Radboud University

Security of Truncated Permutation Without Initial Value

Lorenzo Grassi and Bart Mennink

December 2022

Radboud University, Nijmegen, The Netherlands

Motivation and Overview

(1) A truncated permutation applied to the concatenation of an input value with a **fixed** *IV* is *indifferentiable* from a random function up to a certain bound;

(2) We prove that (almost) the same bound holds **if** the fixed value is replaced by a randomized value;

(3) Concrete applications in the context of parallel variable-length digest generation.

Preliminary

About Public Permutations and Functions

- Many symmetric cryptographic primitives are based on public permutations and/or functions:
  - Even-Mansour construction $x \mapsto k + \mathcal{P}(x + k)$ for a public permutation $\mathcal{P}$;
  - Merkle-Damgård construction instantiated by a public compression function $\mathcal{F}$;
  - sponge/duplex construction instantiated either by a public permutation $\mathcal{P}$ or by a public function $\mathcal{F}$.

- We know how to construct a public permutation $\mathcal{P}$ with "good" properties (e.g., via the SPN strategy).

- *What about the case of public functions?*

About Public Permutations and Functions

- Many symmetric cryptographic primitives are based on public permutations and/or functions:
  - Even-Mansour construction $x \mapsto k + \mathcal{P}(x + k)$ for a public permutation $\mathcal{P}$;
  - Merkle-Damgård construction instantiated by a public compression function $\mathcal{F}$;
  - sponge/duplex construction instantiated either by a public permutation $\mathcal{P}$ or by a public function $\mathcal{F}$.

- We know how to construct a public permutation $\mathcal{P}$ with "good" properties (e.g., via the SPN strategy).

- *What about the case of public functions?*

SOP Function

- Sum of permutations:
  - let $\mathcal{P}_0, \mathcal{P}_1$ be two public (random) permutations over $(\{0,1\}^b, +)$;
  - the Sum-Of-Permutations (SOP) function

$$\mathrm{SOP}(x) = \mathcal{P}_0(x) + \mathcal{P}_1(x)$$

    is indifferentiability from a random function up to 2^b queries [BN18];
- Analogous result holds given a single public (random) permutation $\mathcal{P}$:

$$\mathrm{SOP}'(x) = \mathcal{P}(x\|0) + \mathcal{P}(x\|1)\,.$$

SOP Function

- Sum of permutations:
  - let $\mathcal{P}_0, \mathcal{P}_1$ be two public (random) permutations over $(\{0,1\}^b, +)$;
  - the Sum-Of-Permutations (SOP) function

$$\mathrm{SOP}(x) = \mathcal{P}_0(x) + \mathcal{P}_1(x)$$

    is indifferentiability from a random function up to 2^b queries [BN18];
- Analogous result holds given a single public (random) permutation $\mathcal{P}$:

$$\mathrm{SOP}'(x) = \mathcal{P}(x\|0) + \mathcal{P}(x\|1)\,.$$

TRUNC Function

- Let $\mathcal{P}$ be a public (random) permutation over $(\{0,1\}^b, +)$;
- Let the truncation function

$$\mathrm{TRUNC} : \{0,1\}^{b-m} \to \{0,1\}^n$$

be defined as

$$\mathrm{TRUNC}(x) = \mathrm{left}_n \mathcal{P}(IV \| x)\,,$$

where

- $IV \in \{0,1\}^m$ is an initial value;
- $\mathrm{left}_n : \{0,1\}^b \to \{0,1\}^n$ returns the n leftmost bits (where $n < b$).

