Design of Optimally Indifferentiable-Secure Double-Block-Length Hashing

Yusuke Naito

Mitsubishi Electric Corporation

ASK 2019

Dec. 15, 2019

Topic

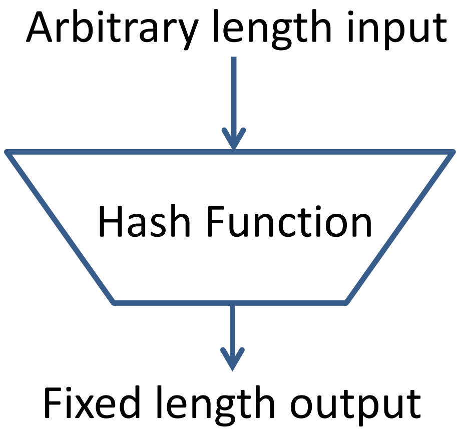
- Double-block-length (DBL) hash functions:
  - underlying primitive: block cipher.
  - security goal: optimal indifferentiable security.
- SAC 2011 DBL hash function [Nai11]:
  - 1st optimally indifferentiable secure,
  - block ciphers with 2*(block size) <= key size,
  - calls a block cipher twice as post-processing.
- Latincrypt 2019 DBL hash function [Nai19]:
  - optimally indifferentiable secure,
  - block ciphers with block size < key size,
  - not require the post-processing.

[Nai11] Yusuke Naito. Blockcipher-Based Double-Length Hash Functions for Pseudorandom Oracles. SAC 2011.

[Nai19] Yusuke Naito. Optimally Indifferentiable Double-Block-Length Hashing Without Post-processing and with Support for Longer Key Than Single Block. LATINCRYPT 2019.

Hash Function

- Interface:
  - Input: arbitrary length
  - Output: fixed length
    (e.g., 256 bits, 384 bits, 512 bits)



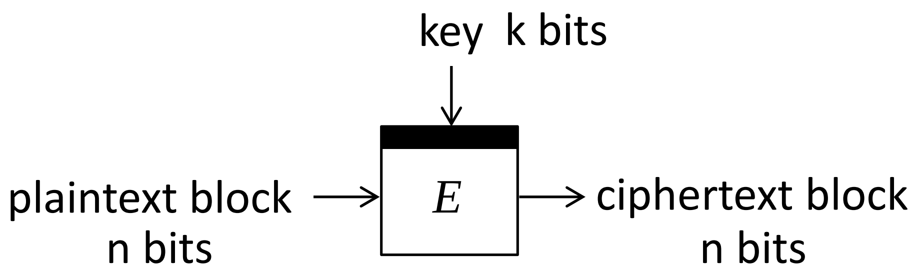
- Applications
  - Message authentication code
  - Pseudorandom function
  - Digital signature
  - Public key encryption
  - ....

Hash Function

- Basic Security Notions of Hash function H
  - Indifferentiability from a random oracle (or indiff. security): behave like a random oracle.
  - Collision Resistance: hard to find inputs M,M' s.t. H(M)=H(M').
  - Second Preimage Resistance: given M', hard to find input M s.t. H(M)=H(M').
  - Preimage Resistance: given z, hard to find input M s.t. H(M)=z.
- Design
  - Block cipher
  - permutation
  - compression function
  - ...

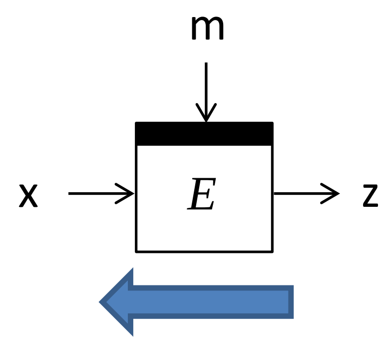
Advantage of block cipher-based design:
When implementing both encryption and hash function,
the underlying primitive can be shared,
thereby reducing the implementation size.

