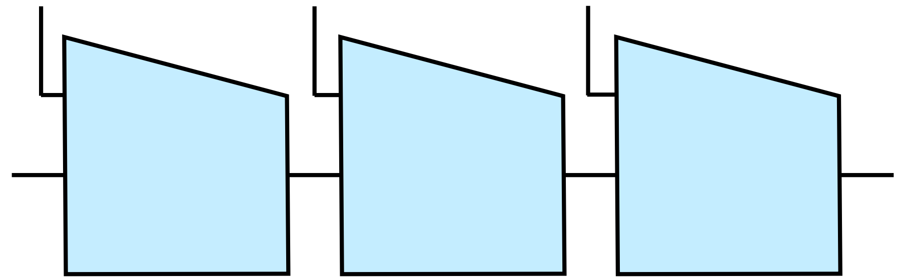
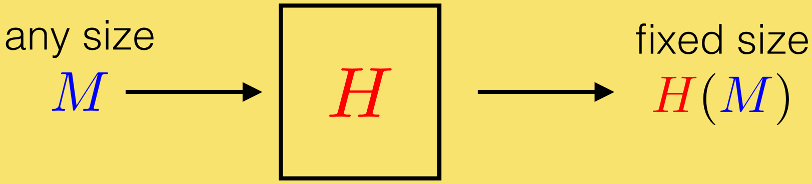


Better Than Advertised:
Improved Collision-Resistance Guarantees for MD-Based Hash Functions

Mihir Bellare   Joseph Jaeger   **Julia Len**

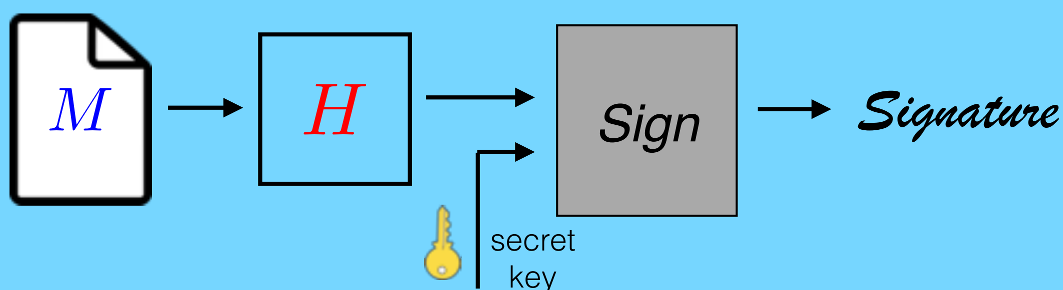
UC San Diego

Hash Functions



$$H : \{0,1\}^* \to \{0,1\}^n$$

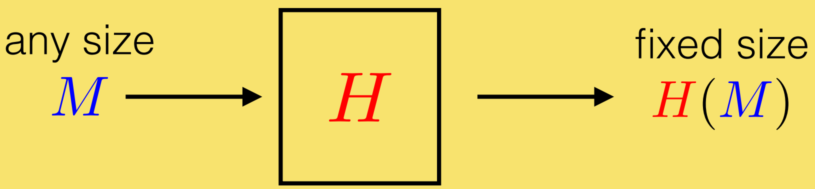
Central usage: Certificates



Main Security Goal: Collision resistance (CR)

Hard to find distinct messages with the same hash in time less than $2^{n/2}$, the time of a birthday attack.

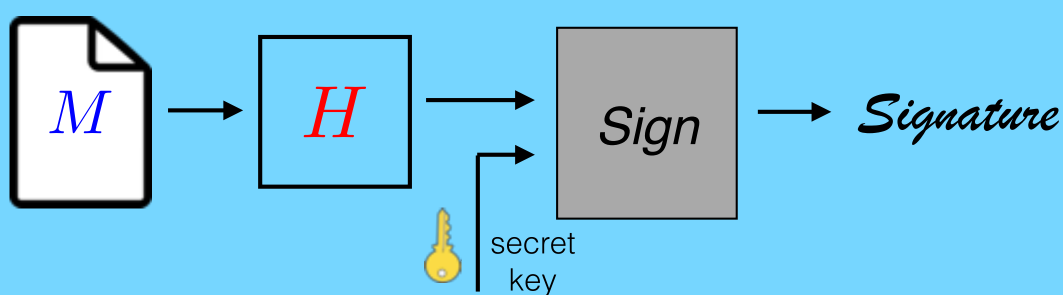
Hash Functions



$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

Generation	H	n
1st	MD4, MD5	128
2nd	SHA-1, SHA-256, SHA-512	160, 256, 512
3rd	SHA3-224, SHA3-256, SHA3-384, SHA3-512	224, 256, 384, 512

