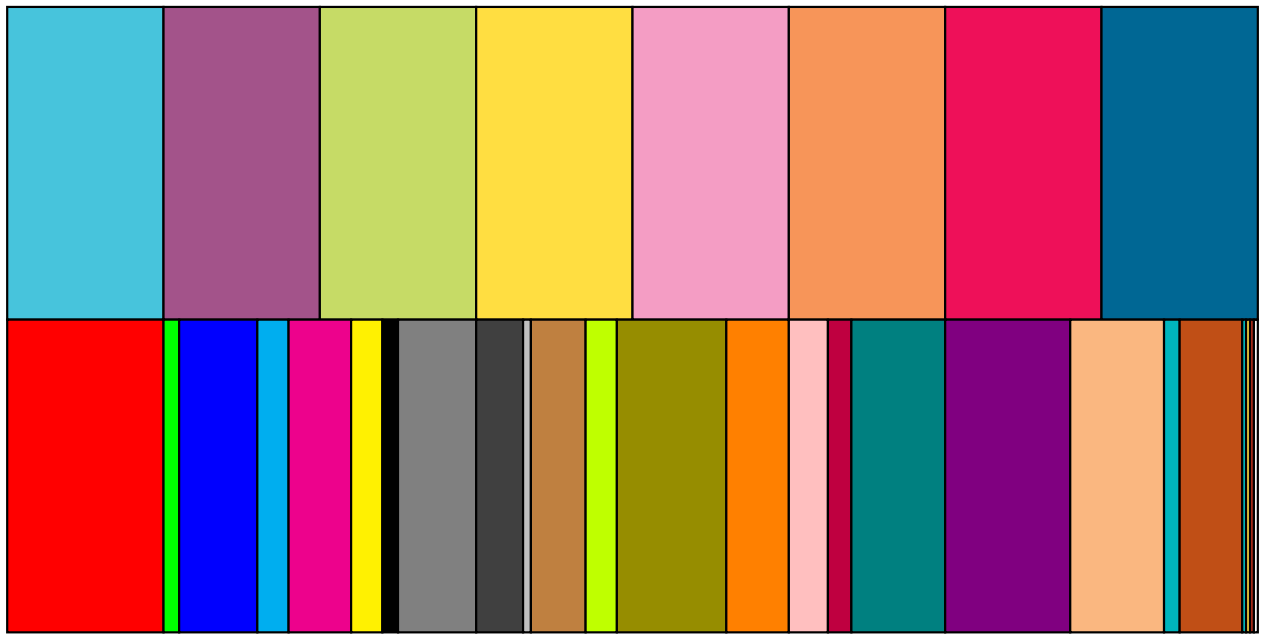


Simulating H'

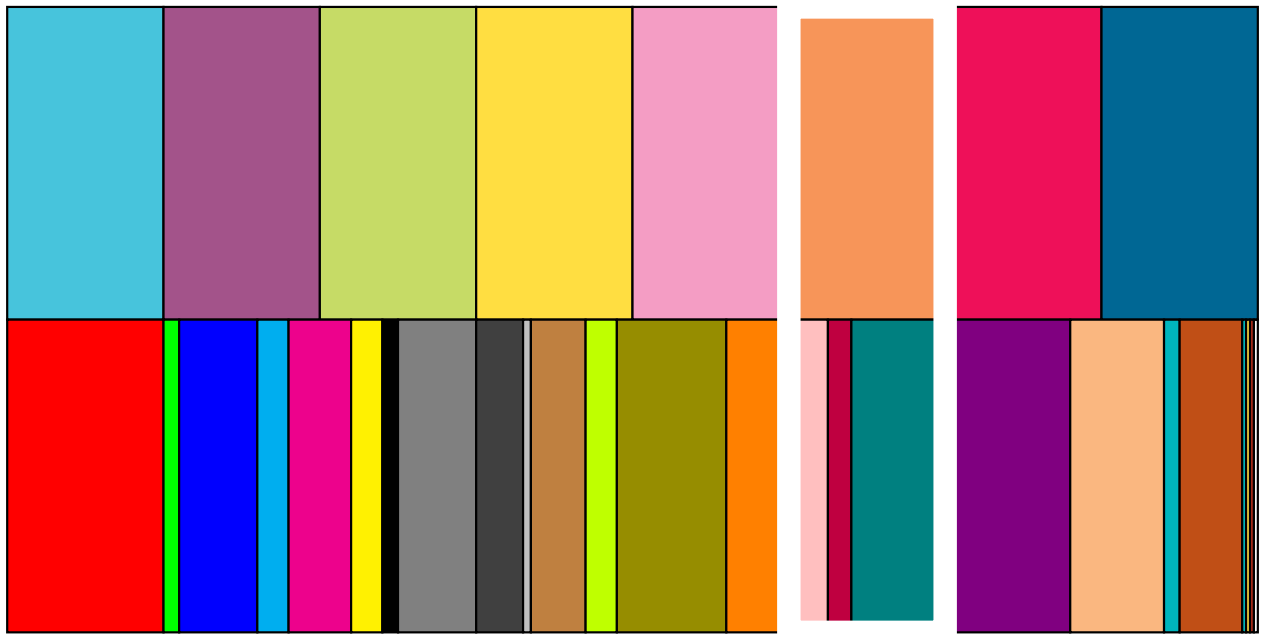
We want to provide the adversary with access to a hash function H' with distribution

