On the Indifferentiability of the Grøstl Hash Function

Bart Mennink (K.U.Leuven)

Joint work with: Elena Andreeva and Bart Preneel

Security and Cryptography for Networks
Amalfi, Italy

September 13, 2010

Outline

1. Preliminaries
2. Differentiability of Grøstail
3. Indifferentiability of Grøstl
4. Conclusions

Outline

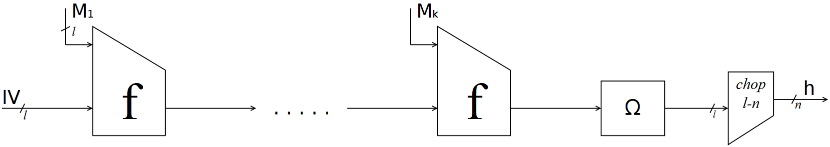
1 Preliminaries

2 Differentiability of Grøstail

3 Indifferentiability of Grøstl

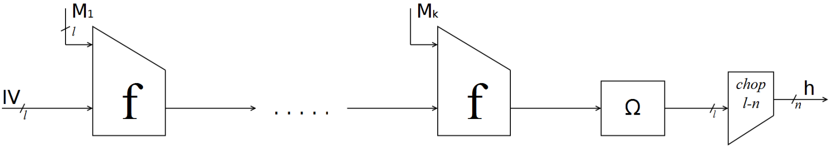
4 Conclusions

Grøstl



- Grøstl: a second round SHA-3 candidate
- Grøstl supports digests of length $n = 224, 256, 384$ or 512 bits
- State size l larger than output size n: $l \geq 2n$

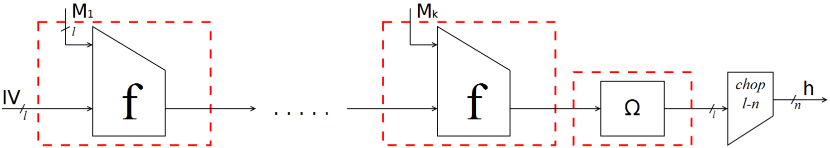
Grøstl



- Grøstl: a second round SHA-3 candidate
- Grøstl supports digests of length $n = 224, 256, 384$ or 512 bits
- State size l larger than output size n: $l \geq 2n$

- Messages of arbitrary length injectively padded into $(M_1, \ldots, M_k)$

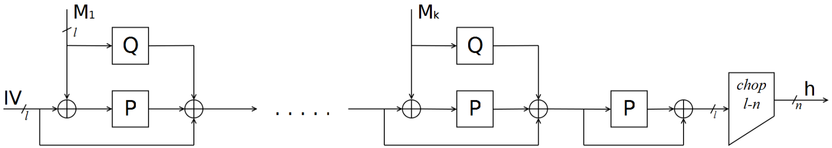
Grøstl



- Grøstl: a second round SHA-3 candidate
- Grøstl supports digests of length $n = 224, 256, 384$ or 512 bits
- State size l larger than output size n: $l \geq 2n$

- Messages of arbitrary length injectively padded into $(M_1, \ldots, M_k)$
- f and Ω are based on two l-bit permutations P, Q
  - Compression function: $f(h, m) = P(h \oplus m) \oplus Q(m) \oplus h$
  - Final transformation: $\Omega(h) = P(h) \oplus h$

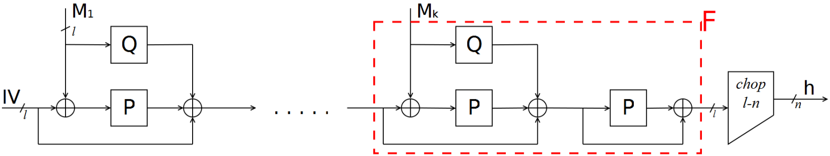
Grøstl



- Grøstl: a second round SHA-3 candidate
- Grøstl supports digests of length $n = 224, 256, 384$ or 512 bits
- State size l larger than output size n: $l \geq 2n$

- Messages of arbitrary length injectively padded into $(M_1, \ldots, M_k)$
- f and Ω are based on two l-bit permutations P, Q
  - Compression function: $f(h, m) = P(h \oplus m) \oplus Q(m) \oplus h$
  - Final transformation: $\Omega(h) = P(h) \oplus h$

Grøstail



- Grøstail: last compression function with the final transformation (i.e., Grøstail is the 'tail' of Grøstl)

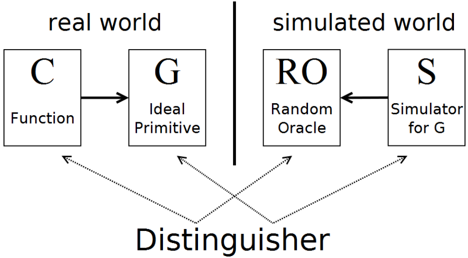
$$\mathrm{F}(h, m) = P(f(h, m)) \oplus f(h, m)$$

Indifferentiability

- Distinguisher $\mathcal{D}$ tries to differentiate a function C from a random oracle RO. $\mathcal{D}$ is allowed to know the underlying structure of C

