Hash Functions

A hash function is a family of functions that:

- are length reducing (the output is smaller than the input),
- and for which it is hard to find collisions.

Hash Functions

A hash function is a family of functions that:

- are length reducing (the output is smaller than the input),
- and for which it is hard to find collisions.

(Gen, h)
$\mathrm{Gen}(1^n)$: Output key s.

Hash Functions

A hash function is a family of functions that:

- are length reducing (the output is smaller than the input),
- and for which it is hard to find collisions.

(Gen, h)
$\mathrm{Gen}(1^n)$: Output key s.
$h^s(x)$: on input $x \in \{0,1\}^*$, output $y \in \{0,1\}^{\ell(n)}$

Hash Functions

A hash function is a family of functions that:

- are length reducing (the output is smaller than the input),
- and for which it is hard to find collisions.

(Gen, h)
Gen(1^n): Output key s.
$h^s(x)$: on input $x \in \{0,1\}^*$, output $y \in \{0,1\}^{\ell(n)}$
For fixed-length hash functions, $x \in \{0,1\}^{\ell'(n)}$, and $\ell'(n) > \ell(n)$.

Hash Functions

A hash function is a family of functions that:

- are length reducing (the output is smaller than the input),
- and for which it is hard to find collisions.

(Gen, h)
Gen(1^n): Output key s.
$h^s(x)$: on input $x \in \{0,1\}^*$, output $y \in \{0,1\}^{\ell(n)}$
For fixed-length hash functions, $x \in \{0,1\}^{\ell'(n)}$, and $\ell'(n) > \ell(n)$.

Security:

Hash Functions

A hash function is a family of functions that:

- are length reducing (the output is smaller than the input),
- and for which it is hard to find collisions.

(Gen, h)
Gen(1^n): Output key s.
$h^s(x)$: on input $x \in \{0,1\}^*$, output $y \in \{0,1\}^{\ell(n)}$
For fixed-length hash functions, $x \in \{0,1\}^{\ell'(n)}$, and $\ell'(n) > \ell(n)$.

Security:

Challenger | Adv

$s \leftarrow \mathrm{Gen}(1^n)$

$\xrightarrow{\quad s \quad}$

$\xleftarrow{\quad x,y \quad}$

$H^s(x) = H^s(y)$

$x \neq y$

Unkeyed Hash Functions in practice

In practice, we most commonly use SHA-256 or its recent replacement, SHA-3.

Unkeyed Hash Functions in practice

In practice, we most commonly use SHA-256 or its recent replacement, SHA-3. These are *unkeyed* hash functions.

Unkeyed Hash Functions in practice

In practice, we most commonly use SHA-256 or its recent replacement, SHA-3.
These are *unkeyed* hash functions.
They've through a rigorous trial-by-fire, and we *believe* that nobody can find a single collision.

Unkeyed Hash Functions in practice

In practice, we most commonly use SHA-256 or its recent replacement, SHA-3.
These are *unkeyed* hash functions.
They've through a rigorous trial-by-fire, and we *believe* that nobody can find a single collision.
However, *what does that mean, formally?!*

Unkeyed Hash Functions in practice

In practice, we most commonly use SHA-256 or its recent replacement, SHA-3.
These are *unkeyed* hash functions.
They've through a rigorous trial-by-fire, and we *believe* that nobody can find a single collision.
However, *what does that mean, formally?!*

Consider the following adversary:

$\underline{\mathcal{A}_{1,2}}$:

  Output 1, 2.

Unkeyed Hash Functions in practice

In practice, we most commonly use SHA-256 or its recent replacement, SHA-3.
These are *unkeyed* hash functions.
They've through a rigorous trial-by-fire, and we *believe* that nobody can find a single collision.
However, *what does that mean, formally?!*

Consider the following adversary:
$\underline{\mathcal{A}_{1,2}}$:
    Output 1, 2.
Most likely, $\mathcal{A}_{1,2}$ does not find a collision.

Unkeyed Hash Functions in practice

In practice, we most commonly use SHA-256 or its recent replacement, SHA-3.
These are *unkeyed* hash functions.
They've through a rigorous trial-by-fire, and we *believe* that nobody can find a single collision.
However, *what does that mean, formally?!*

Consider the following adversary:
$\underline{\mathcal{A}_{1,2}}$:
Output 1, 2.
Most likely, $\mathcal{A}_{1,2}$ does not find a collision.