So when we compute $H(m) =$  , we will choose what $H'(m)$ can be based on this.

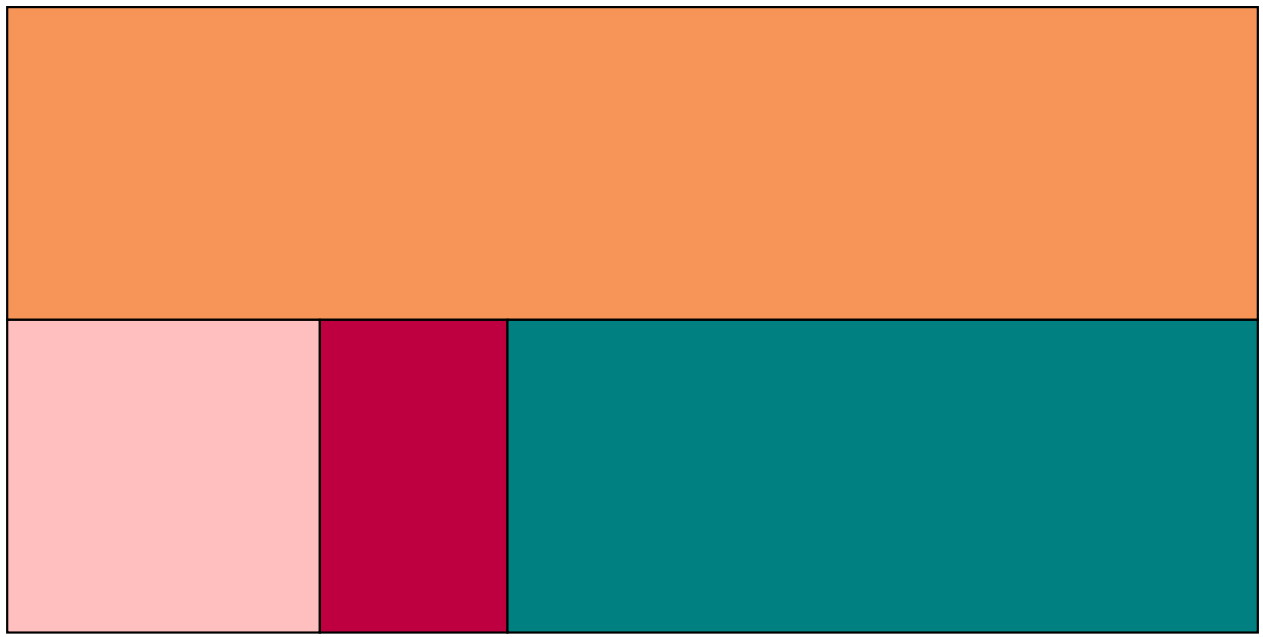
Simulating H'



Simulating H'