Indifferentiability

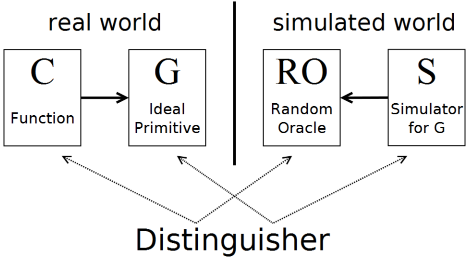
- Distinguisher $\mathcal{D}$ tries to differentiate a function C from a random oracle RO. $\mathcal{D}$ is allowed to know the underlying structure of C



- C with oracle access to G is indifferentiable from RO if there exists a simulator S such that for any $\mathcal{D}$ these games are indistinguishable

Indifferentiability

- Distinguisher $\mathcal{D}$ tries to differentiate a function C from a random oracle RO. $\mathcal{D}$ is allowed to know the underlying structure of C

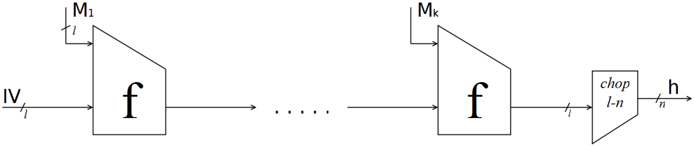


- C with oracle access to G is indifferentiable from RO if there exists a simulator S such that for any $\mathcal{D}$ these games are indistinguishable
- Throughout: G denotes (P, Q), and C is either Grøstl or Grøstail

Assumption: P, Q are independent random permutations

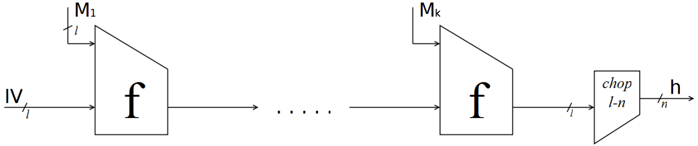
Examples of Indifferentiable Constructions

- For f a random function (chop-Merkle-Damgård) [CDMP05]

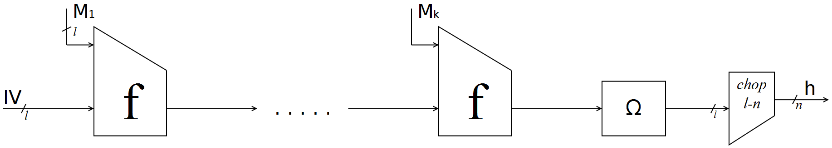


Examples of Indifferentiable Constructions

- For f a random function (chop-Merkle-Damgård) [CDMP05]

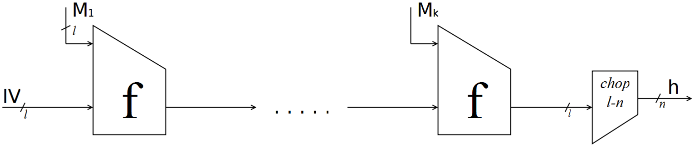


- Application to Grøstl:

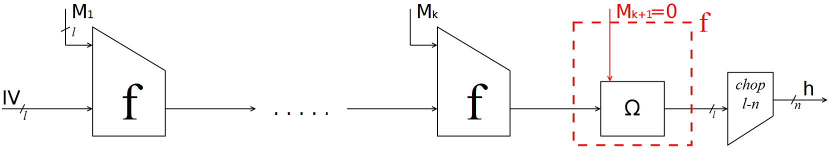


Examples of Indifferentiable Constructions

- For f a random function (chop-Merkle-Damgård) [CDMP05]

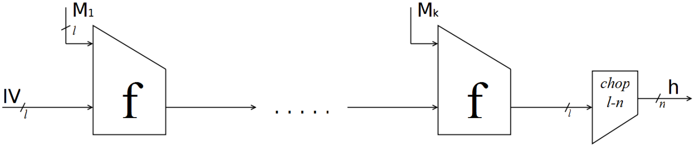


- Application to Grøstl:

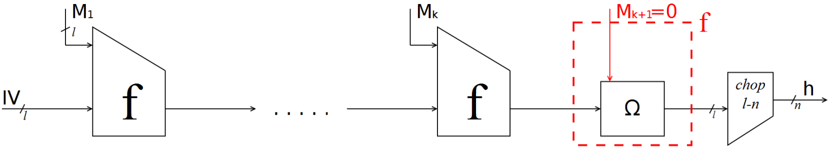


Examples of Indifferentiable Constructions

- For f a random function (chop-Merkle-Damgård) [CDMP05]



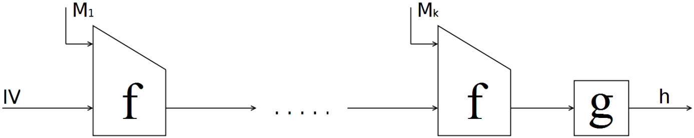
- Application to Grøstl:



- However, for Grøstl, f is non-random (fixed-points $f(h, m) = h$)

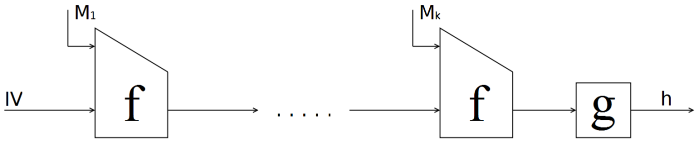
Examples of Indifferentiable Constructions