Block Cipher



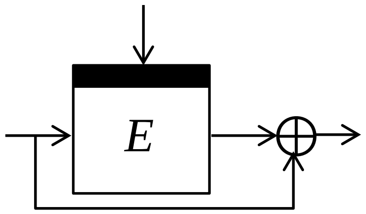
- Family of permutations indexed by keys.
- Mainly used to design symmetric-key crypt. algorithms e.g.,
  - authenticated encryption scheme,
  - message authentication code.
- A block cipher key is random and secret.

Block-Cipher-based Hash Design



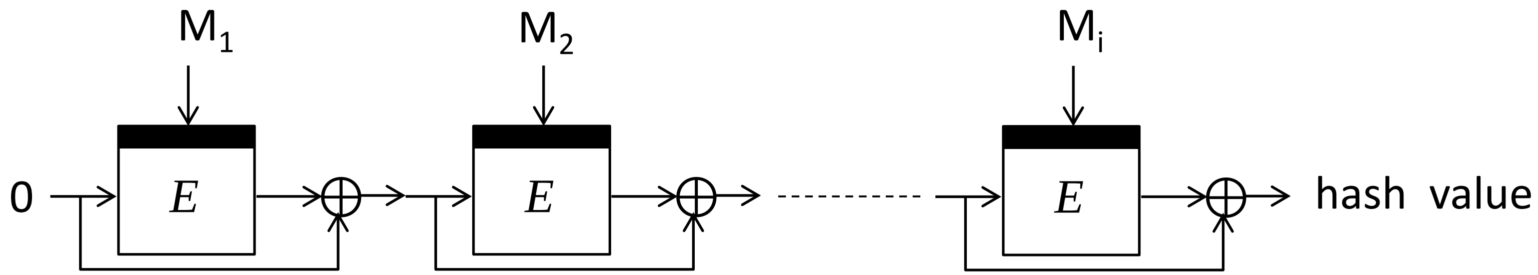
- The input and output become public.
- Using the decryption function of E, the preimage security is broken: given z, the input (x,m) can be found.

Block-Cipher-based Hash Design



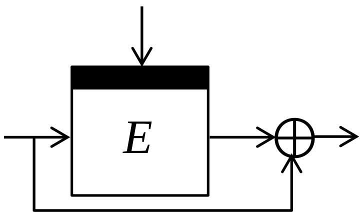
- The input is feed-forwarded to the output e.g., Davies-Meyer mode.
- Collision resistant up to $2^{n/2}$ query complexity (in the ideal (block) cipher model).
- The input length is extended by a domain extender that preserves the collision security such as Merkle-Damgard.

Block-Cipher-based Hash Design



- The input is feed-forwarded to the output
  e.g., Davies-Meyer mode.
- Collision resistant up to $2^{n/2}$ query complexity
  (in the ideal (block) cipher model).
- The input length is extended by a domain extender that preserves the collision security such as Merkle-Damgard.
- Merkle-Damgard with Davies-Meyer is collision resistant up to $2^{n/2}$ query complexity.