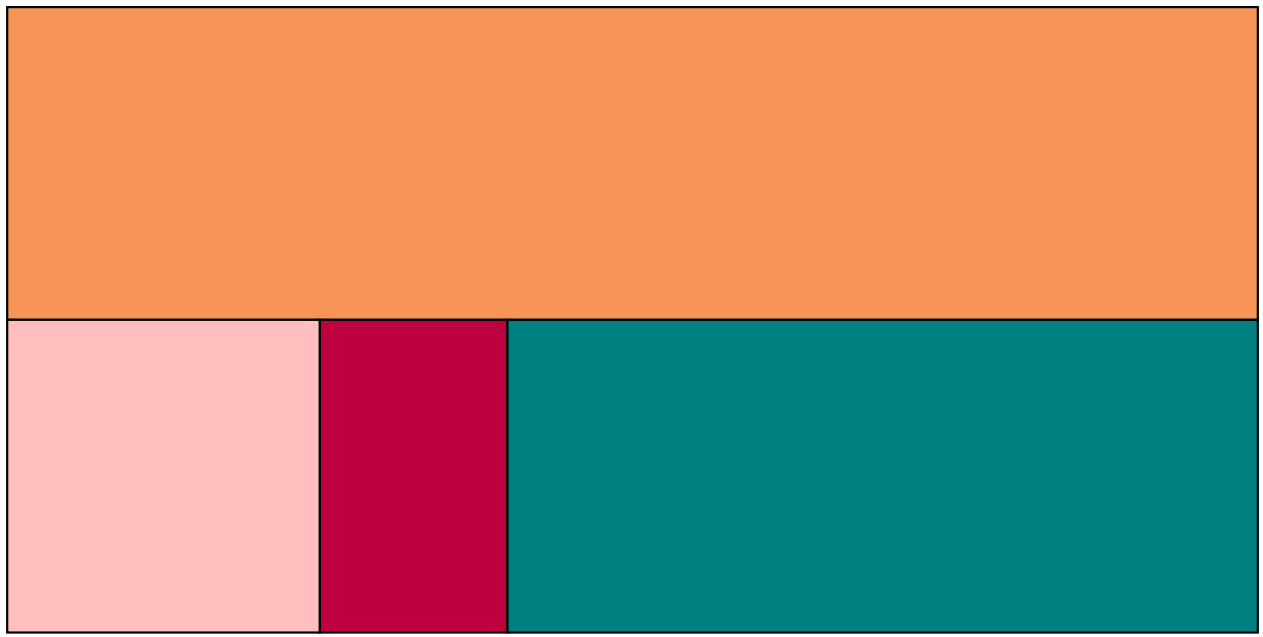


Simulating H'



When $H(m) =$  we will set $H'(m)$ to either  ,  , or  .

Simulating H'



When $H(m) =$  we will set $H'(m)$ to either  ,  , or  . We need a randomness oracle $R : [M] \to \{0,1\}^*$ to decide which. (Querying R does not require us to query H.)

Simulating H'

When the adversary makes a query on a point m, we:

Simulating H'

When the adversary makes a query on a point m, we:

- Compute $H(m)$.

Simulating H'

When the adversary makes a query on a point m, we:

- Compute $H(m)$.
- Obtain 'sufficient' randomness by $R(m)$.

Simulating H'

When the adversary makes a query on a point m, we:

- Compute $H(m)$.
- Obtain 'sufficient' randomness by $R(m)$.
- From this, decide what $H'(m)$ is by breaking up $H(m)$.

Simulating H'

When the adversary makes a query on a point m, we:

- Compute $H(m)$.
- Obtain 'sufficient' randomness by $R(m)$.
- From this, decide what $H'(m)$ is by breaking up $H(m)$.

Note that this only requires one query to H.