- For f ‘preimage aware’ and g a random function [DRS09]

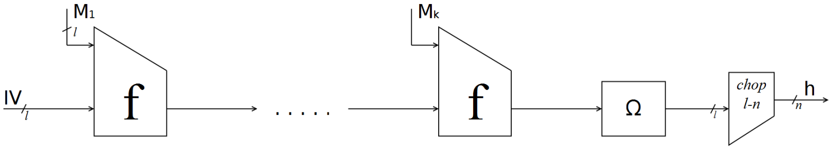


Examples of Indifferentiable Constructions

- For f 'preimage aware' and g a random function [DRS09]

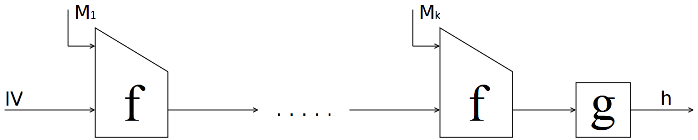


- Application to Grøstl:

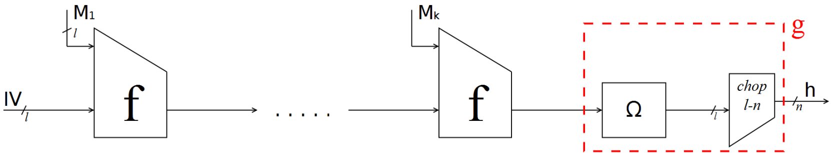


Examples of Indifferentiable Constructions

- For f 'preimage aware' and g a random function [DRS09]

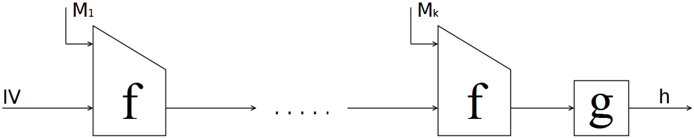


- Application to Grøstl:

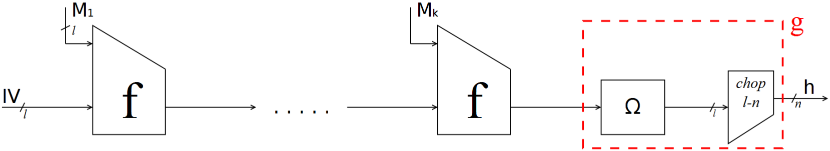


Examples of Indifferentiable Constructions

- For f 'preimage aware' and g a random function [DRS09]



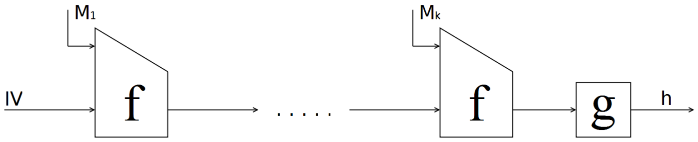
- Application to Grøstl:



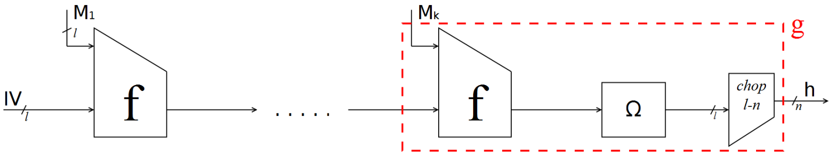
- However, for Grøstl, Ω is non-random (fixed-points $\Omega(h) = h$)

Examples of Indifferentiable Constructions

- For f 'preimage aware' and g a random function [DRS09]



- Application to Grøstl:



- However, for Grøstl, Ω is non-random (fixed-points $\Omega(h) = h$)
- Second attempt: consider $g := \mathrm{chop} \circ \Omega \circ f = \mathrm{chop} \circ F$

Outline

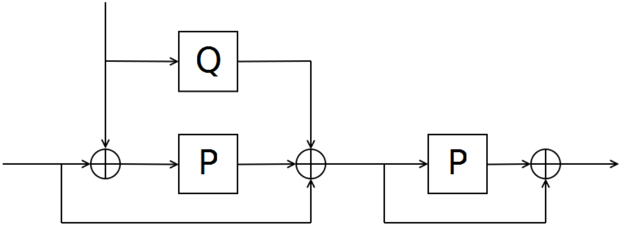
1 Preliminaries

2 Differentiability of Grøstail

3 Indifferentiability of Grøstl

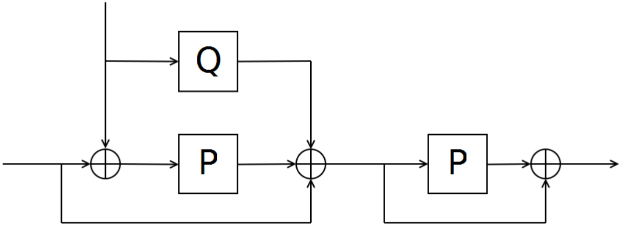
4 Conclusions

Differentiability of Grøstail



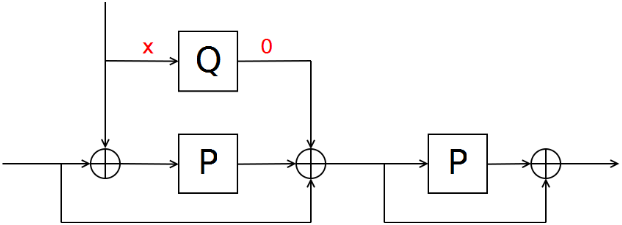
- Design a distinguisher that tricks *any* simulator