Security of TRUNC

Given

- $m =$ size of IV,
- $n =$ output size of the TRUNC,
- $b =$ size of $\mathcal{P}$,

then

- Dodis et al. [DRR+09] proved that TRUNC is indifferentiable from a random function up to
$$\approx \min\left\{2^{\frac{b-n}{2}}, 2^{\frac{b}{2}}, 2^m\right\} \text{ queries};$$
- Choi et al. [CLL19] proved that it is indifferentiable from a random function up to
$$\approx \min\left\{2^{\frac{2b-n}{3}}, \frac{2^{b-n}}{b-n}, 2^m\right\} \text{ queries}.$$

From a Fixed to a Randomized Initial Value

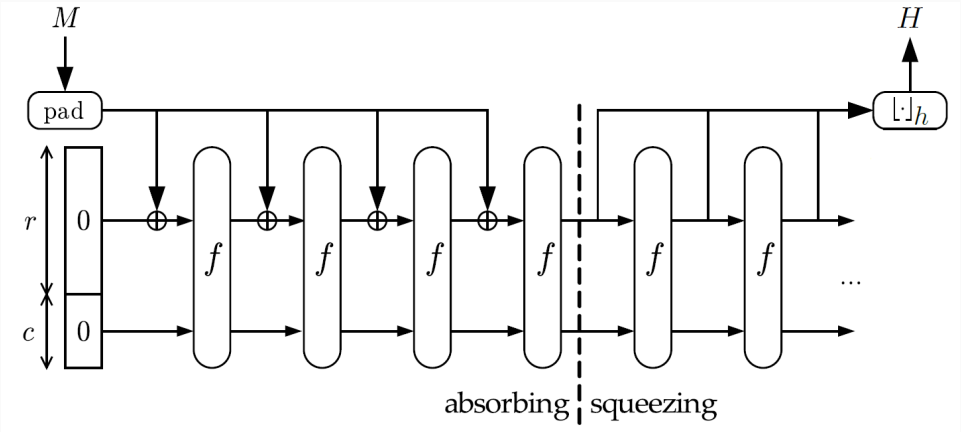
About the Initial Value

- Initial value is crucial for indifferentiability!
- If *omitted* (hence, $m = 0$), the obtained TRUNC function $\mathcal{F}(x) = \mathrm{left}_n(\mathcal{P}(x))$ can be easily distinguished from a random function:
  - $y \xleftarrow{\$} \{0,1\}^b$, and let $x = \mathcal{P}^{-1}(y)$;
  - if $\mathrm{left}_n(y) \neq \mathcal{O}(x)$, then $\mathcal{O}$ is the *random function* with prob. 1.
- *The initial value is overkill:* hash function "instantiated" by a truncated permutation with $m = 0$ can be still secure!

About the Initial Value

- Initial value is crucial for indifferentiability!
- If *omitted* (hence, $m = 0$), the obtained TRUNC function $\mathcal{F}(x) = \mathsf{left}_n(\mathcal{P}(x))$ can be easily distinguished from a random function:
  - $y \xleftarrow{\$} \{0,1\}^b$, and let $x = \mathcal{P}^{-1}(y)$;
  - if $\mathsf{left}_n(y) \neq \mathcal{O}(x)$, then $\mathcal{O}$ is the *random function* with prob. 1.

- *The initial value is overkill:* hash function "instantiated" by a truncated permutation with $m = 0$ can be still secure!



Truncation Without Fixed *IV*

- *Idea:* Replace the *fixed* initial value with a **randomized** one!
- From

$$\mathrm{TRUNC}(x) = \mathrm{left}_n \mathcal{P}(IV \| x)\,,$$

to

$$\mathrm{RTRUNC}(M, x) = \mathrm{left}_n \mathcal{P}(\mathcal{H}(M) \| x)$$

where

- $\mathcal{H} : \{0,1\}^\star \to \{0,1\}^m$ is a hash function;
- $M \in \{0,1\}^\star$ and $x \in \{0,1\}^{b-m}$;
- $\mathcal{P}(\cdot)$, IV, $\mathrm{left}_n(\cdot)$ as before.

Security of RTRUNC

Under the assumption that $\mathcal{H}$ is a random oracle and $\mathcal{P}$ is a random permutation, the RTRUNC construction is indifferentiable from a random oracle up to

$$\approx \min\left\{2^{\frac{2b-n}{3}}, \frac{2^{b-n}}{b-n}, 2^{\frac{m}{2}}\right\}$$

queries, where $m =$ output size of $\mathcal{H}$, and (as before) $n =$ output size of the RTRUNC, $b =$ size of $\mathcal{P}$.