Converting H' collision to H

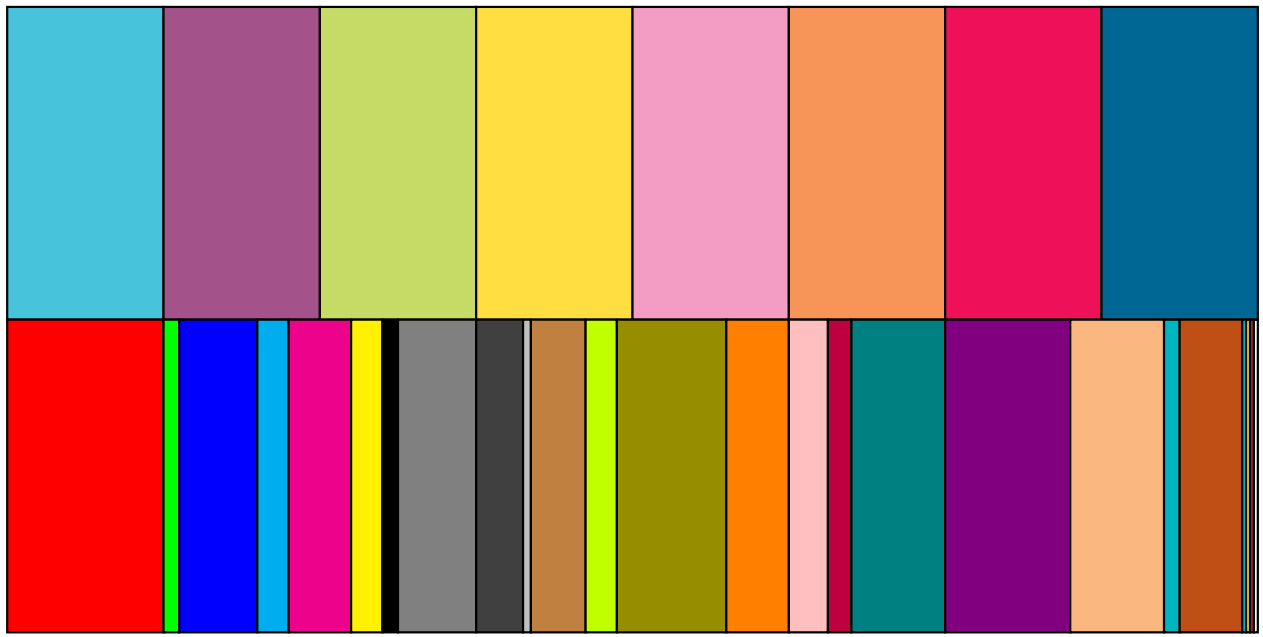
Then note that if the adversary finds $m_1, m_2 \in [M]$ such that $H'(m_1) = H'(m_2) =$  , we have that $H(m_1) = H(m_2) =$  .

Converting H' collision to H

Then note that if the adversary finds $m_1, m_2 \in [M]$ such that $H'(m_1) = H'(m_2) =$  , we have that $H(m_1) = H(m_2) =$  .

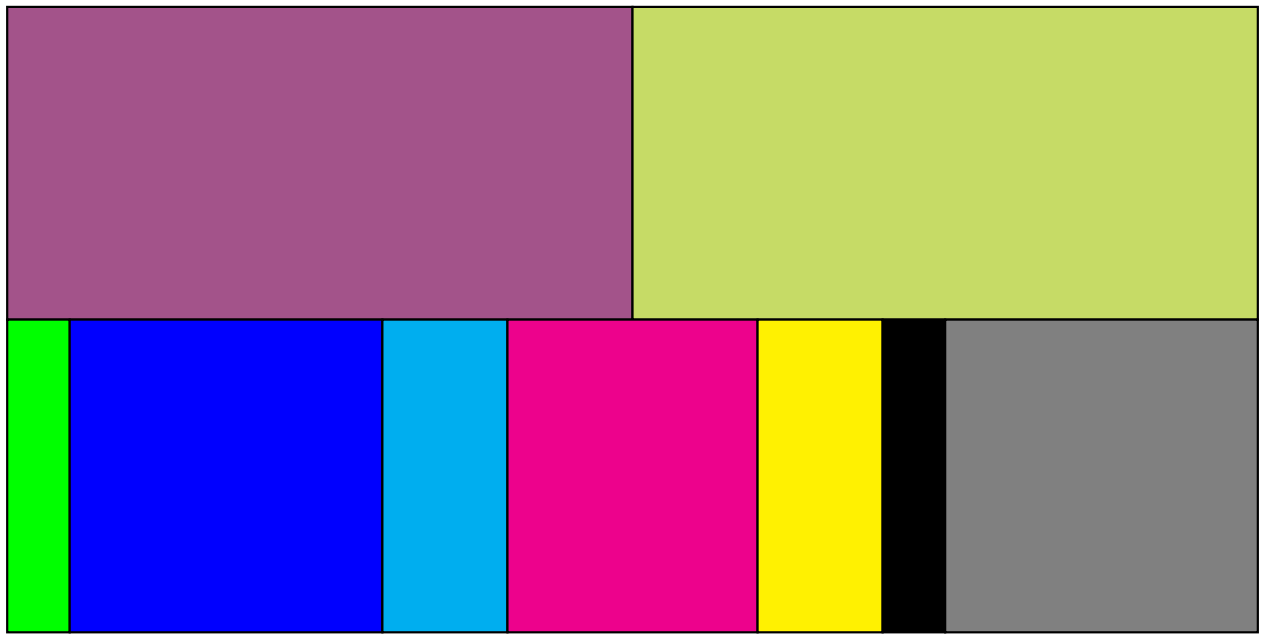
But we don't always have this property.