Differentiability of Grøstail



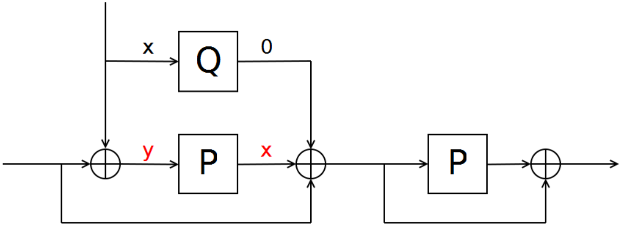
- Design a distinguisher that tricks *any* simulator
- Distinguisher is based on fixed-points for Grøstail

Differentiability of Grøstail



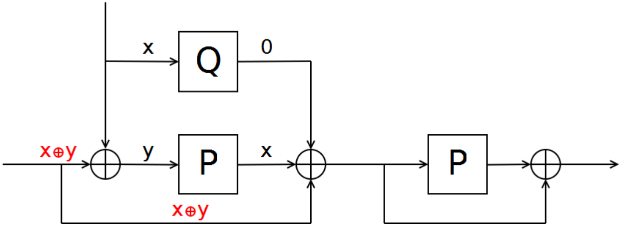
- Design a distinguisher that tricks *any* simulator
- Distinguisher is based on fixed-points for Grøstail
  - Inverse query 0 to Q^{-1} and obtain $x = Q^{-1}(0)$

Differentiability of Grøstail



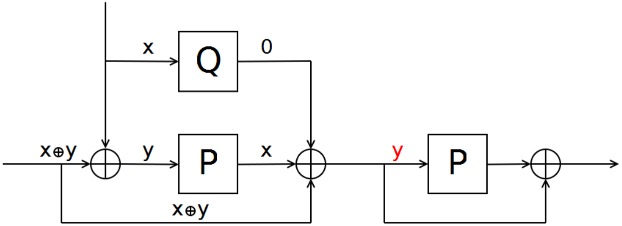
- Design a distinguisher that tricks *any* simulator
- Distinguisher is based on fixed-points for Grøstail
  - Inverse query 0 to Q^{-1} and obtain $x = Q^{-1}(0)$
  - Inverse query x to P^{-1} and obtain $y = P^{-1}(x)$

Differentiability of Grøstail



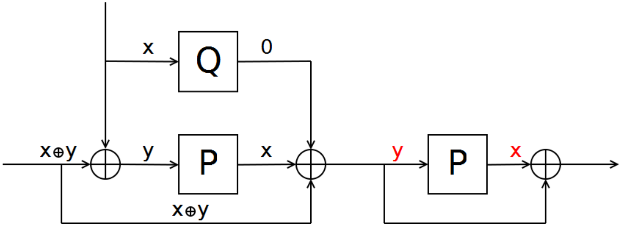
- Design a distinguisher that tricks *any* simulator
- Distinguisher is based on fixed-points for Grøstail
  - Inverse query 0 to Q^{-1} and obtain $x = Q^{-1}(0)$
  - Inverse query x to P^{-1} and obtain $y = P^{-1}(x)$

Differentiability of Grøstail



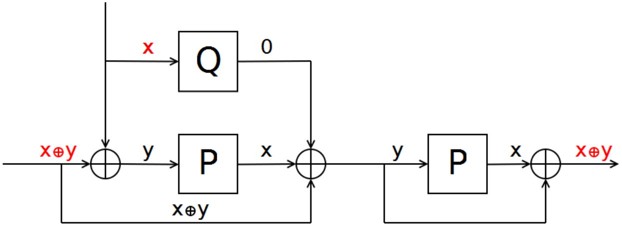
- Design a distinguisher that tricks *any* simulator
- Distinguisher is based on fixed-points for Grøstail
  - Inverse query 0 to Q^{-1} and obtain $x = Q^{-1}(0)$
  - Inverse query x to P^{-1} and obtain $y = P^{-1}(x)$

Differentiability of Grøstail



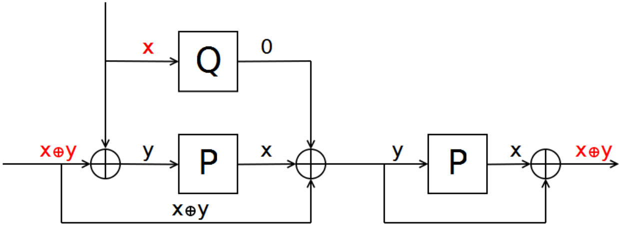
- Design a distinguisher that tricks *any* simulator
- Distinguisher is based on fixed-points for Grøstail
  - Inverse query 0 to Q^{-1} and obtain $x = Q^{-1}(0)$
  - Inverse query x to P^{-1} and obtain $y = P^{-1}(x)$