Consider the following adversary:
$\underline{\mathcal{A}_{1,3}}$:
Output 1, 3.

Unkeyed Hash Functions in practice

In practice, we most commonly use SHA-256 or its recent replacement, SHA-3.
These are *unkeyed* hash functions.
They've through a rigorous trial-by-fire, and we *believe* that nobody can find a single collision.
However, *what does that mean, formally?!*

Consider the following adversary:

$\underline{\mathcal{A}_{1,2}}$:

  Output 1, 2.

Most likely, $\mathcal{A}_{1,2}$ does not find a collision.

Consider the following adversary:

$\underline{\mathcal{A}_{1,3}}$:

  Output 1, 3.

Most likely, $\mathcal{A}_{1,3}$ does not find a collision.

Unkeyed Hash Functions in practice

In practice, we most commonly use SHA-256 or its recent replacement, SHA-3.
These are *unkeyed* hash functions.
They've through a rigorous trial-by-fire, and we *believe* that nobody can find a single collision.
However, *what does that mean, formally?!*

Consider the following adversary:

$\underline{\mathcal{A}_{1,2}}$:
    Output 1, 2.

Most likely, $\mathcal{A}_{1,2}$ does not find a collision.

Consider the following adversary:

$\underline{\mathcal{A}_{1,3}}$:
    Output 1, 3.

Most likely, $\mathcal{A}_{1,3}$ does not find a collision.

Consider the following adversary:

$\underline{\mathcal{A}_{i,j}}$:
    Output i, j.

Unkeyed Hash Functions in practice

In practice, we most commonly use SHA-256 or its recent replacement, SHA-3.
These are *unkeyed* hash functions.
They've through a rigorous trial-by-fire, and we *believe* that nobody can find a single collision.
However, *what does that mean, formally?!*

Consider the following adversary:

$\underline{\mathcal{A}_{1,2}}$:
  Output 1, 2.

Most likely, $\mathcal{A}_{1,2}$ does not find a collision.

Consider the following adversary:

$\underline{\mathcal{A}_{1,3}}$:
  Output 1, 3.

Most likely, $\mathcal{A}_{1,3}$ does not find a collision.

Consider the following adversary:

$\underline{\mathcal{A}_{i,j}}$:
  Output i, j.

For *some* i, j, $\mathcal{A}_{i,j}$ is a polynomial time adversary that outputs a collision!!

Unkeyed Hash Functions in practice

In practice, we most commonly use SHA-256 or its recent replacement, SHA-3.
These are *unkeyed* hash functions.
They've through a rigorous trial-by-fire, and we *believe* that nobody can find a single collision.
However, *what does that mean, formally?!*

Consider the following adversary:

$\underline{\mathcal{A}_{1,2}}$:
  Output 1, 2.

Most likely, $\mathcal{A}_{1,2}$ does not find a collision.

Consider the following adversary:

$\underline{\mathcal{A}_{1,3}}$:
  Output 1, 3.

Most likely, $\mathcal{A}_{1,3}$ does not find a collision.

Consider the following adversary:

$\underline{\mathcal{A}_{i,j}}$:
  Output i, j.

For *some* i, j, $\mathcal{A}_{i,j}$ is a polynomial time adversary that outputs a collision!!

However, if $\mathcal{A}_{i,j}$ were given a randomly chose key s, it is very unlikely that i, j constitute a collision for the specific hash function h^s.

Unkeyed Hash Functions in practice

In practice, we most commonly use SHA-256 or its recent replacement, SHA-3.
These are *unkeyed* hash functions.
They've through a rigorous trial-by-fire, and we *believe* that nobody can find a single collision.
However, *what does that mean, formally?!*

Consider the following adversary:
$\underline{\mathcal{A}_{1,2}}$:
Output 1, 2.
Most likely, $\mathcal{A}_{1,2}$ does not find a collision.

Consider the following adversary:
$\underline{\mathcal{A}_{1,3}}$:
Output 1, 3.
Most likely, $\mathcal{A}_{1,3}$ does not find a collision.

Consider the following adversary:
$\underline{\mathcal{A}_{i,j}}$:
Output i, j.
For *some* i, j, $\mathcal{A}_{i,j}$ is a polynomial time adversary that outputs a collision!!

