Radboud University

Block-Cipher-Based Tree Hashing

Aldo Gunsing

Crypto 2022

Hash Functions

- A hash function $H : \{0,1\}^* \to \{0,1\}^n$ maps an arbitrarily length input to a fixed length output

Hash Functions

- A hash function $H : \{0,1\}^* \to \{0,1\}^n$ maps an arbitrarily length input to a fixed length output
- Multiple security notions for hash functions, e.g. collision resistance: it should be difficult to find $x \neq x'$ with $H(x) = H(x')$

Hash Functions

- A hash function $H : \{0,1\}^* \to \{0,1\}^n$ maps an arbitrarily length input to a fixed length output
- Multiple security notions for hash functions, e.g. collision resistance: it should be difficult to find $x \neq x'$ with $H(x) = H(x')$
- Stronger notion states that every output should be uniformly random and independent

Hash Functions

- A hash function $H : \{0,1\}^* \to \{0,1\}^n$ maps an arbitrarily length input to a fixed length output
- Multiple security notions for hash functions, e.g. collision resistance: it should be difficult to find $x \neq x'$ with $H(x) = H(x')$
- Stronger notion states that every output should be uniformly random and independent
- Formalized as indifferentiability by Maurer et al. [MRH04] and specified for hashing by Coron et al. [CDMP05]

MACs

- A MAC $S : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^n$ is a keyed function that maps an arbitrary length input to a fixed length output

MACs

- A MAC $S : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$ is a keyed function that maps an arbitrary length input to a fixed length output
- Used to check integrity of a message

MACs

- A MAC $S : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$ is a keyed function that maps an arbitrary length input to a fixed length output
- Used to check integrity of a message
- Two properties important for this presentation:

MACs

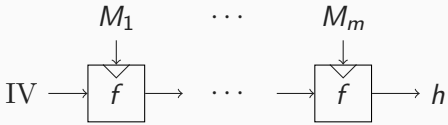
- A MAC $S : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$ is a keyed function that maps an arbitrary length input to a fixed length output
- Used to check integrity of a message
- Two properties important for this presentation:
  - Given the tag $h_1 = S(K, M_1)$ for an unknown key K but known message M_1 it should not be possible to forge (M_2, h_2) such that $h_2 = S(K, M_2)$

MACs

- A MAC $S : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$ is a keyed function that maps an arbitrary length input to a fixed length output
- Used to check integrity of a message
- Two properties important for this presentation:
  - Given the tag $h_1 = S(K, M_1)$ for an unknown key K but known message M_1 it should not be possible to forge (M_2, h_2) such that $h_2 = S(K, M_2)$
  - For an indifferentiable hash function H, the MAC $S(K, M) = H(K \parallel M)$ is secure with a security level of $\min(k, n/2)$

Length Extension

- Merkle-Damgård hashes based on a fixed-sized compression function f

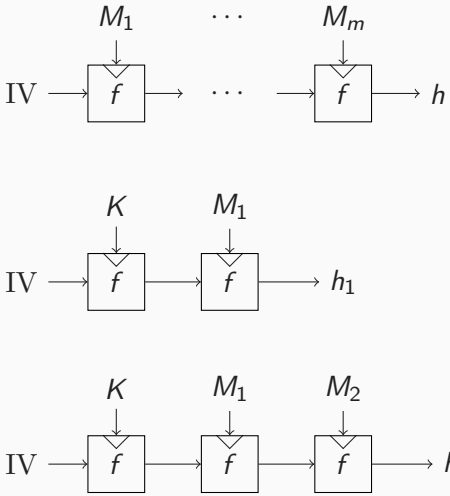


Length Extension

- Merkle-Damgård hashes based on a fixed-sized compression function f
- Susceptible to the length extension attack

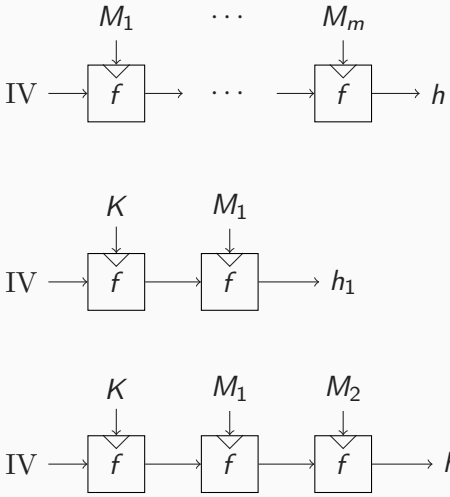
Length Extension

- Merkle-Damgård hashes based on a fixed-sized compression function f
- Susceptible to the length extension attack
- Suppose we use Merkle-Damgård in the MAC $S(K, M) = H(K \parallel M)$



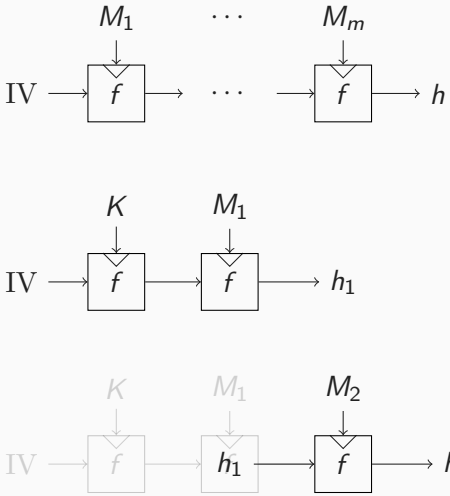
Length Extension

- Merkle-Damgård hashes based on a fixed-sized compression function f
- Susceptible to the length extension attack
- Suppose we use Merkle-Damgård in the MAC $S(K, M) = H(K \parallel M)$
- Given $h_1 = H(K \parallel M_1)$ it is possible to compute $h_2 = H(K \parallel M_1 \parallel M_2)$ for any M_2, without knowing K