For k bits of security:

$$m \geq 2 \cdot k\,,$$

$$b \geq \max\left\{m, k+n, \frac{3 \cdot k + n}{2}\right\}\,.$$

Security of RTRUNC

Under the assumption that $\mathcal{H}$ is a random oracle and $\mathcal{P}$ is a random permutation, the RTRUNC construction is indifferentiable from a random oracle up to

$$\approx \min\left\{2^{\frac{2b-n}{3}}, \frac{2^{b-n}}{b-n}, 2^{\frac{m}{2}}\right\}$$

queries, where $m =$ output size of $\mathcal{H}$, and (as before) $n =$ output size of the RTRUNC, $b =$ size of $\mathcal{P}$.

For k bits of security:

$$m \geq 2 \cdot k\,,$$
$$b \geq \max\left\{m, k+n, \frac{3 \cdot k + n}{2}\right\}\,.$$

Sketch of the Proof (1/2)

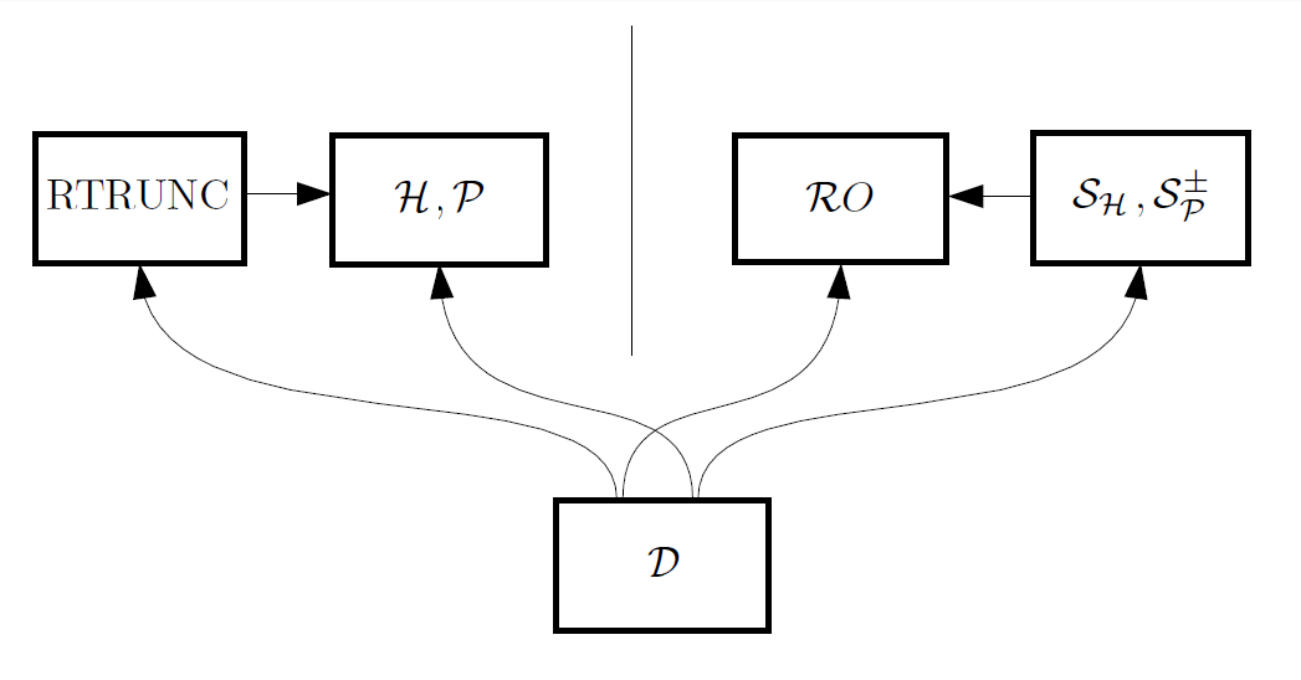


Figure: The indifferentiability model [MRH04, CDM+05].