Differentiability of Grøstail



- Design a distinguisher that tricks *any* simulator
- Distinguisher is based on fixed-points for Grøstail
  - Inverse query 0 to Q^{-1} and obtain $x = Q^{-1}(0)$
  - Inverse query x to P^{-1} and obtain $y = P^{-1}(x)$
  - For Grøstail, we have $F(x \oplus y, x) = x \oplus y$

Differentiability of Grøstail



- Design a distinguisher that tricks *any* simulator
- Distinguisher is based on fixed-points for Grøstail
  - Inverse query 0 to Q^{-1} and obtain $x = Q^{-1}(0)$
  - Inverse query x to P^{-1} and obtain $y = P^{-1}(x)$
  - For Grøstail, we have $F(x \oplus y, x) = x \oplus y$
- The simulator answers queries to Q^{-1} and P^{-1} such that the same relation holds for the random oracle: $RO(x \oplus y, x) = x \oplus y$
  $\rightarrow$ fixed-point for RO

Outline

1 Preliminaries

2 Differentiability of Grøstail

3 Indifferentiability of Grøstl
- How to Prove Indifferentiability?
- Design of the Simulator

4 Conclusions

How to Prove Indifferentiability?

- Construct a simulator S that mimics behavior of P, Q
- Answers from simulator need to be consistent with RO:

  any relation between Grøstl and P, Q, also holds between RO and S

- Simulator has access to RO

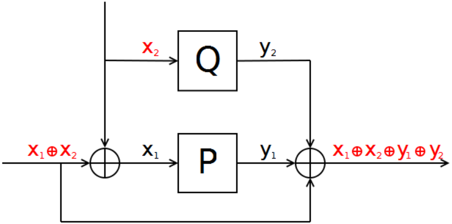
How to Prove Indifferentiability?

- Simulator maintains a graph (V, E): edges represent evaluations of f

How to Prove Indifferentiability?

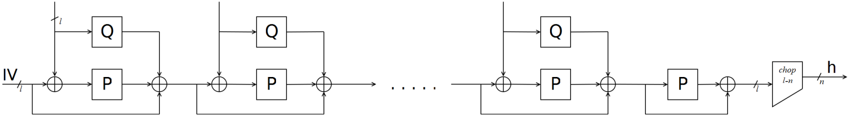
- Simulator maintains a graph (V, E): edges represent evaluations of f

- Suppose $P(x_1) = y_1$ and $Q(x_2) = y_2$
  Then $f(x_1 \oplus x_2, x_2) = x_1 \oplus x_2 \oplus y_1 \oplus y_2$



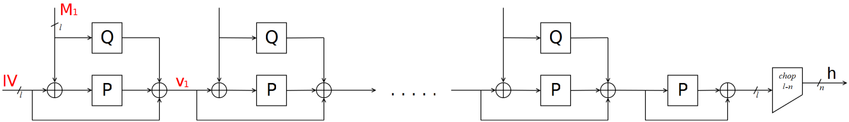
- Edge denoted by $x_1 \oplus x_2 \xrightarrow{x_2} x_1 \oplus x_2 \oplus y_1 \oplus y_2$

How to Prove Indifferentiability?



- Simulator considers tree $(r(V), E)$ starting from IV
- Suppose $IV \xrightarrow{M_1} v_1 \xrightarrow{M_2} v_2 \cdots v_{k-1} \xrightarrow{M_k} v_k$ is in the tree

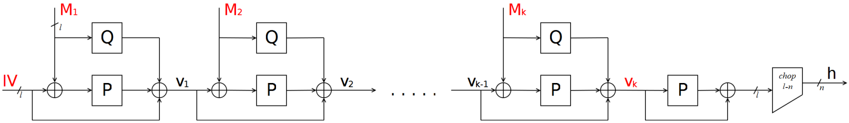
How to Prove Indifferentiability?



- Simulator considers tree $(r(V), E)$ starting from IV
- Suppose $IV \xrightarrow{M_1} v_1 \xrightarrow{M_2} v_2 \cdots v_{k-1} \xrightarrow{M_k} v_k$ is in the tree

  $f(IV, M_1) = v_1$

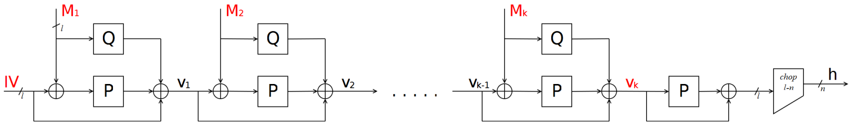
How to Prove Indifferentiability?



- Simulator considers tree $(r(V), E)$ starting from IV
- Suppose $IV \xrightarrow{M_1} v_1 \xrightarrow{M_2} v_2 \cdots v_{k-1} \xrightarrow{M_k} v_k$ is in the tree

$$f(IV, M_1) = v_1 \qquad f(v_1, M_2) = v_2 \qquad \dots \qquad f(v_{k-1}, M_k) = v_k$$

How to Prove Indifferentiability?