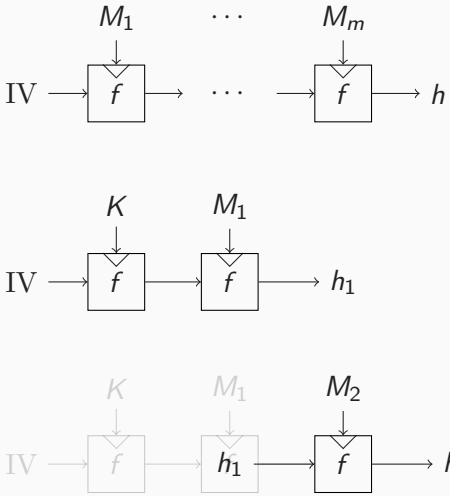
Length Extension

- Merkle-Damgård hashes based on a fixed-sized compression function f
- Susceptible to the length extension attack
- Suppose we use Merkle-Damgård in the MAC $S(K, M) = H(K \parallel M)$
- Given $h_1 = H(K \parallel M_1)$ it is possible to compute $h_2 = H(K \parallel M_1 \parallel M_2)$ for any M_2, without knowing K



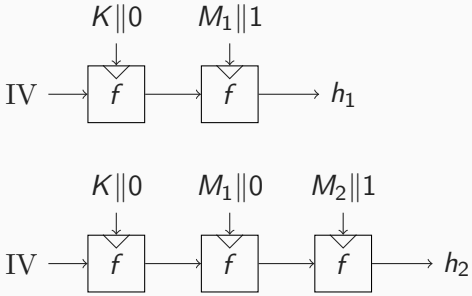
Length Extension

- Merkle-Damgård hashes based on a fixed-sized compression function f
- Susceptible to the length extension attack
- Suppose we use Merkle-Damgård in the MAC $S(K, M) = H(K \parallel M)$
- Given $h_1 = H(K \parallel M_1)$ it is possible to compute $h_2 = H(K \parallel M_1 \parallel M_2)$ for any M_2, without knowing K
- This means that $H(K \parallel M)$ is not a secure MAC, hence H is not indifferentiable



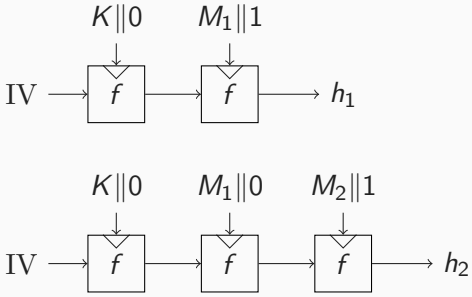
Domain Separation

- An easy way to fix the construction is to use domain separation on the final node



Domain Separation

- An easy way to fix the construction is to use domain separation on the final node
- This construction is shown indifferentiable using an ideal compression function f by Coron et al. [CDMP05]



Block-Cipher-Based Hashing

- Hashing modes are built on top of a smaller ideal compression function f

Block-Cipher-Based Hashing

- Hashing modes are built on top of a smaller ideal compression function f
- Such f cannot be built directly

Block-Cipher-Based Hashing

- Hashing modes are built on top of a smaller ideal compression function f
- Such f cannot be built directly
- A possible building block is a block cipher $\mathcal{E}$

Block-Cipher-Based Hashing

- Hashing modes are built on top of a smaller ideal compression function f
- Such f cannot be built directly
- A possible building block is a block cipher $\mathcal{E}$
- The popular Davies-Meyer construction $f(K, X) = \mathcal{E}(K, X) \oplus X$ is not indifferentiable from an ideal compression function

Block-Cipher-Based Hashing

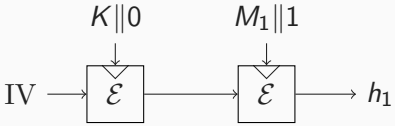
- Hashing modes are built on top of a smaller ideal compression function f
- Such f cannot be built directly
- A possible building block is a block cipher $\mathcal{E}$
- The popular Davies-Meyer construction $f(K, X) = \mathcal{E}(K, X) \oplus X$ is not indifferentiable from an ideal compression function
- Dedicated analyses required using underlying block cipher $\mathcal{E}$

Block-Cipher-Based Hashing

- Hashing modes are built on top of a smaller ideal compression function f
- Such f cannot be built directly
- A possible building block is a block cipher $\mathcal{E}$
- The popular Davies-Meyer construction $f(K, X) = \mathcal{E}(K, X) \oplus X$ is not indifferentiable from an ideal compression function
- Dedicated analyses required using underlying block cipher $\mathcal{E}$
- Finding sufficient conditions for such constructions, among others, was investigated by Daemen et al. [DMA18]

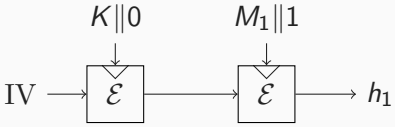
Insecure Mode

- However, the paper contains an error and proves some faulty constructions secure, as was pointed out by Samuel Neves



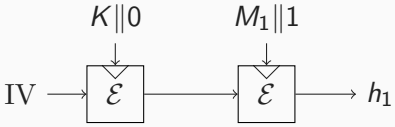
Insecure Mode

- However, the paper contains an error and proves some faulty constructions secure, as was pointed out by Samuel Neves
- Given $h_1 = H(K \parallel M_1)$ and M_1 it is possible to compute $h_2 = H(K \parallel M_2)$ for any M_2, without knowing K



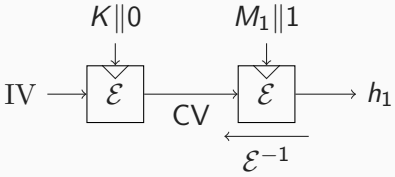
Insecure Mode

- However, the paper contains an error and proves some faulty constructions secure, as was pointed out by Samuel Neves
- Given $h_1 = H(K \parallel M_1)$ and M_1 it is possible to compute $h_2 = H(K \parallel M_2)$ for any M_2, without knowing K
- The intermediate chaining value CV is equal to $\mathcal{E}^{-1}(M_1 \| 1, h_1)$