Sketch of the Proof (2/2)

- Strategy of the proof similar to the one proposed by Choi et al. [CLL19] (but with some crucial differences!): see the paper for details;
- For comparison:
  - TRUNC is indifferentiable from a random oracle up to

$$\approx \min\left\{2^{\frac{2b-n}{3}}, \frac{2^{b-n}}{b-n}, \mathbf{2^m}\right\} \text{ queries;}$$

  - RTRUNC is indifferentiable from a random oracle up to

$$\approx \min\left\{2^{\frac{2b-n}{3}}, \frac{2^{b-n}}{b-n}, \mathbf{2^{\frac{m}{2}}}\right\} \text{ queries;}$$

  - *Difference:* existence of collisions at the output of $\mathcal{H}$.

Sketch of the Proof (2/2)

- Strategy of the proof similar to the one proposed by Choi et al. [CLL19] (but with some crucial differences!): see the paper for details;
- For comparison:
  - TRUNC is indifferentiable from a random oracle up to

$$\approx \min\left\{2^{\frac{2b-n}{3}}, \frac{2^{b-n}}{b-n}, \mathbf{2^m}\right\} \text{ queries;}$$

  - RTRUNC is indifferentiable from a random oracle up to

$$\approx \min\left\{2^{\frac{2b-n}{3}}, \frac{2^{b-n}}{b-n}, \mathbf{2^{\frac{m}{2}}}\right\} \text{ queries;}$$

  - *Difference:* existence of collisions at the output of $\mathcal{H}$.

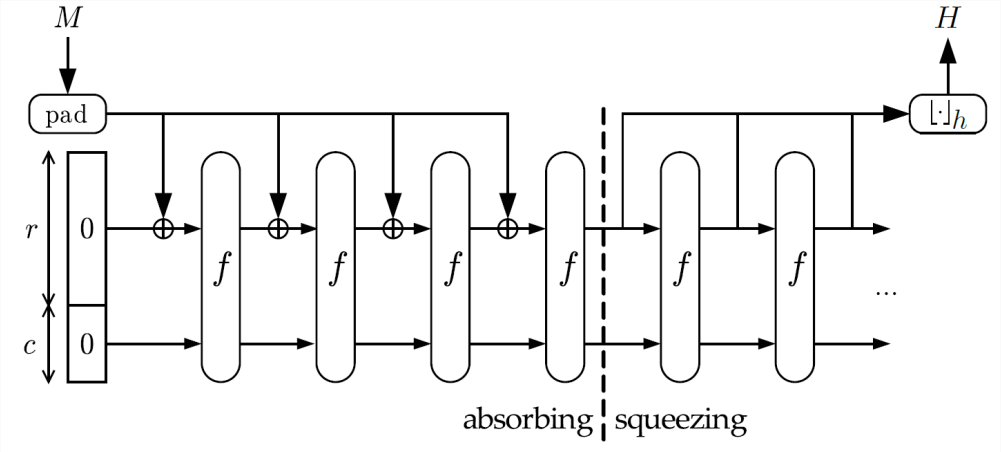
Applications

Parallel Digest Generation (1/2)

- Merkle-Damgård construction [Mer89, Dam89] outputs a *fixed* sized digest:
  - inputs: $M = M_1 \| \cdots \| M_\mu$, and $h_0 = IV$;
  - output: h_μ, where

$$\forall i \in \{0, 1, \ldots, \mu - 1\} : \qquad h_{i+1} = \mathcal{F}(h_i, M_i) ;$$

- sponge construction [BDP+07] does allow for arbitrarily sized outputs, but the *output generation is inherently sequential*:



Parallel Digest Generation (2/2)

- Quest for *parallel digest generation*: evaluating several permutations simultaneously in modern CPUs is faster than evaluating them in sequence;
- In the case of PRFs: see the Farfalle construction proposed by Bertoni et al. [BDH+17] at ToSC 2017;
- In the case of hashing: "Mask Generation Functions" (MGFs);

$\longrightarrow$ we propose a *new MGF construction* – called Cascade-MGF – based on RTRUNC: it extends any hash function $\mathcal{H}$ into a parallel eXtendable Output Function (XOF) that generates arbitrarily sized outputs.