However, if $\mathcal{A}_{i,j}$ were given a randomly chose key s, it is very unlikely that i, j constitute a collision for the specific hash function h^s.
Put another way: we believe there is no adversary that can win for many keys simultaneously. So choosing a random key is safe.

Unkeyed Hash Functions in practice

In practice, we most commonly use SHA-256 or its recent replacement, SHA-3.
These are *unkeyed* hash functions.
They've through a rigorous trial-by-fire, and we *believe* that nobody can find a single collision.
However, *what does that mean, formally?!*

Consider the following adversary:
$\underline{\mathcal{A}_{1,2}}$:
  Output 1, 2.
Most likely, $\mathcal{A}_{1,2}$ does not find a collision.

Consider the following adversary:
$\underline{\mathcal{A}_{1,3}}$:
  Output 1, 3.
Most likely, $\mathcal{A}_{1,3}$ does not find a collision.

Consider the following adversary:
$\underline{\mathcal{A}_{i,j}}$:
  Output i, j.
For *some* i, j, $\mathcal{A}_{i,j}$ is a polynomial time adversary that outputs a collision!!

However, if $\mathcal{A}_{i,j}$ were given a randomly chose key s, it is very unlikely that i, j constitute a collision for the specific hash function h^s.
Put another way: we believe there is no adversary that can win for many keys simultaneously. So choosing a random key is safe.
In practice, we don't know of any adversaries that can find any collision in SHA-256 or SHA-3, so we use these unkeyed hash functions anyway.

Do hash outputs look random?

We shouldn't confuse pseudorandom functions with hash functions!

Do hash outputs look random?

We shouldn't confuse pseudorandom functions with hash functions!
The security definition only says that it is hard to find collisions. It does not say that the output looks like a random string.

Do hash outputs look random?

We shouldn't confuse pseudorandom functions with hash functions!
The security definition only says that it is hard to find collisions. It does not say that the output looks like a random string.

Let (Gen, h) be a collision resistant hash function.

Do hash outputs look random?

We shouldn't confuse pseudorandom functions with hash functions!
The security definition only says that it is hard to find collisions. It does not say that the output looks like a random string.

Let (Gen, h) be a collision resistant hash function.

Define $(\mathsf{Gen}, \hat{h})$:
$\hat{h}^s(x) = 1||h^s(x)$

Do hash outputs look random?

We shouldn't confuse pseudorandom functions with hash functions!
The security definition only says that it is hard to find collisions. It does not say that the output looks like a random string.

Let (Gen, h) be a collision resistant hash function.

Define $(\mathsf{Gen}, \hat{h})$:
$\hat{h}^s(x) = 1 || h^s(x)$
Exercise: prove that if (Gen, h) is collision resistant, then $(\mathsf{Gen}, \hat{h})$ is collision resistant.

Do hash outputs look random?

We shouldn't confuse pseudorandom functions with hash functions!
The security definition only says that it is hard to find collisions. It does not say that the output looks like a random string.

Let (Gen, h) be a collision resistant hash function.

Define $(\mathrm{Gen}, \hat{h})$:
$\hat{h}^s(x) = 1||h^s(x)$
Exercise: prove that if (Gen, h) is collision resistant, then $(\mathrm{Gen}, \hat{h})$ is collision resistant.
Clearly the output of $\hat{h}$ is not random looking!

Weaker Security Notions

A weaker security notion, called second pre-image resistance, or target collision resistance, is implied by collision resistance.

Weaker Security Notions

A weaker security notion, called second pre-image resistance, or target collision resistance, is implied by collision resistance.

Second Preimage Resistance:
$\mathcal{A}$ is given random key s, and a random input x.
$\mathcal{A}$ has to output $\hat{x}$ such that $h^s(\hat{x}) = h^s(x)$.

Weaker Security Notions

A weaker security notion, called second pre-image resistance, or target collision resistance, is implied by collision resistance.

Second Preimage Resistance:
$\mathcal{A}$ is given random key s, and a random input x.
$\mathcal{A}$ has to output $\hat{x}$ such that $h^s(\hat{x}) = h^s(x)$.

Clearly if $\mathcal{A}$ can win at this game, there exists an adversary $\hat{\mathcal{A}}$ that can win at the collision resistant game.
$\hat{\mathcal{A}}$ can simply sample x itself, run $\mathcal{A}$ to get $\hat{x}$, and output collision $x, \hat{x}$.