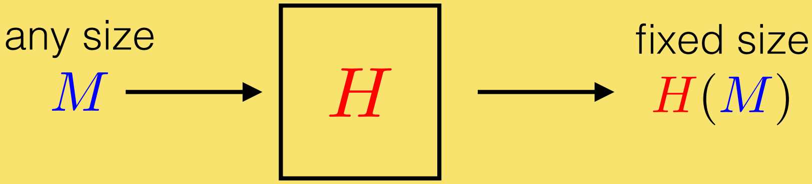
Central usage: Certificates



Main Security Goal: Collision resistance (CR)

Hard to find distinct messages with the same hash in time less than $2^{n/2}$, the time of a birthday attack.

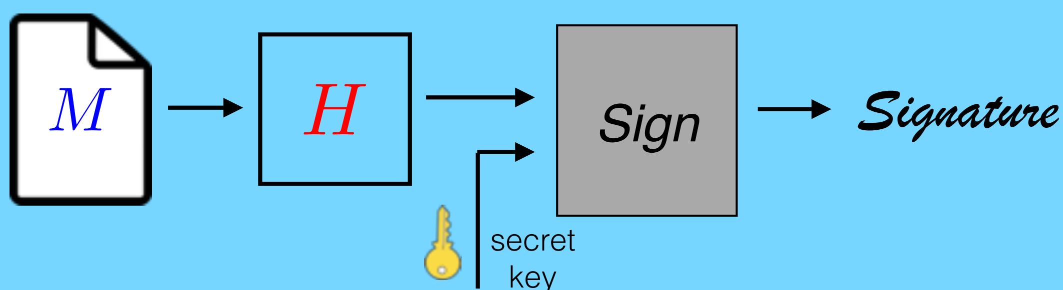
Hash Functions



$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

Generation	H	n
1st	~~MD4~~, MD5	128
2nd	SHA-1, SHA-256, SHA-512	160, 256, 512
3rd	SHA3-224, SHA3-256, SHA3-384, SHA3-512	224, 256, 384, 512

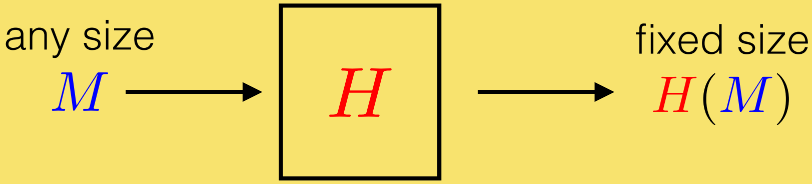
Central usage: Certificates



Main Security Goal: Collision resistance (CR)

Hard to find distinct messages with the same hash in time less than $2^{n/2}$, the time of a birthday attack.

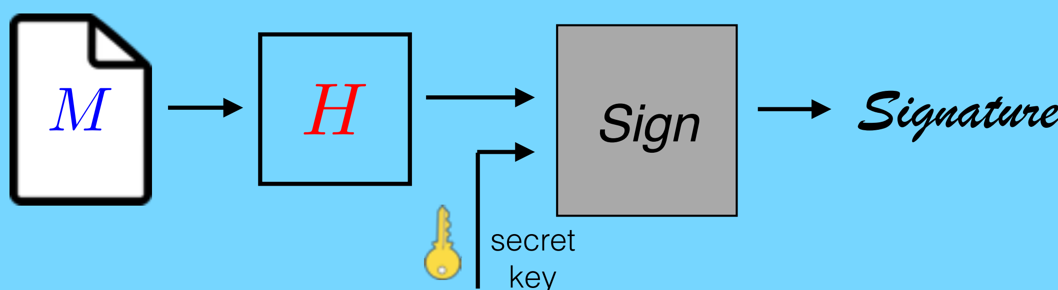
Hash Functions



$$H : \{0,1\}^* \to \{0,1\}^n$$

Generation	H	n
1st	~~MD4~~, ~~MD5~~	128
2nd	SHA-1, SHA-256, SHA-512	160, 256, 512
3rd	SHA3-224, SHA3-256, SHA3-384, SHA3-512	224, 256, 384, 512