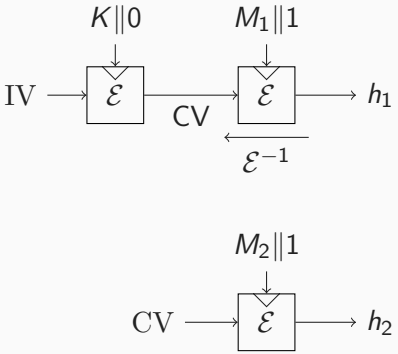
Insecure Mode

- However, the paper contains an error and proves some faulty constructions secure, as was pointed out by Samuel Neves
- Given $h_1 = H(K \parallel M_1)$ and M_1 it is possible to compute $h_2 = H(K \parallel M_2)$ for any M_2, without knowing K
- The intermediate chaining value CV is equal to $\mathcal{E}^{-1}(M_1 \| 1, h_1)$



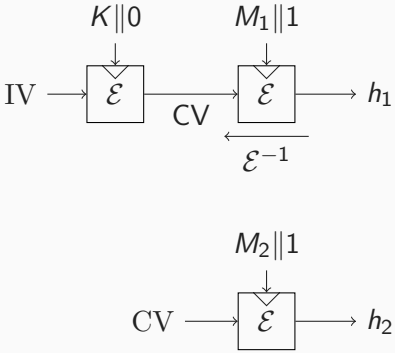
Insecure Mode

- However, the paper contains an error and proves some faulty constructions secure, as was pointed out by Samuel Neves
- Given $h_1 = H(K \parallel M_1)$ and M_1 it is possible to compute $h_2 = H(K \parallel M_2)$ for any M_2, without knowing K
- The intermediate chaining value CV is equal to $\mathcal{E}^{-1}(M_1\|1, h_1)$



Insecure Mode

- However, the paper contains an error and proves some faulty constructions secure, as was pointed out by Samuel Neves
- Given $h_1 = H(K \parallel M_1)$ and M_1 it is possible to compute $h_2 = H(K \parallel M_2)$ for any M_2, without knowing K
- The intermediate chaining value CV is equal to $\mathcal{E}^{-1}(M_1\|1, h_1)$
- Hence this construction is not secure, but it does satisfy the conditions



Cause of the Error

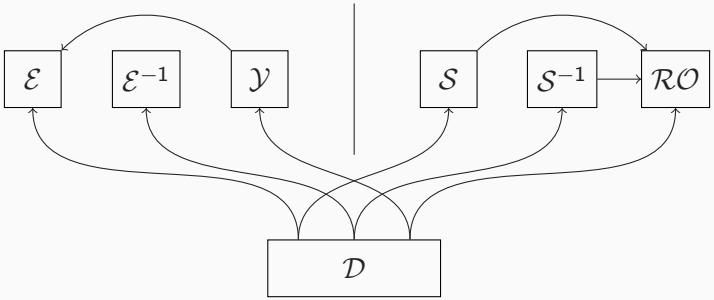
- A quick fix is achieved by additionally requiring sufficient truncation and is proven in an errata by Gunsing et al. [GDM20]

Cause of the Error

- A quick fix is achieved by additionally requiring sufficient truncation and is proven in an errata by Gunsing et al. [GDM20]
- A more thorough look reveals that the error is more fundamental about the ideal world in the indifferentiability setting

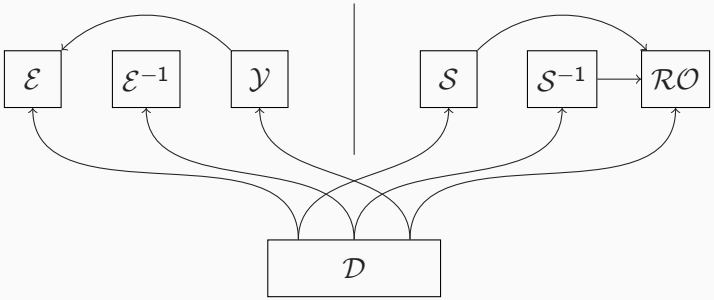
Indifferentiability

- Distinguisher $\mathcal{D}$ distinguishes between the real world and the ideal world



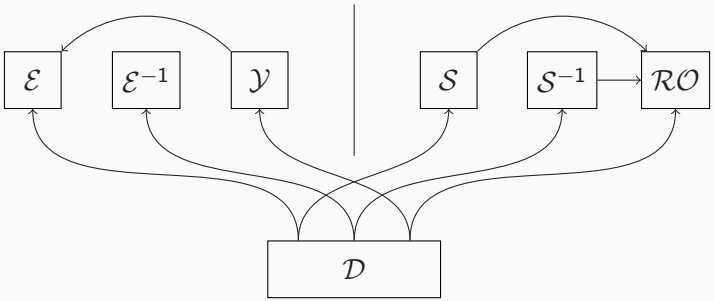
Indifferentiability

- Distinguisher $\mathcal{D}$ distinguishes between the real world and the ideal world
- Both primitive and construction queries



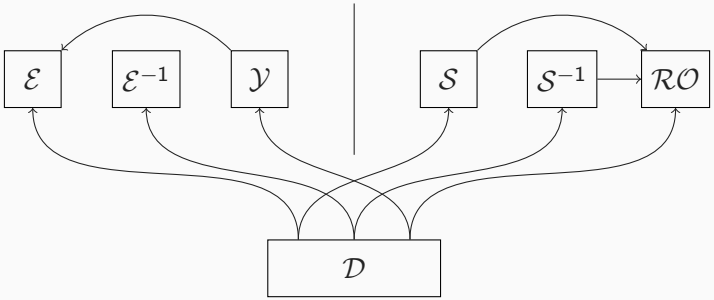
Indifferentiability

- Distinguisher $\mathcal{D}$ distinguishes between the real world and the ideal world
- Both primitive and construction queries
- Real world is a block cipher $\mathcal{E}$ (primitive) and a construction $\mathcal{Y}$ (construction)