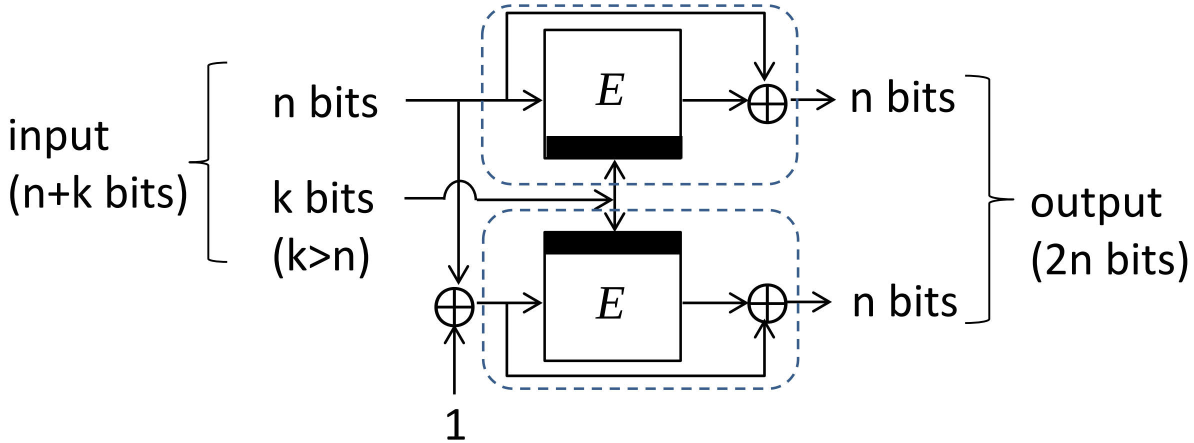
Problem of Single-Block-Length Scheme



- The output lengths of block ciphers are commonly ≤ 128, e.g., AES: n = 128.
- The output lengths of block ciphers are too short.
  - n = 128: 2^{64} security from the birthday attack.
  - Hash functions such as SHA-2, -3 are designed so that the output lengths are ≥ 224.

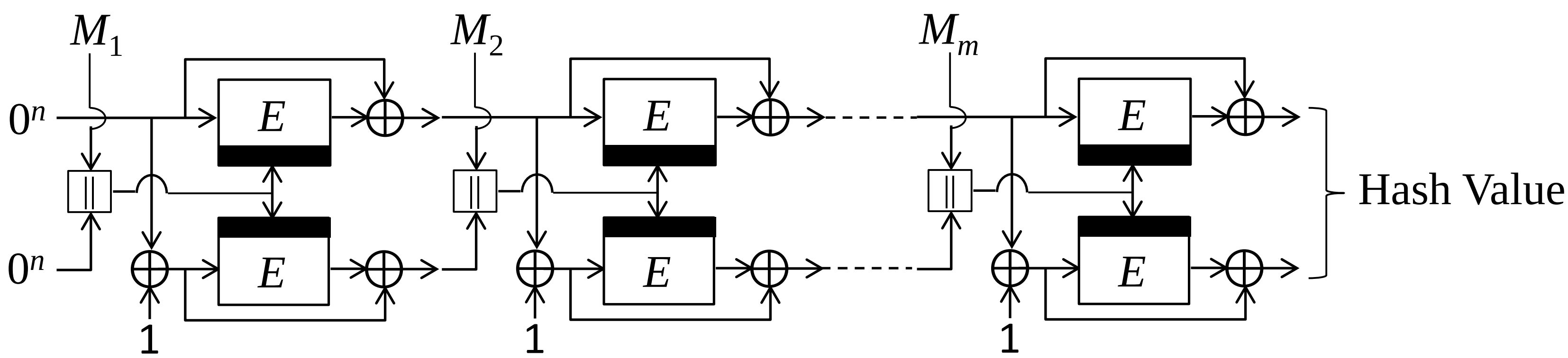
DBL Compression Function

Hirose's scheme



- The output length is extended by calling a block cipher twice.
- Hirose's scheme, Tandem-DM, Abreast-DM, etc.
- Collision resistant up to $O(2^n)$ query complexity (optimal collision security).