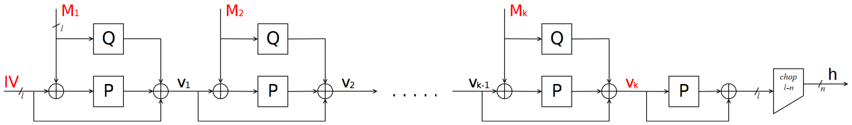


- Simulator considers tree $(r(V), E)$ starting from IV
- Suppose $IV \xrightarrow{M_1} v_1 \xrightarrow{M_2} v_2 \cdots v_{k-1} \xrightarrow{M_k} v_k$ is in the tree

$$f(IV, M_1) = v_1 \qquad f(v_1, M_2) = v_2 \qquad \dots \qquad f(v_{k-1}, M_k) = v_k$$

- Suppose $(M_1, \dots, M_k) = \mathrm{pad}(M)$ for some M
  $\rightarrow P(v_k)$ *should* satisfy $\mathrm{chop}_{l-n}(P(v_k) \oplus v_k) = RO(M)$

How to Prove Indifferentiability?



- Simulator considers tree $(r(V), E)$ starting from IV
- Suppose $IV \xrightarrow{M_1} v_1 \xrightarrow{M_2} v_2 \cdots v_{k-1} \xrightarrow{M_k} v_k$ is in the tree

$$f(IV, M_1) = v_1 \qquad f(v_1, M_2) = v_2 \qquad \dots \qquad f(v_{k-1}, M_k) = v_k$$

- Suppose $(M_1, \dots, M_k) = \mathrm{pad}(M)$ for some M
  $\rightarrow P(v_k)$ *should* satisfy $\mathrm{chop}_{l-n}(P(v_k) \oplus v_k) = RO(M)$

Simulator can only guarantee this if v_k is queried to P *after* path to v_k is paved!

Design of the Simulator

- Denote $\bar{r}(V)$: nodes in $r(V)$ labeled by a correctly padded message

Design of the Simulator

- Denote $\bar{r}(V)$: nodes in $r(V)$ labeled by a correctly padded message

- Simulator mimics $P, Q \rightarrow$ four interfaces $\mathsf{S}_P, \mathsf{S}_Q, \mathsf{S}_{P^{-1}}, \mathsf{S}_{Q^{-1}}$
- Simulator increases $\bar{r}(V) \cap \mathrm{dom}(P)$ and $r(V)$ as little as possible
  - $\bar{r}(V) \cap \mathrm{dom}(P)$: if $IV \xrightarrow{M} v$, we require $\mathrm{chop}_{l-n}(P(v) \oplus v) = RO(M)$
  - $r(V)$: each path in $r(V)$ may eventually lead to a node in $\bar{r}(V)$

Design of the Simulator

- Denote $\bar{r}(V)$: nodes in $r(V)$ labeled by a correctly padded message

- Simulator mimics $P, Q \rightarrow$ four interfaces $\mathsf{S}_P, \mathsf{S}_Q, \mathsf{S}_{P^{-1}}, \mathsf{S}_{Q^{-1}}$
- Simulator increases $\bar{r}(V) \cap \mathrm{dom}(P)$ and $r(V)$ as little as possible
  - $\bar{r}(V) \cap \mathrm{dom}(P)$: if $IV \xrightarrow{M} v$, we require $\mathrm{chop}_{l-n}(P(v) \oplus v) = RO(M)$
  - $r(V)$: each path in $r(V)$ may eventually lead to a node in $\bar{r}(V)$

- Sometimes, simulator may be forced to increase these sets

	S_P	S_Q	$\mathsf{S}_{P^{-1}}$	$\mathsf{S}_{Q^{-1}}$
$\bar{r}(V) \cap \mathrm{dom}(P)$	assure $\mathrm{chop}(P(v) \oplus v) = RO(M)$	—	—	—
$r(V)$	assure min. increase	assure min. increase	—	—

On query $S_P(x_1) \to y_1$:

$y_1 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathrm{rng}(P)$
if S is forced to increase $r(V)$:
    assure minimal increase
if S is forced to increase $\bar{r}(V) \cap \mathrm{dom}(P)$:
    assure $\mathrm{chop}_{l-n}(y_1 \oplus x_1) = RO(M)$

On query $S_Q(x_2) \to y_2$:

$y_2 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathrm{rng}(Q)$
if S is forced to increase $r(V)$:
    assure minimal increase
avoid increasing $\bar{r}(V) \cap \mathrm{dom}(P)$

On query $S_{P-1}(y_1) \to x_1$:

$x_1 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathrm{dom}(P)$
avoid increasing $r(V)$
avoid increasing $\bar{r}(V) \cap \mathrm{dom}(P)$

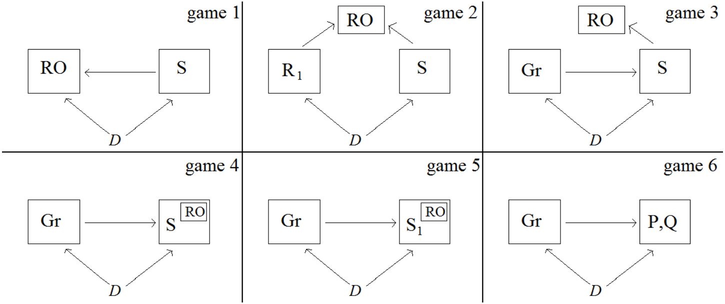
On query $S_{Q-1}(y_2) \to x_2$:

$x_2 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathrm{dom}(Q)$
avoid increasing $r(V)$

- In every 'assure' and 'avoid', there is a **GOTO**-statement

	S_P	S_Q	S_{P-1}	S_{Q-1}
$\bar{r}(V) \cap \mathrm{dom}(P)$	assure $\mathrm{chop}(P(v) \oplus v) = RO(M)$	—	—	—
$r(V)$	assure min. increase	assure min. increase	—	—

Proof (idea)



- R_1 is a relay algorithm
- S_1 is S with **GOTO**-statements removed

$$\mathrm{Adv}(\mathcal{D}) = O(\Pr(\textbf{GOTO}\text{ is executed})) = O\left(\frac{(Kq)^4}{2^l}\right)$$

($\mathcal{D}$ makes at most q queries of length at most K message blocks)

Our Result

$$\mathrm{Adv}(\mathcal{D}) = O\left(\frac{(Kq)^4}{2^l}\right)$$

($\mathcal{D}$ makes at most q queries of length at most K message blocks)

- Grøstl behaves like a random oracle up to $2^{n/2}$ queries (recall $l \geq 2n$)

Our Result

$$\mathrm{Adv}(\mathcal{D}) = O\left(\frac{(Kq)^4}{2^l}\right)$$

($\mathcal{D}$ makes at most q queries of length at most K message blocks)

- Grøstl behaves like a random oracle up to $2^{n/2}$ queries (recall $l \geq 2n$)
- Other second round SHA-3 candidates, like JH, Keccak and Shabal, have recently also been proven indifferentiable

	bound for $n = 256$
Grøstl	$O(q^4/2^{512})$
JH	$O(q^3/2^{512})$
Keccak	$O(q^2/2^{512})$
Shabal	$O(q^2/2^{896})$

Our Result

$$\mathrm{Adv}(\mathcal{D}) = O\left(\frac{(Kq)^4}{2^l}\right)$$

($\mathcal{D}$ makes at most q queries of length at most K message blocks)

- Grøstl behaves like a random oracle up to $2^{n/2}$ queries (recall $l \geq 2n$)
- Other second round SHA-3 candidates, like JH, Keccak and Shabal, have recently also been proven indifferentiable

	bound for $n = 256$	state size
Grøstl	$O(q^4/2^{512})$	$l = 512$
JH	$O(q^3/2^{512})$	$l = 1024$
Keccak	$O(q^2/2^{512})$	$l = 1600$
Shabal	$O(q^2/2^{896})$	$l = 1408$

Our Result

$$\mathrm{Adv}(\mathcal{D}) = O\left(\frac{(Kq)^4}{2^l}\right)$$

($\mathcal{D}$ makes at most q queries of length at most K message blocks)

- Grøstl behaves like a random oracle up to $2^{n/2}$ queries (recall $l \geq 2n$)
- Other second round SHA-3 candidates, like JH, Keccak and Shabal, have recently also been proven indifferentiable

	bound for $n = 256$	state size
Grøstl	$O(q^4/2^{512})$	$l = 512$
JH	$O(q^3/2^{512})$	$l = 1024$
Keccak	$O(q^2/2^{512})$	$l = 1600$
Shabal	$O(q^2/2^{896})$	$l = 1408$

- Would Grøstl have state size 1024, we would have $O(q^4/2^{1024})$

Outline

1. Preliminaries
2. Differentiability of Grøstail
3. Indifferentiability of Grøstl
4. Conclusions

Conclusions

- We proved the indifferentiability of Grøstl, under the assumption that P, Q are two independent random permutations
- Grøstl behaves like a random oracle up to $2^{n/2}$ queries

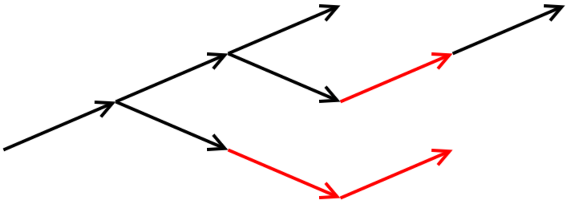
- Existing results could not be carried over to Grøstl:
  - Compression function and final transformation are differentiable
  - We additionally demonstrated that Grøstail is differentiable

- Open problem: improving the bound

Thank you for your attention!

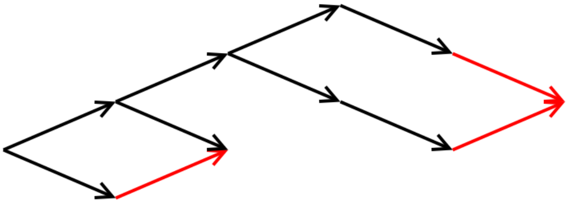
SUPPORTING SLIDES!!!