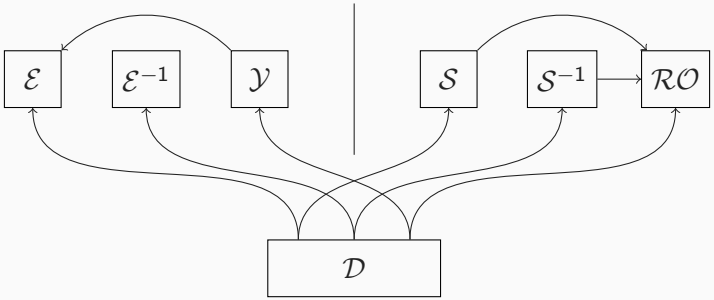
Indifferentiability

- Distinguisher $\mathcal{D}$ distinguishes between the real world and the ideal world
- Both primitive and construction queries
- Real world is a block cipher $\mathcal{E}$ (primitive) and a construction $\mathcal{Y}$ (construction)
- Ideal world is a simulator $\mathcal{S}$ (primitive) and a random oracle $\mathcal{RO}$ (construction)



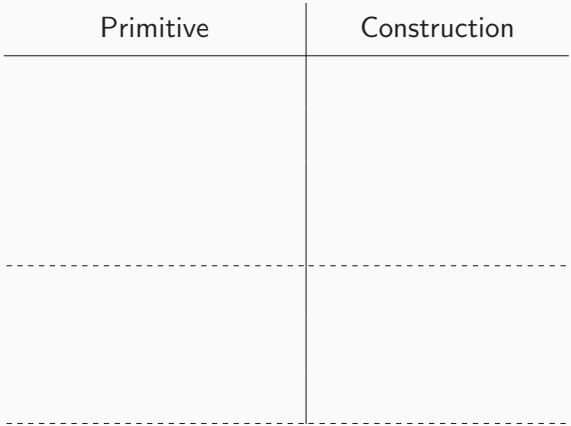
Indifferentiability

- Distinguisher $\mathcal{D}$ distinguishes between the real world and the ideal world
- Both primitive and construction queries
- Real world is a block cipher $\mathcal{E}$ (primitive) and a construction $\mathcal{Y}$ (construction)
- Ideal world is a simulator $\mathcal{S}$ (primitive) and a random oracle $\mathcal{RO}$ (construction)
- Both forward and backward direction for the primitive queries



Flawed Reasoning

- Modify the distinguisher $\mathcal{D}$ to an equivalent one $\mathcal{D}'$: ✓



Flawed Reasoning

- Modify the distinguisher $\mathcal{D}$ to an equivalent one $\mathcal{D}'$: ✓
  - Interact like $\mathcal{D}$

Primitive	Construction
	$\mathcal{RO}(K \parallel M_1) = h_1$
$\mathcal{S}^{-1}(M_1 \| 1, h_1) = \mathrm{CV}'$	
$\mathcal{S}(M_2 \| 1, \mathrm{CV}') = h_2'$	
	$\mathcal{RO}(K \parallel M_2) = h_2$

Flawed Reasoning

- Modify the distinguisher $\mathcal{D}$ to an equivalent one $\mathcal{D}'$: ✓
  - Interact like $\mathcal{D}$
  - Add verification queries for all construction queries

Primitive	Construction
	$\mathcal{RO}(K \parallel M_1) = h_1$
$\mathcal{S}^{-1}(M_1 \| 1, h_1) = \mathrm{CV}'$	
$\mathcal{S}(M_2 \| 1, \mathrm{CV}') = h_2'$	
	$\mathcal{RO}(K \parallel M_2) = h_2$
$\mathcal{S}(K \| 0, \mathrm{IV}) = \mathrm{CV}$	
$\mathcal{S}(M_1 \| 1, \mathrm{CV}) = h_1$	
$\mathcal{S}(M_2 \| 1, \mathrm{CV}) = h_2$	

Flawed Reasoning

- Modify the distinguisher $\mathcal{D}$ to an equivalent one $\mathcal{D}'$: ✓
  - Interact like $\mathcal{D}$
  - Add verification queries for all construction queries
  - Output the same decision as $\mathcal{D}$

Primitive	Construction
	$\mathcal{RO}(K \parallel M_1) = h_1$
$\mathcal{S}^{-1}(M_1 \| 1, h_1) = \mathrm{CV}'$	
$\mathcal{S}(M_2 \| 1, \mathrm{CV}') = h_2'$	
	$\mathcal{RO}(K \parallel M_2) = h_2$
$\mathcal{S}(K \| 0, \mathrm{IV}) = \mathrm{CV}$	
$\mathcal{S}(M_1 \| 1, \mathrm{CV}) = h_1$	
$\mathcal{S}(M_2 \| 1, \mathrm{CV}) = h_2$	
$h_2' \stackrel{?}{=} h_2$	

Flawed Reasoning

- Modify the distinguisher $\mathcal{D}$ to an equivalent one $\mathcal{D}'$: ✓
  - Interact like $\mathcal{D}$
  - Add verification queries for all construction queries
  - Output the same decision as $\mathcal{D}$
- Note that these queries contain duplicate information ✓

Primitive	Construction
	$\mathcal{RO}(K \parallel M_1) = h_1$
$\mathcal{S}^{-1}(M_1 \| 1, h_1) = \mathrm{CV}'$	
$\mathcal{S}(M_2 \| 1, \mathrm{CV}') = h_2'$	
	$\mathcal{RO}(K \parallel M_2) = h_2$
$\mathcal{S}(K \| 0, \mathrm{IV}) = \mathrm{CV}$	
$\mathcal{S}(M_1 \| 1, \mathrm{CV}) = h_1$	
$\mathcal{S}(M_2 \| 1, \mathrm{CV}) = h_2$	
$h_2' \stackrel{?}{=} h_2$	