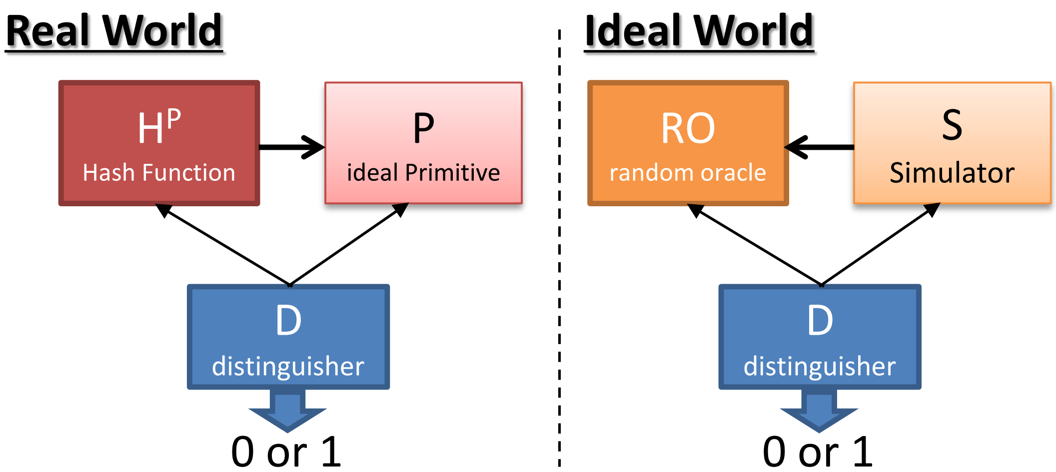
DBL Hash Function



- Optimal collision resistant DBL hash function (security up to $O(2^n)$ query complexity):
  - Optimal collision resistant DBL compression function + Merkle-Damgard.
- Indifferentiable secure DBL hash function:
  - Collision resistance $\nrightarrow$ Indifferentiability e.g., Merkle-Damgard.
  - We need a new DBL construction to achieve indifferentiable security.

Indifferentiability from Random Oracle

- Stronger security notion than collision, (second) preimage security.
- Indiff. hash function behaves as a random oracle.
- Indistinguishability between real and ideal worlds.



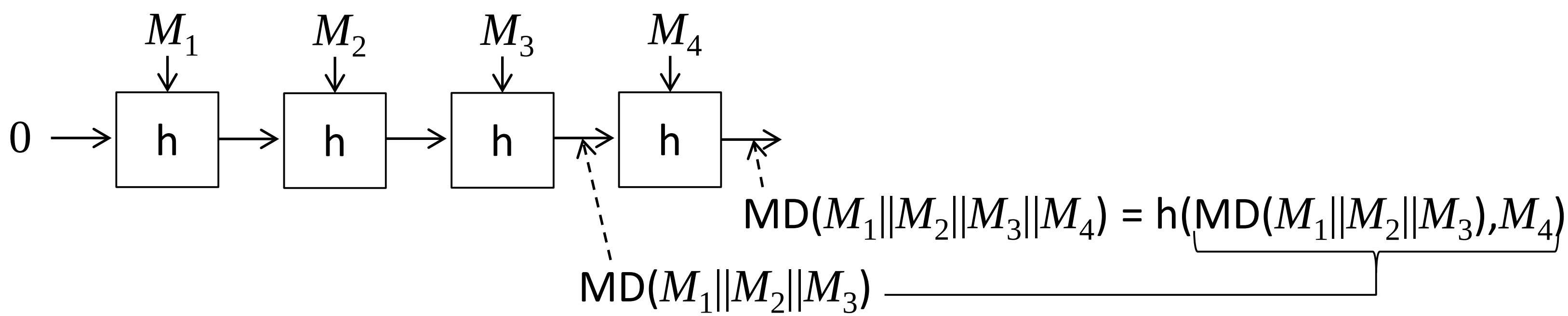
- H^P is indiff. from a random oracle or indiff. secure if
  - ∃ S s.t. no D can distinguish between real and ideal worlds.
- Optimally indifferentiable secure DBL hash function: security up to $O(2^n)$ query complexity (block cipher calls or message blocks).

Collision resistance$\nrightarrow$Indifferentiability

- Merkle-Damgard is NOT indiff. secure due to the length extension attack.

<u>Merkle-Damgard</u>

- Iterated function.
- There is a relation between MD($M||M^*$) and MD(M).