Weaker Security Notions

A weaker security notion, called second pre-image resistance, or target collision resistance, is implied by collision resistance.

Second Preimage Resistance:
$\mathcal{A}$ is given random key s, and a random input x.
$\mathcal{A}$ has to output $\hat{x}$ such that $h^s(\hat{x}) = h^s(x)$.

Clearly if $\mathcal{A}$ can win at this game, there exists an adversary $\hat{\mathcal{A}}$ that can win at the collision resistant game.
$\hat{\mathcal{A}}$ can simply sample x itself, run $\mathcal{A}$ to get $\hat{x}$, and output collision $x, \hat{x}$.

Preimage Resistance:
$\mathcal{A}$ is given a random key s and a random output y.
$\mathcal{A}$ has to output some x such that $h^s(x) = y$.

Weaker Security Notions

A weaker security notion, called second pre-image resistance, or target collision resistance, is implied by collision resistance.

Second Preimage Resistance:
$\mathcal{A}$ is given random key s, and a random input x.
$\mathcal{A}$ has to output $\hat{x}$ such that $h^s(\hat{x}) = h^s(x)$.

Clearly if $\mathcal{A}$ can win at this game, there exists an adversary $\hat{\mathcal{A}}$ that can win at the collision resistant game.
$\hat{\mathcal{A}}$ can simply sample x itself, run $\mathcal{A}$ to get $\hat{x}$, and output collision $x, \hat{x}$.

Preimage Resistance:
$\mathcal{A}$ is given a random key s and a random output y.
$\mathcal{A}$ has to output some x such that $h^s(x) = y$.

Claim: If $\mathcal{A}$ can win at this game, then there exists an adversary $\hat{\mathcal{A}}$ that can win at the second preimage resistance game.

Weaker Security Notions

A weaker security notion, called second pre-image resistance, or target collision resistance, is implied by collision resistance.

Second Preimage Resistance:
$\mathcal{A}$ is given random key s, and a random input x.
$\mathcal{A}$ has to output $\hat{x}$ such that $h^s(\hat{x}) = h^s(x)$.

Clearly if $\mathcal{A}$ can win at this game, there exists an adversary $\hat{\mathcal{A}}$ that can win at the collision resistant game.
$\hat{\mathcal{A}}$ can simply sample x itself, run $\mathcal{A}$ to get $\hat{x}$, and output collision $x, \hat{x}$.

Preimage Resistance:
$\mathcal{A}$ is given a random key s and a random output y.
$\mathcal{A}$ has to output some x such that $h^s(x) = y$.

Claim: If $\mathcal{A}$ can win at this game, then there exists an adversary $\hat{\mathcal{A}}$ that can win at the second preimage resistance game.
$\hat{\mathcal{A}}$ is given s and x.

Weaker Security Notions

A weaker security notion, called second pre-image resistance, or target collision resistance, is implied by collision resistance.

Second Preimage Resistance:
$\mathcal{A}$ is given random key s, and a random input x.
$\mathcal{A}$ has to output $\hat{x}$ such that $h^s(\hat{x}) = h^s(x)$.

Clearly if $\mathcal{A}$ can win at this game, there exists an adversary $\hat{\mathcal{A}}$ that can win at the collision resistant game.
$\hat{\mathcal{A}}$ can simply sample x itself, run $\mathcal{A}$ to get $\hat{x}$, and output collision $x, \hat{x}$.

Preimage Resistance:
$\mathcal{A}$ is given a random key s and a random output y.
$\mathcal{A}$ has to output some x such that $h^s(x) = y$.

Claim: If $\mathcal{A}$ can win at this game, then there exists an adversary $\hat{\mathcal{A}}$ that can win at the second preimage resistance game.
$\hat{\mathcal{A}}$ is given s and x.
It computes $y = h^s(x)$, and gives s, y to $\mathcal{A}$.

Weaker Security Notions

A weaker security notion, called second pre-image resistance, or target collision resistance, is implied by collision resistance.

Second Preimage Resistance:
$\mathcal{A}$ is given random key s, and a random input x.
$\mathcal{A}$ has to output $\hat{x}$ such that $h^s(\hat{x}) = h^s(x)$.