Flawed Reasoning

- Modify the distinguisher $\mathcal{D}$ to an equivalent one $\mathcal{D}'$: ✓
  - Interact like $\mathcal{D}$
  - Add verification queries for all construction queries
  - Output the same decision as $\mathcal{D}$
- Note that these queries contain duplicate information ✓
- Ignore the construction queries, leaving only the primitive ones ✗

Primitive	Construction
$\mathcal{S}^{-1}(M_1 \| 1, h_1) = \mathrm{CV}'$	
$\mathcal{S}(M_2 \| 1, \mathrm{CV}') = h_2'$	
$\mathcal{S}(K \| 0, \mathrm{IV}) = \mathrm{CV}$	
$\mathcal{S}(M_1 \| 1, \mathrm{CV}) = h_1$	
$\mathcal{S}(M_2 \| 1, \mathrm{CV}) = h_2$	
$h_2' \stackrel{?}{=} h_2$	

Flawed Reasoning

- Modify the distinguisher $\mathcal{D}$ to an equivalent one $\mathcal{D}'$: ✓
  - Interact like $\mathcal{D}$
  - Add verification queries for all construction queries
  - Output the same decision as $\mathcal{D}$
- Note that these queries contain duplicate information ✓
- Ignore the construction queries, leaving only the primitive ones ✗
- Disregards that the construction queries can have influence on later queries

Primitive	Construction
$\mathcal{S}^{-1}(M_1 \| 1, h_1) = \mathrm{CV}'$	
$\mathcal{S}(M_2 \| 1, \mathrm{CV}') = h_2'$	
$\mathcal{S}(K \| 0, \mathrm{IV}) = \mathrm{CV}$	
$\mathcal{S}(M_1 \| 1, \mathrm{CV}) = h_1$	
$\mathcal{S}(M_2 \| 1, \mathrm{CV}) = h_2$	
$h_2' \stackrel{?}{=} h_2$	

Impact and Resolution

- Fundamental flaw ignoring a major part of the interaction between the oracles

Impact and Resolution

- Fundamental flaw ignoring a major part of the interaction between the oracles
- Flawed reasoning present in many other indifferentiability papers:

Impact and Resolution

- Fundamental flaw ignoring a major part of the interaction between the oracles
- Flawed reasoning present in many other indifferentiability papers:
  - [CN08, ABR21] on hashing

Impact and Resolution

- Fundamental flaw ignoring a major part of the interaction between the oracles
- Flawed reasoning present in many other indifferentiability papers:
  - [CN08, ABR21] on hashing
  - [MPN10, MP15, Lee17, BN18] on sum of permutations

Impact and Resolution

- Fundamental flaw ignoring a major part of the interaction between the oracles
- Flawed reasoning present in many other indifferentiability papers:
  - [CN08, ABR21] on hashing
  - [MPN10, MP15, Lee17, BN18] on sum of permutations
- Reasoning holds in, and only in, the sequential indifferentiability setting, where all primitive queries have to be made before the construction ones

Impact and Resolution

- Fundamental flaw ignoring a major part of the interaction between the oracles
- Flawed reasoning present in many other indifferentiability papers:
  - [CN08, ABR21] on hashing
  - [MPN10, MP15, Lee17, BN18] on sum of permutations
- Reasoning holds in, and only in, the sequential indifferentiability setting, where all primitive queries have to be made before the construction ones
- Results downgraded to this weaker setting

Fixing the Error for Hashing

- The errata requires sufficient truncation

Fixing the Error for Hashing

- The errata requires sufficient truncation
- However, this is unsatisfactory as not all modes can effectively apply this

Fixing the Error for Hashing

- The errata requires sufficient truncation
- However, this is unsatisfactory as not all modes can effectively apply this
- In this work we consider different options:

Fixing the Error for Hashing

- The errata requires sufficient truncation
- However, this is unsatisfactory as not all modes can effectively apply this
- In this work we consider different options:
  - Truncation with some variations

Fixing the Error for Hashing

- The errata requires sufficient truncation
- However, this is unsatisfactory as not all modes can effectively apply this
- In this work we consider different options:
  - Truncation with some variations
  - Enveloped finalization generalized from Enveloped Merkle-Damgård [BR06]

Fixing the Error for Hashing

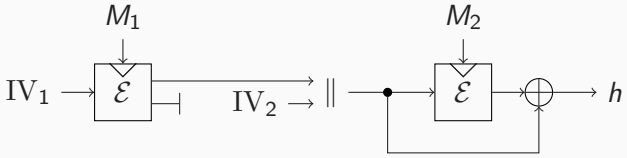
- The errata requires sufficient truncation
- However, this is unsatisfactory as not all modes can effectively apply this
- In this work we consider different options:
  - Truncation with some variations
  - Enveloped finalization generalized from Enveloped Merkle-Damgård [BR06]
  - Feed-forward finalization

Fixing the Error for Hashing

- The errata requires sufficient truncation
- However, this is unsatisfactory as not all modes can effectively apply this
- In this work we consider different options:
  - Truncation with some variations
  - Enveloped finalization generalized from Enveloped Merkle-Damgård [BR06]
  - Feed-forward finalization
- For this presentation, we focus on the feed-forward