Converting H' collision to H



Converting H' collision to H

Converting H' collision to H



Say the adversary finds $m_1, m_2 \in [M]$ such that $H'(m_1) = H'(m_2) =$  . What does this mean for $H(m_1)$ and $H(m_2)$?

Converting H' collision to H

If $H'(m_1) = H'(m_2) =$  , we could have:

Converting H' collision to H

If $H'(m_1) = H'(m_2) =$  , we could have:

- $H(m_1) =$  , $H(m_2) =$  .
- $H(m_1) =$  , $H(m_2) =$  .
- $H(m_1) =$  , $H(m_2) =$  .
- $H(m_1) =$  , $H(m_2) =$  .

Converting H' collision to H

If $H'(m_1) = H'(m_2) =$  , we could have:

- $H(m_1) =$  , $H(m_2) =$  .
- $H(m_1) =$  , $H(m_2) =$  .
- $H(m_1) =$  , $H(m_2) =$  .
- $H(m_1) =$  , $H(m_2) =$  .

However, note that we do not provide the adversary with direct access to H, only to H'.

Converting H' collision to H

If $H'(m_1) = H'(m_2) =$  , we could have:

- $H(m_1) =$  , $H(m_2) =$  .
- $H(m_1) =$  , $H(m_2) =$  .
- $H(m_1) =$  , $H(m_2) =$  .
- $H(m_1) =$  , $H(m_2) =$  .

However, note that we do not provide the adversary with direct access to H, only to H'.

The adversary has no way to tell which

Converting H' collision to H

If $H'(m_1) = H'(m_2) =$  , we could have:

- $H(m_1) =$  , $H(m_2) =$  .
- $H(m_1) =$  , $H(m_2) =$  .
- $H(m_1) =$  , $H(m_2) =$  .
- $H(m_1) =$  , $H(m_2) =$  .

However, note that we do not provide the adversary with direct access to H, only to H'.

The adversary has no way to tell which — carefully accounting, we can see that

$$\Pr[H(m_1) = H(m_2)] \geq 1/2.$$

(Probability over the R oracle and the adversary's internal randomness).

Simulating $U_{H'}$

For a *quantum*-enabled adversary, we must quantumize this process, e.g., on input of a query $|\Psi\rangle$, we:

Simulating $U_{H'}$

For a *quantum*-enabled adversary, we must quantumize this process, e.g., on input of a query $|\Psi\rangle$, we:

- Make a quantum query to U_H to get $U_H|\Psi\rangle$,

Simulating $U_{H'}$

For a *quantum*-enabled adversary, we must quantumize this process, e.g., on input of a query $|\Psi\rangle$, we:

- Make a quantum query to U_H to get $U_H|\Psi\rangle$,
- Pass the *message register* part of $|\Psi\rangle$ through U_R to obtain randomness to break apart the uniform distribution,

Simulating $U_{H'}$

For a *quantum*-enabled adversary, we must quantumize this process, e.g., on input of a query $|\Psi\rangle$, we:

- Make a quantum query to U_H to get $U_H|\Psi\rangle$,
- Pass the *message register* part of $|\Psi\rangle$ through U_R to obtain randomness to break apart the uniform distribution,
- Pass this result into a quantum circuit that breaks apart the uniform distribution to get the desired min-entropy k distribution.

Simulating $U_{H'}$

For a *quantum*-enabled adversary, we must quantumize this process, e.g., on input of a query $|\Psi\rangle$, we:

- Make a quantum query to U_H to get $U_H|\Psi\rangle$,
- Pass the *message register* part of $|\Psi\rangle$ through U_R to obtain randomness to break apart the uniform distribution,
- Pass this result into a quantum circuit that breaks apart the uniform distribution to get the desired min-entropy k distribution.
- Perform garbage collection and pass result to the adversary.