Clearly if $\mathcal{A}$ can win at this game, there exists an adversary $\hat{\mathcal{A}}$ that can win at the collision resistant game.
$\hat{\mathcal{A}}$ can simply sample x itself, run $\mathcal{A}$ to get $\hat{x}$, and output collision $x, \hat{x}$.

Preimage Resistance:
$\mathcal{A}$ is given a random key s and a random output y.
$\mathcal{A}$ has to output some x such that $h^s(x) = y$.

Claim: If $\mathcal{A}$ can win at this game, then there exists an adversary $\hat{\mathcal{A}}$ that can win at the second preimage resistance game.
$\hat{\mathcal{A}}$ is given s and x.
It computes $y = h^s(x)$, and gives s, y to $\mathcal{A}$.
$\mathcal{A}$ returns $\hat{x}$ such that $h^s(\hat{x}) = y$. If $\hat{x} \neq x$, $\hat{\mathcal{A}}$ outputs $\hat{x}$ and wins.

Weaker Security Notions

A weaker security notion, called second pre-image resistance, or target collision resistance, is implied by collision resistance.

Second Preimage Resistance:
$\mathcal{A}$ is given random key s, and a random input x.
$\mathcal{A}$ has to output $\hat{x}$ such that $h^s(\hat{x}) = h^s(x)$.

Clearly if $\mathcal{A}$ can win at this game, there exists an adversary $\hat{\mathcal{A}}$ that can win at the collision resistant game.
$\hat{\mathcal{A}}$ can simply sample x itself, run $\mathcal{A}$ to get $\hat{x}$, and output collision $x, \hat{x}$.

Preimage Resistance:
$\mathcal{A}$ is given a random key s and a random output y.
$\mathcal{A}$ has to output some x such that $h^s(x) = y$.

Claim: If $\mathcal{A}$ can win at this game, then there exists an adversary $\hat{\mathcal{A}}$ that can win at the second preimage resistance game.
$\hat{\mathcal{A}}$ is given s and x.
It computes $y = h^s(x)$, and gives s, y to $\mathcal{A}$.
$\mathcal{A}$ returns $\hat{x}$ such that $h^s(\hat{x}) = y$. If $\hat{x} \neq x$, $\hat{\mathcal{A}}$ outputs $\hat{x}$ and wins.

How likely is it that $\hat{x} = x$?
Note that $\mathcal{A}$ is only given y and does not know how it was computed:
x was sampled at random, $y = h^s(x)$.
Just as easily, it could have been $\hat{x}$ sampled at random, and $y = h^s(\hat{x})$!

Weaker Security Notions

A weaker security notion, called second pre-image resistance, or target collision resistance, is implied by collision resistance.

Second Preimage Resistance:

$\mathcal{A}$ is given random key s, and a random input x.

$\mathcal{A}$ has to output $\hat{x}$ such that $h^s(\hat{x}) = h^s(x)$.

Clearly if $\mathcal{A}$ can win at this game, there exists an adversary $\hat{\mathcal{A}}$ that can win at the collision resistant game.

$\hat{\mathcal{A}}$ can simply sample x itself, run $\mathcal{A}$ to get $\hat{x}$, and output collision $x, \hat{x}$.

Preimage Resistance:

$\mathcal{A}$ is given a random key s and a random output y.

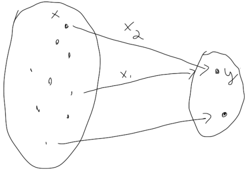
$\mathcal{A}$ has to output some x such that $h^s(x) = y$.

Claim: If $\mathcal{A}$ can win at this game, then there exists an adversary $\hat{\mathcal{A}}$ that can win at the second preimage resistance game.

$\hat{\mathcal{A}}$ is given s and x.

It computes $y = h^s(x)$, and gives s, y to $\mathcal{A}$.

$\mathcal{A}$ returns $\hat{x}$ such that $h^s(\hat{x}) = y$. If $\hat{x} \neq x$, $\hat{\mathcal{A}}$ outputs $\hat{x}$ and wins.



Weaker Security Notions

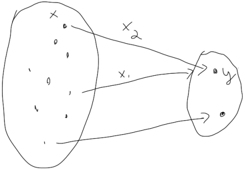
A weaker security notion, called second pre-image resistance, or target collision resistance, is implied by collision resistance.