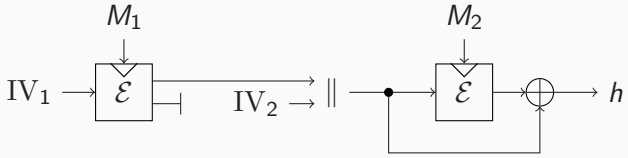
Feed-Forward Finalization

- A common construction is to use a feed-forward on top of the block cipher



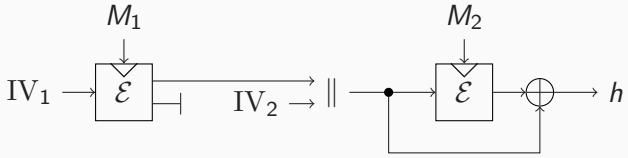
Feed-Forward Finalization

- A common construction is to use a feed-forward on top of the block cipher
- Only required for the final compression call



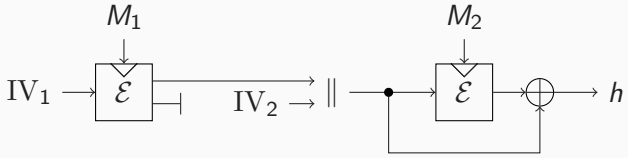
Feed-Forward Finalization

- A common construction is to use a feed-forward on top of the block cipher
- Only required for the final compression call
- IV_2 could have multiple possible values, like a counter



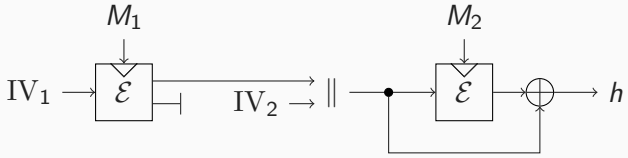
Feed-Forward Finalization

- A common construction is to use a feed-forward on top of the block cipher
- Only required for the final compression call
- IV_2 could have multiple possible values, like a counter
- The simulator has to loop over all possible IV_2 and query the random oracle



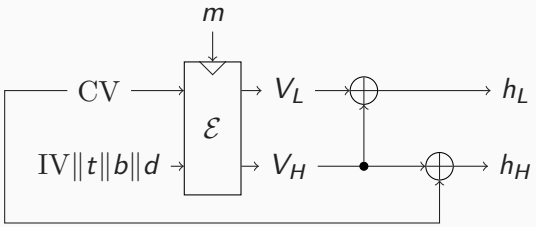
Feed-Forward Finalization

- A common construction is to use a feed-forward on top of the block cipher
- Only required for the final compression call
- IV_2 could have multiple possible values, like a counter
- The simulator has to loop over all possible IV_2 and query the random oracle
- Does not influence the advantage of $\mathcal{D}$, but does increase the query complexity of the simulator $\mathcal{S}$, which should be limited



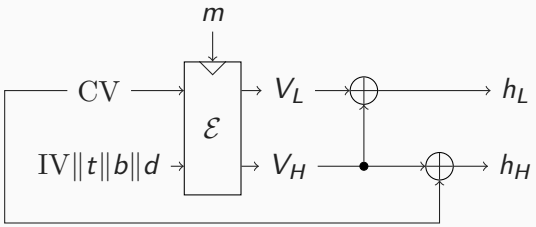
BLAKE3

- BLAKE3 is a recent block-cipher-based tree hashing mode



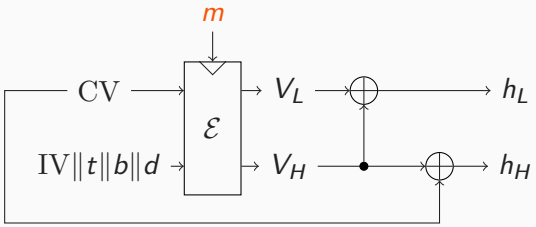
BLAKE3

- BLAKE3 is a recent block-cipher-based tree hashing mode
- We focus on its final compression call



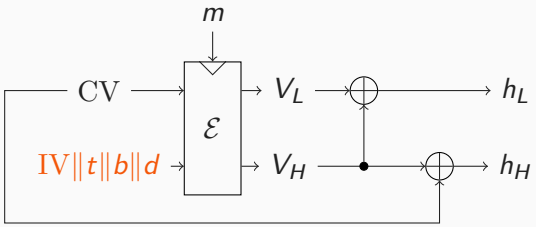
BLAKE3

- BLAKE3 is a recent block-cipher-based tree hashing mode
- We focus on its final compression call
- The key input is equal to the final message block m



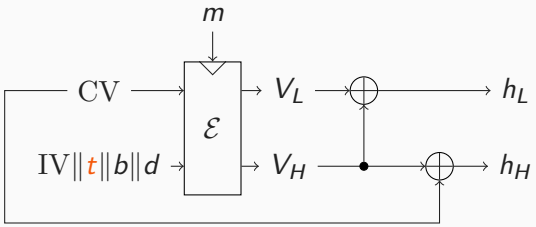
BLAKE3

- BLAKE3 is a recent block-cipher-based tree hashing mode
- We focus on its final compression call
- The key input is equal to the final message block m
- The non-CV data input contains an IV, a counter t, the number of bytes b in the message and some flags d



BLAKE3

- BLAKE3 is a recent block-cipher-based tree hashing mode
- We focus on its final compression call
- The key input is equal to the final message block m
- The non-CV data input contains an IV, a counter t, the number of bytes b in the message and some flags d
- The counter t is of particular interest



BLAKE3: Extendable Output

- BLAKE3 uses truncation for fixed output, which is covered previously and secure

BLAKE3: Extendable Output

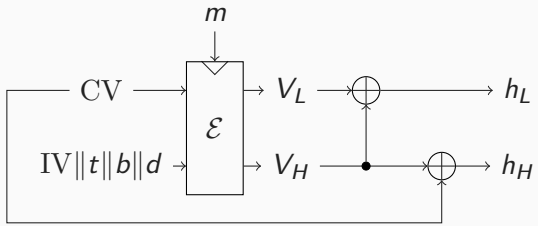
- BLAKE3 uses truncation for fixed output, which is covered previously and secure
- For extendable output it uses a counter