Parallel Digest Generation (2/2)

- Quest for *parallel digest generation*: evaluating several permutations simultaneously in modern CPUs is faster than evaluating them in sequence;
- In the case of PRFs: see the Farfalle construction proposed by Bertoni et al. [BDH+17] at ToSC 2017;
- In the case of hashing: "Mask Generation Functions" (MGFs);

$\longrightarrow$ we propose a *new MGF construction* – called Cascade-MGF – based on RTRUNC: it extends any hash function $\mathcal{H}$ into a parallel eXtendable Output Function (XOF) that generates arbitrarily sized outputs.

Cascade-MGF

- Let $\mathcal{H} : \{0,1\}^\star \to \{0,1\}^m$ be a (fixed output length) hash function, and let $\mathcal{P} : \{0,1\}^b \to \{0,1\}^b$ a public (random) permutation;
- Cascade-MGF:

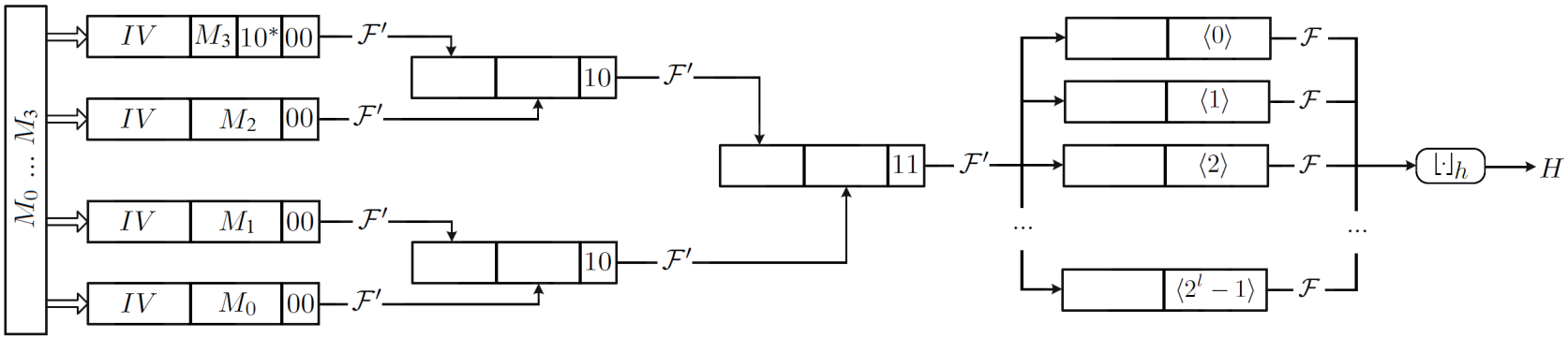
$$\mathrm{RTRUNC}(M, \langle 0 \rangle_l) \| \cdots \| \mathrm{RTRUNC}(M, \langle i \rangle_l) \| \cdots$$

  where

$$\mathrm{RTRUNC}(M, x) := \mathrm{left}_n \mathcal{P}(\mathcal{H}(M), x)\,;$$

- Cascade-MGF behaves as a (variable output length) random oracle if $\mathcal{H}$ behaves as a (fixed output length) random oracle.

Example: Cascade-MGF Construction via Tree Hash



Counter-MGF [KS98]

- Let $\mathcal{H} : \{0,1\}^\star \to \{0,1\}^m$ be a (fixed output length) hash function;
- (Variable output length) Counter-MGF:

$$\mathcal{H}(M\|\langle 0\rangle_l)\|\mathcal{H}(M\|\langle 1\rangle_l)\|\cdots\|\mathcal{H}(M\|\langle i\rangle_l)\|\cdots$$

- Counter-MGF behaves as a (variable output length) random oracle if $\mathcal{H}$ behaves as a (fixed output length) random oracle [SY12].