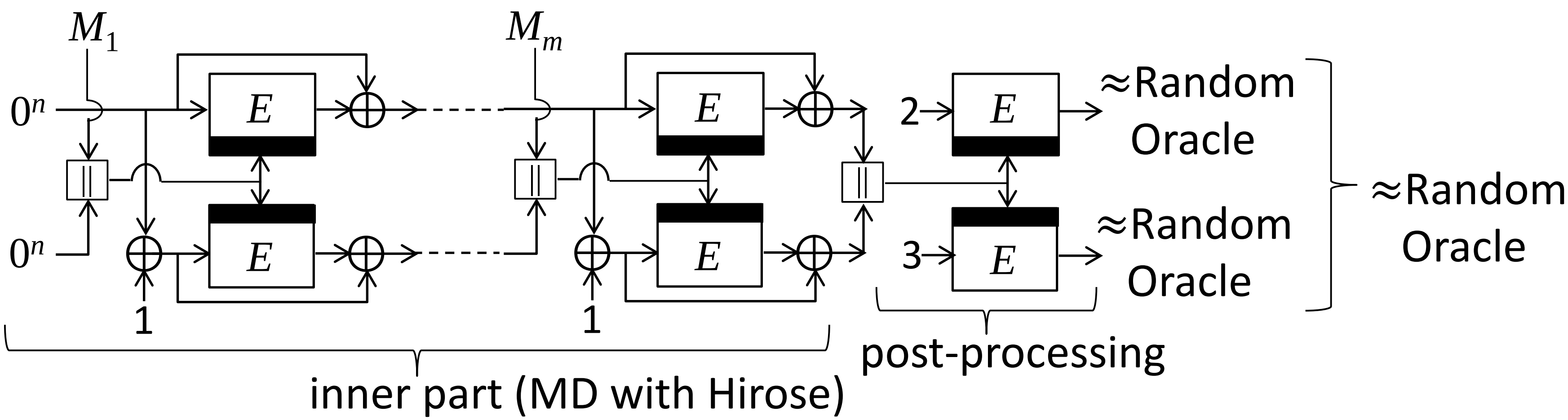


<u>Random Oracle</u>

- Monolithic function.
- There is no relation between RO($M||M^*$) and RO(M).

- In order to avoid the length extension attack, a hash value should not become an internal state.

SAC 2011 DBL Hash Function [Nai11]

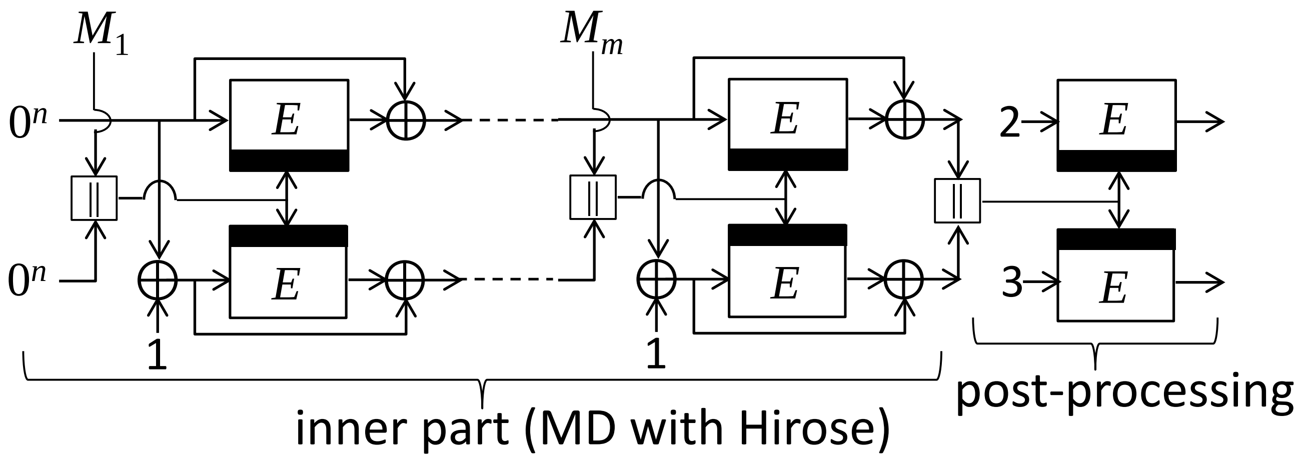


- Merkle-Damgard with Hirose + post-processing.
- Optimal indiff. secure (security up to $O(2^n)$ query complexity).
  - Post-processing can be seen as a random oracle.
  - ∀ hash query: the output of the inner part is fresh (or preimage-aware)
    ⇒ the hash function becomes
    a random oracle (indifferentiable from a random oracle).
  - Preimage aware: collision security + preimage-like security.

security up to $O(2^n)$ query complexity

Indiff. Secure DBL Hash Functions

- SAC2011 DBL hash function [Nai11]



  - Optimal Indiff. security ($O(2^n)$ security).
  - Drawbacks:
    - Require the post-processing (2 additional block cipher calls).
    - Require a block cipher with $k \geq 2n$,
      i.e., not support block ciphers with $k<2n$ e.g., AES-128, -192.
- PBGV-based by Gong et al. (Des. Codes Crypt.):
  - Support block ciphers with short keys $k>n$ (AES-192, -256),
  - Not achieve optimal indiff. security ($O(2^{n/2})$ security).