BLAKE3: Extendable Output

- BLAKE3 uses truncation for fixed output, which is covered previously and secure
- For extendable output it uses a counter
- Let $h_t = H(M, t)$ denote the output of the hash with counter t
- Then the full output is equal to $h_0 \parallel h_1 \parallel h_2 \parallel \cdots$

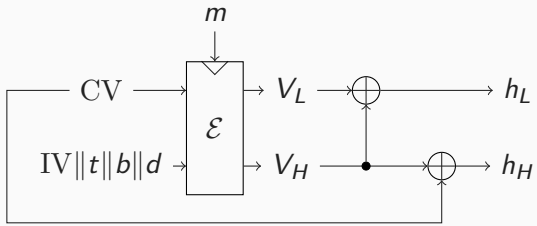
BLAKE3: Security Analysis

- BLAKE3 uses feed-forward finalization



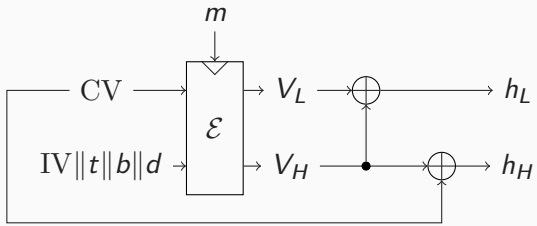
BLAKE3: Security Analysis

- BLAKE3 uses feed-forward finalization
- The non-CV input IV_2 should have limited possibilities, but there are many



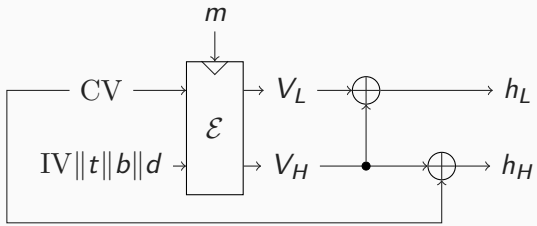
BLAKE3: Security Analysis

- BLAKE3 uses feed-forward finalization
- The non-CV input IV_2 should have limited possibilities, but there are many
- Leads to a peculiar property:



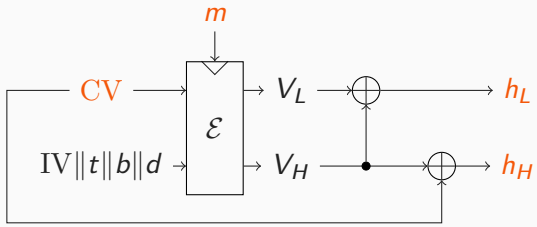
BLAKE3: Security Analysis

- BLAKE3 uses feed-forward finalization
- The non-CV input IV_2 should have limited possibilities, but there are many
- Leads to a peculiar property:
  - Assume a message M and its output $h = H(M, t)$ at some offset t are known, but t itself not



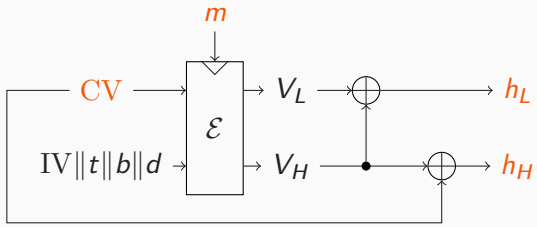
BLAKE3: Security Analysis

- BLAKE3 uses feed-forward finalization
- The non-CV input IV_2 should have limited possibilities, but there are many
- Leads to a peculiar property:
  - Assume a message M and its output $h = H(M, t)$ at some offset t are known, but t itself not
  - The offset t can be computed directly, while ideally brute-force is required



BLAKE3: Security Analysis

- BLAKE3 uses feed-forward finalization
- The non-CV input IV_2 should have limited possibilities, but there are many
- Leads to a peculiar property:
  - Assume a message M and its output $h = H(M, t)$ at some offset t are known, but t itself not
  - The offset t can be computed directly, while ideally brute-force is required



M, $H(M, t)$ known
then t known

BLAKE3: Insecure MAC

- As a consequence, the offset t cannot be secret

BLAKE3: Insecure MAC

- As a consequence, the offset t cannot be secret
- Define the following contrived MAC:

BLAKE3: Insecure MAC

- As a consequence, the offset t cannot be secret
- Define the following contrived MAC:
  - Let $K = K_1 \parallel K_2 \in \{0,1\}^{128}$ (with $|K_1| = 70$ and $|K_2| = 58$) be a key and $M \in \{0,1\}^*$ a message

BLAKE3: Insecure MAC

- As a consequence, the offset t cannot be secret
- Define the following contrived MAC:
  - Let $K = K_1 \parallel K_2 \in \{0,1\}^{128}$ (with $|K_1| = 70$ and $|K_2| = 58$) be a key and $M \in \{0,1\}^*$ a message
  - The tag is given by $H(M \parallel K_1, t = K_2)$, using the offset for part of the key

BLAKE3: Insecure MAC

- As a consequence, the offset t cannot be secret
- Define the following contrived MAC:
  - Let $K = K_1 \parallel K_2 \in \{0,1\}^{128}$ (with $|K_1| = 70$ and $|K_2| = 58$) be a key and $M \in \{0,1\}^*$ a message
  - The tag is given by $H(M \parallel K_1, t = K_2)$, using the offset for part of the key
- Allows for a key-recovery attack in 2^{70} queries, while 2^{128} is expected:

BLAKE3: Insecure MAC

- As a consequence, the offset t cannot be secret
- Define the following contrived MAC:
  - Let $K = K_1 \parallel K_2 \in \{0,1\}^{128}$ (with $|K_1| = 70$ and $|K_2| = 58$) be a key and $M \in \{0,1\}^*$ a message
  - The tag is given by $H(M \parallel K_1, t = K_2)$, using the offset for part of the key
- Allows for a key-recovery attack in 2^{70} queries, while 2^{128} is expected:
  - Let M and $h = H(M \parallel K_1, t = K_2)$ be given