Chained-MGF [Nist09]

- Let $\mathcal{H} : \{0,1\}^\star \to \{0,1\}^m$ be a (fixed output length) hash function, and let $\mathcal{F} : \{0,1\}^{m+l} \to \{0,1\}^n$ a one way function;
- Chained-MGF:

$$\mathcal{F}(\mathcal{H}(M), \langle 0 \rangle_l) \| \mathcal{F}(\mathcal{H}(M), \langle 1 \rangle_l) \| \cdots \| \mathcal{F}(\mathcal{H}(M), \langle i \rangle_l) \| \cdots$$

- Chained-MGF behaves as a (variable output length) random oracle if $\mathcal{H}$ and $\mathcal{F}$ behave as (fixed output length) random oracles [SY12].

Cascade-MGF versus Chained-MGF with TRUNC (1/2)

Given

- $k =$ be the security level,
- $\mathcal{P} =$ public (random) permutation over $\{0,1\}^b$ for $b \geq 3 \cdot k$,
- $\mathcal{H}(M) \in \{0,1\}^{2 \cdot k}$,

then:

- $l = b - 2 \cdot k$ bits of counter for Cascade-MGF:

$$\mathsf{left}_{2 \cdot k} \mathcal{P}(\underbrace{\mathcal{H}(M)}_{2 \cdot k \text{ bits}} \| \langle i \rangle_l) \, ;$$

- $l = b - 3 \cdot k$ bits of counter for Chained-MGF instantiated by TRUNC:

$$\mathcal{F}(\mathcal{H}(M), \langle i \rangle_l) = \mathsf{left}_{2 \cdot k} \mathcal{P}(\underbrace{IV}_{k \text{ bits}} \| \underbrace{\mathcal{H}(M)}_{2 \cdot k \text{ bits}} \| \langle i \rangle_l) \, .$$

Cascade-MGF versus Chained-MGF with TRUNC (2/2)

Difference of k bits in the counter is crucial for permutations $\mathcal{P}$ over "small" fields:

- GIMLI [BKL+17] and XOODOO [DHV+18] are two permutations over $b = 384$ bits;
- assume $k = 128$ bits;
- Chained-MGF with TRUNC:

$$l = b - 3 \cdot k = 0 \text{ bits}$$

versus Cascade-MGF:

$$l = b - 2 \cdot k = 128 \text{ bits}$$

$\longrightarrow$ 1 digest block versus 2^{128} digest blocks!

Cascade-MGF versus Chained-MGF with SOP

Given $\mathcal{P}_0, \mathcal{P}_1 =$ two public (random) permutations over $\{0,1\}^b$ for $b \geq 2 \cdot k$:

- Chained-MGF instantiated by SOP:

$$\mathcal{F}(\mathcal{H}(M), \langle i \rangle_l) = \mathcal{P}_0(\mathcal{H}(M) \| \langle i \rangle_l) + \mathcal{P}_1(\mathcal{H}(M) \| \langle i \rangle_l)$$

  where $l = b - 2 \cdot k$ bits of counter;

- Cascade-MGF always improves over Chained-MGF with SOP:

$$\underbrace{b-k}_{\text{size digest block}} \geq \underbrace{b}_{\text{size digest block}} \cdot \underbrace{\frac{1}{2}}_{\text{2 permutation calls}}$$

  that is, $b \geq 2 \cdot k$, which is always satisfied.

Cascade-MGF versus Chained-MGF with SOP

Given $\mathcal{P}_0, \mathcal{P}_1 =$ two public (random) permutations over $\{0,1\}^b$ for $b \geq 2 \cdot k$:

- Chained-MGF instantiated by SOP:

$$\mathcal{F}(\mathcal{H}(M), \langle i \rangle_l) = \mathcal{P}_0(\mathcal{H}(M) \| \langle i \rangle_l) + \mathcal{P}_1(\mathcal{H}(M) \| \langle i \rangle_l)$$

  where $l = b - 2 \cdot k$ bits of counter;

- Cascade-MGF always improves over Chained-MGF with SOP:

$$\underbrace{b-k}_{\text{size digest block}} \geq \underbrace{b}_{\text{size digest block}} \cdot \underbrace{\frac{1}{2}}_{\text{2 permutation calls}}$$

  that is, $b \geq 2 \cdot k$, which is always satisfied.