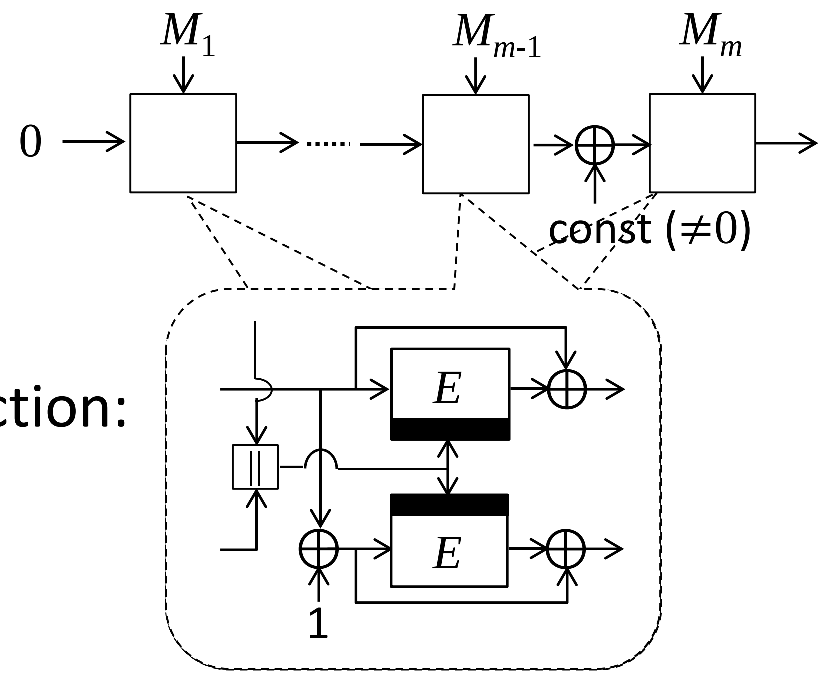
Open Problem for Indiff. DBL Hash

	Optimal Indiff. Security (2^n security)	Without Post-Processing	Support for Key Length k<2n
SAC 2011 [Nai11]	Yes	No	No
PBGV	No	Yes	Yes
?	Yes	Yes	Yes

- Open Problem:
  Design a DBL hash function
  - with optimal indiff. security (security up to $O(2^n)$ query complexity),
  - without post-processing, and
  - with support for block ciphers with k<2n.
- Latincrypt 2019 DBL hash function:
  - satisfies all these requirements.

Latincrypt 2019 DBL Hash Function [Nai19]

- MDPH, a combination of
  - the MDP domain extender:
  - Hirose's DBL compression function:

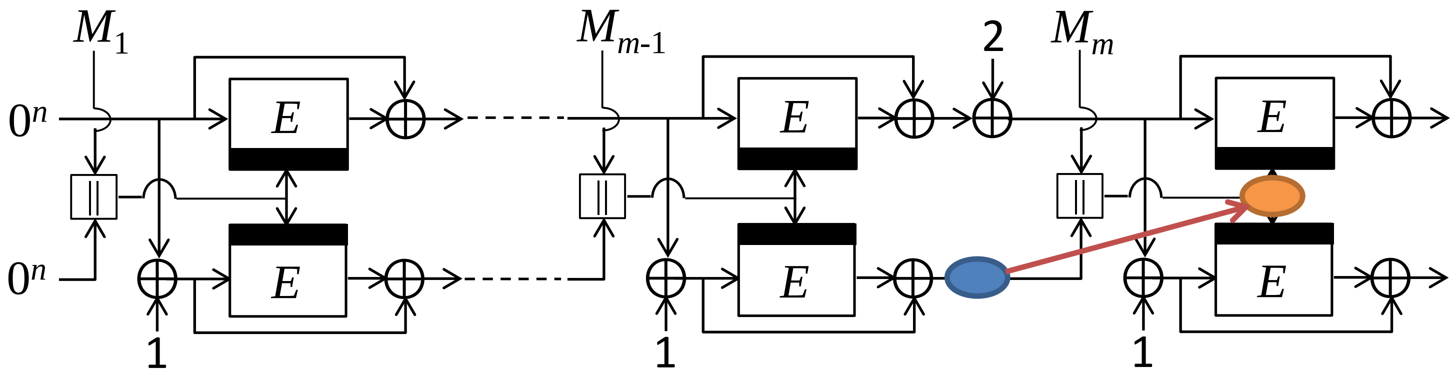


- We need to carefully combine them to have optimal indiff. security due to the difference between a random oracle and an ideal cipher.
  - Random oracle: outputs are random.
  - Ideal cipher: outputs are distinct for the same key.

The difference might offer an attack with $O(2^{n/2})$ query complexity (if all key elements are the same).

Latincrypt 2019 DBL Hash Function [Nai19]

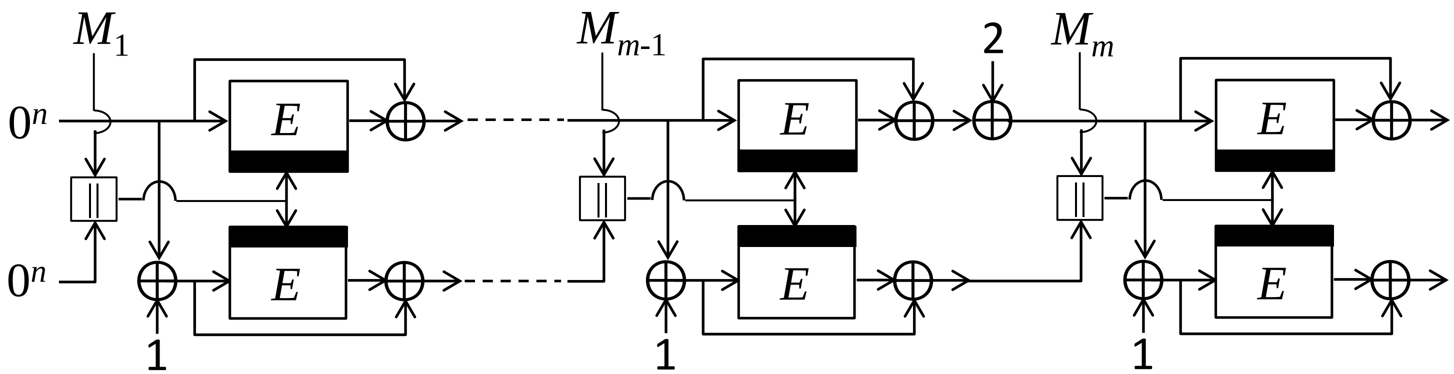
- We consider the following combination.



- Important Point to achieve optimal indiff. security.
  - The output becomes the part of the key element .
  - The multi-collision technique on : # the same key can be small.
    - If #multi-collision on $\leq \mu$, then # the same key on $\leq \mu$.
    - The attack complexity using the difference: $O(2^n/\mu)$.
    - When μ=n, the multi-collision probability is balanced with the attack probability.

MDPH is indiff. secure up to $O(2^n/n)$ query complexity: (nearly) optimal.

Summary of MDPH



- Achieve (nearly) optimal indiff. security ($O(2^n/n)$ security).
- Not require the post-processing.
- Support block ciphers with k > n (e.g., AES-192, AES-256).

Conclusion

	Optimal Indiff. Security (2^n security)	Without Post-Processing	Key Size
SAC2011 [Nai11]	Yes	No	$2n \leqq k$
Latincrypt2019 [Nai19]	Yes (nearly optimal)	Yes	$n<k$

- SAC 2011 DBL hash function:
  - 1st optimal indifferentiable-secure scheme.
- Latincrypt 2019 DBL hash function: MDPH
  - (nearly) optimal indiff. security.
  - w/o post-processing; support short key block ciphers (k>n).
- Open problem:
  - Design an optimally indiff. secure DBL hash function supporting block ciphers with k=n (such as AES-128).

Thank you for your attention!