BLAKE3: Insecure MAC

- As a consequence, the offset t cannot be secret
- Define the following contrived MAC:
  - Let $K = K_1 \parallel K_2 \in \{0,1\}^{128}$ (with $|K_1| = 70$ and $|K_2| = 58$) be a key and $M \in \{0,1\}^*$ a message
  - The tag is given by $H(M \parallel K_1, t = K_2)$, using the offset for part of the key
- Allows for a key-recovery attack in 2^{70} queries, while 2^{128} is expected:
  - Let M and $h = H(M \parallel K_1, t = K_2)$ be given

M, $H(M, t)$ known
then t known

BLAKE3: Insecure MAC

- As a consequence, the offset t cannot be secret
- Define the following contrived MAC:
  - Let $K = K_1 \parallel K_2 \in \{0,1\}^{128}$ (with $|K_1| = 70$ and $|K_2| = 58$) be a key and $M \in \{0,1\}^*$ a message
  - The tag is given by $H(M \parallel K_1, t = K_2)$, using the offset for part of the key
- Allows for a key-recovery attack in 2^{70} queries, while 2^{128} is expected:
  - Let M and $h = H(M \parallel K_1, t = K_2)$ be given
  - Let $K_1' \in \{0,1\}^{70}$ be a guess

M, $H(M,t)$ known

then t known

BLAKE3: Insecure MAC

- As a consequence, the offset t cannot be secret
- Define the following contrived MAC:
  - Let $K = K_1 \parallel K_2 \in \{0,1\}^{128}$ (with $|K_1| = 70$ and $|K_2| = 58$) be a key and $M \in \{0,1\}^*$ a message
  - The tag is given by $H(M \parallel K_1, t = K_2)$, using the offset for part of the key
- Allows for a key-recovery attack in 2^{70} queries, while 2^{128} is expected:
  - Let M and $h = H(M \parallel K_1, t = K_2)$ be given
  - Let $K_1' \in \{0,1\}^{70}$ be a guess
  - The offset, equal to the remaining key, $t = K_2' \in \{0,1\}^{58}$, can be computed directly

M, $H(M,t)$ known
then t known

Conclusion

- Many previous indifferentiability proofs contain a fundamental error

Conclusion

- Many previous indifferentiability proofs contain a fundamental error
- Results downgraded to the weaker sequential indifferentiability

Conclusion

- Many previous indifferentiability proofs contain a fundamental error
- Results downgraded to the weaker sequential indifferentiability
- For block-cipher-based tree hashing an unsatisfactory fix is to require truncation

Conclusion

- Many previous indifferentiability proofs contain a fundamental error
- Results downgraded to the weaker sequential indifferentiability
- For block-cipher-based tree hashing an unsatisfactory fix is to require truncation
- We proved other finalizations secure, including the common feed-forward

Conclusion

- Many previous indifferentiability proofs contain a fundamental error
- Results downgraded to the weaker sequential indifferentiability
- For block-cipher-based tree hashing an unsatisfactory fix is to require truncation
- We proved other finalizations secure, including the common feed-forward
- BLAKE3 can be proven indifferentiable, but requires a public offset for the extendable output

Conclusion

- Many previous indifferentiability proofs contain a fundamental error
- Results downgraded to the weaker sequential indifferentiability
- For block-cipher-based tree hashing an unsatisfactory fix is to require truncation
- We proved other finalizations secure, including the common feed-forward
- BLAKE3 can be proven indifferentiable, but requires a public offset for the extendable output

Thank you for your attention!

Elena Andreeva, Rishiraj Bhattacharyya, and Arnab Roy.
Compactness of Hashing Modes and Efficiency Beyond Merkle Tree.
In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 92–123. Springer, 2021.

Srimanta Bhattacharya and Mridul Nandi.
Full Indifferentiable Security of the Xor of Two or More Random Permutations Using the χ^2 Method.
In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 387–412. Springer, 2018.

Mihir Bellare and Thomas Ristenpart.
Multi-Property-Preserving Hash Domain Extension and the EMD Transform.
In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, volume 4284 of *Lecture Notes in Computer Science*, pages 299–314. Springer, 2006.

Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya.
Merkle-Damgård Revisited: How to Construct a Hash Function.
In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, 2005.

Donghoon Chang and Mridul Nandi.

Improved Indifferentiability Security Analysis of chopMD Hash Function.
In Kaisa Nyberg, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, volume 5086 of *Lecture Notes in Computer Science*, pages 429–443. Springer, 2008.


Joan Daemen, Bart Mennink, and Gilles Van Assche.
Sound Hashing Modes of Arbitrary Functions, Permutations, and Block Ciphers.
IACR Trans. Symmetric Cryptol., 2018(4):197–228, 2018.


Aldo Gunsing, Joan Daemen, and Bart Mennink.
Errata to Sound Hashing Modes of Arbitrary Functions, Permutations, and Block Ciphers.
IACR Trans. Symmetric Cryptol., 2020(3):362–366, 2020.


Jooyoung Lee.

Indifferentiability of the Sum of Random Permutations Toward Optimal Security.
IEEE Trans. Inf. Theory, 63(6):4050–4054, 2017.


Bart Mennink and Bart Preneel.
On the XOR of Multiple Random Permutations.
In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*, volume 9092 of *Lecture Notes in Computer Science*, pages 619–634. Springer, 2015.


Avradip Mandal, Jacques Patarin, and Valérie Nachef.
Indifferentiability beyond the Birthday Bound for the Xor of Two Public Random Permutations.
In Guang Gong and Kishan Chand Gupta, editors, *Progress in Cryptology - INDOCRYPT 2010 - 11th International Conference on Cryptology in India,*

Hyderabad, India, December 12-15, 2010. Proceedings, volume 6498 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2010.

Ueli M. Maurer, Renato Renner, and Clemens Holenstein.
Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology.
In Moni Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.