Design of the Simulator



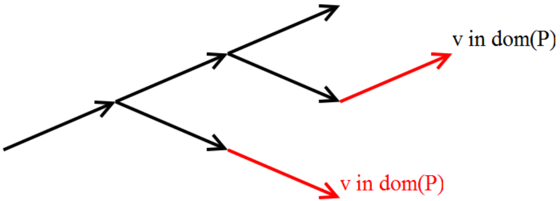
- Black = old edges, Red = newly added edges in a certain query
- When adding edges to tree, make sure that:
  1. End node of edge has no outgoing edge in the updated graph

Design of the Simulator



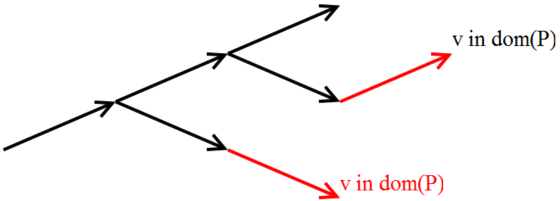
- Black = old edges, Red = newly added edges in a certain query
- When adding edges to tree, make sure that:
  1. End node of edge has no outgoing edge in the updated graph
  2. End node of edge has no incoming edge in the updated graph (except for the edge itself, of course)

Design of the Simulator



- Black = old edges, Red = newly added edges in a certain query
- When adding edges to tree, make sure that:
  1. End node of edge has no outgoing edge in the updated graph
  2. End node of edge has no incoming edge in the updated graph (except for the edge itself, of course)
  3. $\bar{r}(V)$ is *never* increased with a node in the updated $\mathrm{dom}(P)$

Design of the Simulator



- Black = old edges, Red = newly added edges in a certain query
- When adding edges to tree, make sure that:
  1. End node of edge has no outgoing edge in the updated graph
  2. End node of edge has no incoming edge in the updated graph (except for the edge itself, of course)
  3. $\bar{r}(V)$ is *never* increased with a node in the updated $\mathrm{dom}(P)$
- In general: $r(V)$ and $\bar{r}(V) \cap \mathrm{dom}(P)$ should be increased as little as possible

Design of the Simulator

- In inverse queries, the simulator can always avoid increasing $r(V)$ and $\bar{r}(V) \cap \mathrm{dom}(P)$

Design of the Simulator

- In inverse queries, the simulator can always avoid increasing $r(V)$ and $\bar{r}(V) \cap \mathrm{dom}(P)$
- In forward queries, the simulator may be *forced* to increase $r(V)$ or $\bar{r}(V) \cap \mathrm{dom}(P)$
  - Simulator is forced to increase $r(V)$
    - Make sure that 1.-3. (of previous slide) are satisfied

Design of the Simulator

- In inverse queries, the simulator can always avoid increasing $r(V)$ and $\bar{r}(V) \cap \mathrm{dom}(P)$
- In forward queries, the simulator may be *forced* to increase $r(V)$ or $\bar{r}(V) \cap \mathrm{dom}(P)$
  - Simulator is forced to increase $r(V)$
    - Make sure that 1.-3. (of previous slide) are satisfied
  - Simulator is forced to increase $\bar{r}(V) \cap \mathrm{dom}(P)$
    - This only happens if $\mathcal{D}$ queries $P(v)$ for $v \in \bar{r}(V)$
    - Simulator uses RO to output $P(v)$ such that $\mathrm{chop}_{l-n}(P(v) \oplus v) = RO(M)$
- Notice that $\bar{r}(V)$ only increases if $r(V)$ increases

Proof (idea)

$$\Pr\left(\mathcal{D}^{RO,\mathsf{S}^{RO}}=1\right)=\Pr\left(\mathcal{D}^{G_1}=1\right)$$

$$\Pr\left(\mathcal{D}^{G_1}=1\right)=\Pr\left(\mathcal{D}^{G_2}=1\right)$$

$$\left|\Pr\left(\mathcal{D}^{G_2}=1\right)-\Pr\left(\mathcal{D}^{G_3}=1\right)\right|\leq\Pr\left(\mathcal{D}^{G_2}\text{ sets }\textbf{bad}\right)+\Pr\left(\mathcal{D}^{G_3}\text{ sets }\textbf{bad}\right)$$

$$\Pr\left(\mathcal{D}^{G_3}=1\right)=\Pr\left(\mathcal{D}^{G_4}=1\right)$$

$$\left|\Pr\left(\mathcal{D}^{G_4}=1\right)-\Pr\left(\mathcal{D}^{G_5}=1\right)\right|\leq\Pr\left(\mathcal{D}^{G_4}\text{ sets }\textbf{bad}\right)$$

$$\left|\Pr\left(\mathcal{D}^{G_5}=1\right)-\Pr\left(\mathcal{D}^{G_6}=1\right)\right|\leq\frac{r_P^2}{2^l}$$

$$\Pr\left(\mathcal{D}^{G_6}=1\right)=\Pr\left(\mathcal{D}^{\mathsf{Gr}^{P,Q},(P,Q)}=1\right)$$

$$\begin{aligned}\left|\Pr\left(\mathcal{D}^{\mathsf{Gr}^{P,Q},(P,Q)}=1\right)-\Pr\left(\mathcal{D}^{RO,\mathsf{S}^{RO}}=1\right)\right|&\leq 3\Pr\left(\mathcal{D}^{G_4}\text{ sets }\textbf{bad}\right)+\frac{r_P^2}{2^l}\\&=O\left(\frac{(Kq)^4}{2^l}\right)\end{aligned}$$