Summary

Summary

- We proved that the truncated permutation construction instantiated by a randomized value is still indifferentiability from a random function (up to a certain limit);
- New Cascade-MGF construction for extending any hash function $\mathcal{H}$ into a parallel XOF;
- Concrete practical advantages of Cascade-MGF with respect to other MGF constructions.

Thanks for your attention!

Questions?

Comments?

Cascade-MGF versus Counter-MGF (with Tree Hash)

- Comparison between Counter-MGF and Cascade-MGF: **not** possible without being specific about the hash function $\mathcal{H}$;
- *If* $\mathcal{H}$ = *tree hash*, then the cost (= number function/permutation calls) for generating an output of arbitrary length:
  - independent of the size of the input message for Cascade-MGF;
  - depends on it for Counter-MGF (minimum if size of input = power of 2):
  - cost for Cascade-MGF $<$ cost for Counter-MGF.

Cascade-MGF versus Counter-MGF (with Tree Hash)

- Comparison between Counter-MGF and Cascade-MGF: **not** possible without being specific about the hash function $\mathcal{H}$;
- *If* $\mathcal{H}$ = *tree hash*, then the cost (= number function/permutation calls) for generating an output of arbitrary length:
  - independent of the size of the input message for Cascade-MGF;
  - depends on it for Counter-MGF (minimum if size of input = power of 2):
  - cost for Cascade-MGF $<$ cost for Counter-MGF.

References i

S. Bhattacharya and M. Nandi

Full Indifferentiable Security of the Xor of Two or More Random Permutations Using the χ^2 Method.

EUROCRYPT 2018

D. J. Bernstein, S. Kölbl, S. Lucks, P. M. C. Massolino, F. Mendel, K. Nawaz, T. Schneider, P. Schwabe, F.-X. Standaert, Y. Todo and B. Viguier

Gimli : A Cross-Platform Permutation.

CHES 2017

References ii


G. Bertoni, J. Daemen, S. Hoffert, M. Peeters, G. Van Assche and R. Van Keer
Farfalle: parallel permutation-based cryptography.
FSE/ToSC 2017


G. Bertoni, J. Daemen, M. Peeters and G. Van Assche
Sponge functions.
Ecrypt Hash Workshop 2007


W. Choi, B. Lee and J. Lee
Indifferentiability of Truncated Random Permutations.
ASIACRYPT 2019

References iii


J. Coron, Y. Dodis, C. Malinaud and P. Puniya
Merkle-Damgård Revisited: How to Construct a Hash Function.
CRYPTO 2005


J. Daemen, S. Hoffert, G. Van Assche and R. Van Keer
The design of Xoodoo and Xoofff.
FSE/ToSC 2018


I. Damgård
A Design Principle for Hash Functions.
CRYPTO 1989

References iv


Y. Dodis, L. Reyzin, R. L. Rivest and E. Shen
Indifferentiability of Permutation-Based Compression Functions and Tree-Based Modes of Operation, with Applications to MD6.
FSE 2009


B. Kaliski and J. Staddon
PKCS #1: RSA Cryptography Specifications Version 2.0.

RFC 1998


U. M. Mauer, R. Renner and C. Holestein
Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology.
TCC 2004

References v


R. C. Merkle

A Certified Digital Signature.

CRYPTO 1989


NIST

NIST SP800-108: Recommendation for Key Derivation Using Pseudorandom Functions.

2009


K. Suzuki and K. Yasuda

On the security of the cryptographic mask generation functions standardized by ANSI, IEEE, ISO/IEC, and NIST.

NTT Technical Review 2012