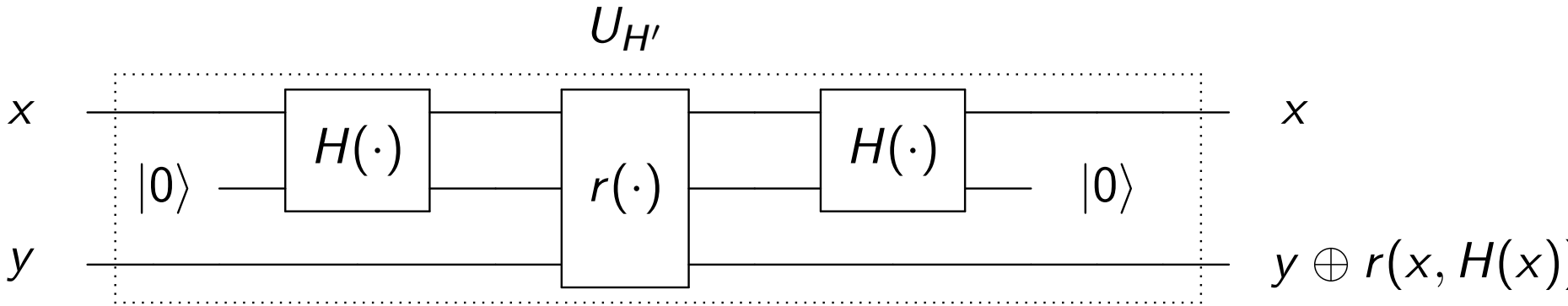


Figure: Quantum circuit that implements function $U_{H'}$ using two oracle calls to U_H.

- For each query the adversary makes to $U_{H'}$, we make two queries to U_H.

- For each query the adversary makes to $U_{H'}$, we make two queries to U_H.
- The function H' has *exactly* the correct distribution.

- For each query the adversary makes to $U_{H'}$, we make two queries to U_H.
- The function H' has *exactly* the correct distribution.
- The adversary finds a collision with probability p.

- For each query the adversary makes to $U_{H'}$, we make two queries to U_H.
- The function H' has *exactly* the correct distribution.
- The adversary finds a collision with probability p.
- This collision corresponds to a collision in H with probability $1/2$.

- For each query the adversary makes to $U_{H'}$, we make two queries to U_H.
- The function H' has *exactly* the correct distribution.
- The adversary finds a collision with probability p.
- This collision corresponds to a collision in H with probability $1/2$.

Since we know that $2^{k/3}$ queries are needed to break the collision resistance of H with constant probability, the adversary needs to make at least $2^{k/3}$ queries to H'.

Open Questions

- Can we get distribution conversions that are *time efficient* (not just query efficient)?
- Can we get bounds when distributions aren't entirely known?
- Lower bounds in terms of $\beta(D)$ rather than k?

Summary

- Main result: Finding a collision in a min-entropy k distributed hash function takes $2^{k/3}$ quantum queries.

Summary

- Main result: Finding a collision in a min-entropy k distributed hash function takes $2^{k/3}$ quantum queries.
- There exist min-entropy k distributions that take $2^{k/2}$ queries to find a collision.

Summary

- Main result: Finding a collision in a min-entropy k distributed hash function takes $2^{k/3}$ quantum queries.
- There exist min-entropy k distributions that take $2^{k/2}$ queries to find a collision.
- There exists an algorithm that finds a collision in a min-entropy k distribution in $\beta^{1/3}$ queries.

Summary

- Main result: Finding a collision in a min-entropy k distributed hash function takes $2^{k/3}$ quantum queries.
- There exist min-entropy k distributions that take $2^{k/2}$ queries to find a collision.
- There exists an algorithm that finds a collision in a min-entropy k distribution in $\beta^{1/3}$ queries.
- Preimage/second preimage difficulty on min-entropy k distributions can also be characterized with β and analysed with distribution conversions.

Summary

- Main result: Finding a collision in a min-entropy k distributed hash function takes $2^{k/3}$ quantum queries.
- There exist min-entropy k distributions that take $2^{k/2}$ queries to find a collision.
- There exists an algorithm that finds a collision in a min-entropy k distribution in $\beta^{1/3}$ queries.
- Preimage/second preimage difficulty on min-entropy k distributions can also be characterized with β and analysed with distribution conversions.

Thank you!