Second Preimage Resistance:
$\mathcal{A}$ is given random key s, and a random input x.
$\mathcal{A}$ has to output $\hat{x}$ such that $h^s(\hat{x}) = h^s(x)$.

Clearly if $\mathcal{A}$ can win at this game, there exists an adversary $\hat{\mathcal{A}}$ that can win at the collision resistant game.
$\hat{\mathcal{A}}$ can simply sample x itself, run $\mathcal{A}$ to get $\hat{x}$, and output collision $x, \hat{x}$.

Preimage Resistance:
$\mathcal{A}$ is given a random key s and a random output y.
$\mathcal{A}$ has to output some x such that $h^s(x) = y$.

Claim: If $\mathcal{A}$ can win at this game, then there exists an adversary $\hat{\mathcal{A}}$ that can win at the second preimage resistance game.
$\hat{\mathcal{A}}$ is given s and x.
It computes $y = h^s(x)$, and gives s, y to $\mathcal{A}$.
$\mathcal{A}$ returns $\hat{x}$ such that $h^s(\hat{x}) = y$. If $\hat{x} \neq x$, $\hat{\mathcal{A}}$ outputs $\hat{x}$ and wins.



$\Pr[\hat{x} \neq x] = \frac{1}{2}$

Weaker Security Notions

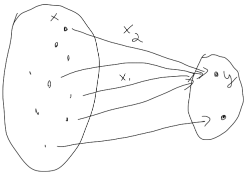
A weaker security notion, called second pre-image resistance, or target collision resistance, is implied by collision resistance.

Second Preimage Resistance:
$\mathcal{A}$ is given random key s, and a random input x.
$\mathcal{A}$ has to output $\hat{x}$ such that $h^s(\hat{x}) = h^s(x)$.

Clearly if $\mathcal{A}$ can win at this game, there exists an adversary $\hat{\mathcal{A}}$ that can win at the collision resistant game.
$\hat{\mathcal{A}}$ can simply sample x itself, run $\mathcal{A}$ to get $\hat{x}$, and output collision $x, \hat{x}$.

Preimage Resistance:
$\mathcal{A}$ is given a random key s and a random output y.
$\mathcal{A}$ has to output some x such that $h^s(x) = y$.

Claim: If $\mathcal{A}$ can win at this game, then there exists an adversary $\hat{\mathcal{A}}$ that can win at the second preimage resistance game.
$\hat{\mathcal{A}}$ is given s and x.
It computes $y = h^s(x)$, and gives s, y to $\mathcal{A}$.
$\mathcal{A}$ returns $\hat{x}$ such that $h^s(\hat{x}) = y$. If $\hat{x} \neq x$, $\hat{\mathcal{A}}$ outputs $\hat{x}$ and wins.



Weaker Security Notions

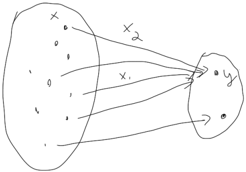
A weaker security notion, called second pre-image resistance, or target collision resistance, is implied by collision resistance.

Second Preimage Resistance:
$\mathcal{A}$ is given random key s, and a random input x.
$\mathcal{A}$ has to output $\hat{x}$ such that $h^s(\hat{x}) = h^s(x)$.

Clearly if $\mathcal{A}$ can win at this game, there exists an adversary $\hat{\mathcal{A}}$ that can win at the collision resistant game.
$\hat{\mathcal{A}}$ can simply sample x itself, run $\mathcal{A}$ to get $\hat{x}$, and output collision $x, \hat{x}$.

Preimage Resistance:
$\mathcal{A}$ is given a random key s and a random output y.
$\mathcal{A}$ has to output some x such that $h^s(x) = y$.

Claim: If $\mathcal{A}$ can win at this game, then there exists an adversary $\hat{\mathcal{A}}$ that can win at the second preimage resistance game.
$\hat{\mathcal{A}}$ is given s and x.
It computes $y = h^s(x)$, and gives s, y to $\mathcal{A}$.
$\mathcal{A}$ returns $\hat{x}$ such that $h^s(\hat{x}) = y$. If $\hat{x} \neq x$, $\hat{\mathcal{A}}$ outputs $\hat{x}$ and wins.



$$\Pr[\hat{x} \neq x] = \frac{|h^{-1}(y)|-1}{|h^{-1}(y)|}$$