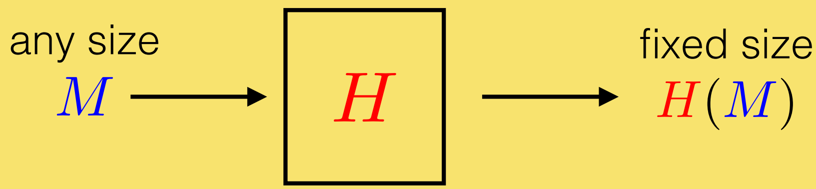
Central usage: Certificates



Main Security Goal: Collision resistance (CR)

Hard to find distinct messages with the same hash in time less than $2^{n/2}$, the time of a birthday attack.

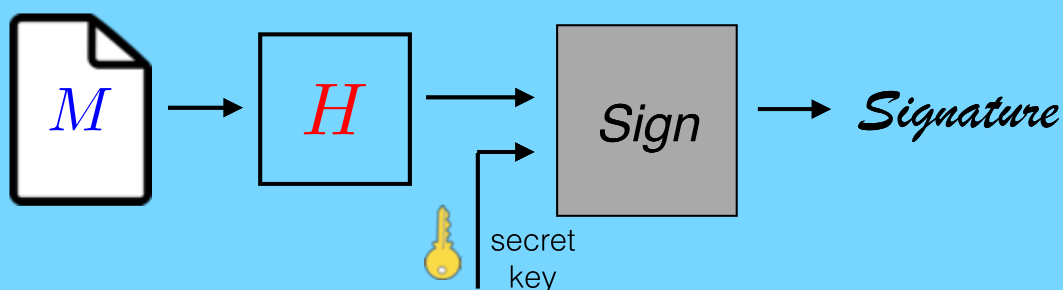
Hash Functions



$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

Generation	H	n
1st	~~MD4~~, ~~MD5~~	128
2nd	~~SHA-1~~, SHA-256, SHA-512	160, 256, 512
3rd	SHA3-224, SHA3-256, SHA3-384, SHA3-512	224, 256, 384, 512

Central usage: Certificates



Collisions in H lead to certificate forgery. SHA-1 collision leading to browsers no longer accepting SHA-1-based certificates.

Main Security Goal: Collision resistance (CR)

Hard to find distinct messages with the same hash in time less than $2^{n/2}$, the time of a birthday attack.

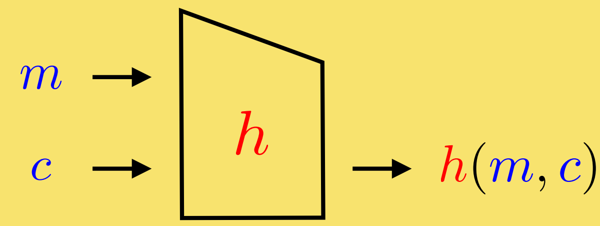
[SBKAM17]

https://shattered.io/

How hash functions are built

How hash functions are built

Step 1: Design a compression function h



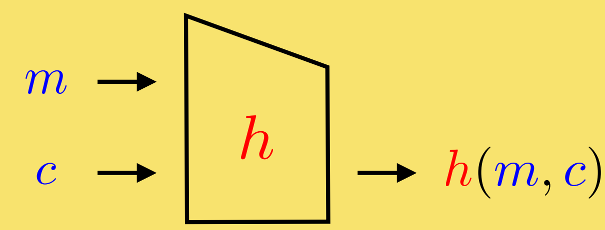
$$h : \{0,1\}^{h.ml + h.cl} \rightarrow \{0,1\}^{h.cl}$$

H	$h.ml$	$h.cl$
MD5	512	128
SHA-1	512	160
SHA-256	512	256
SHA-512	1024	512

How hash functions are built

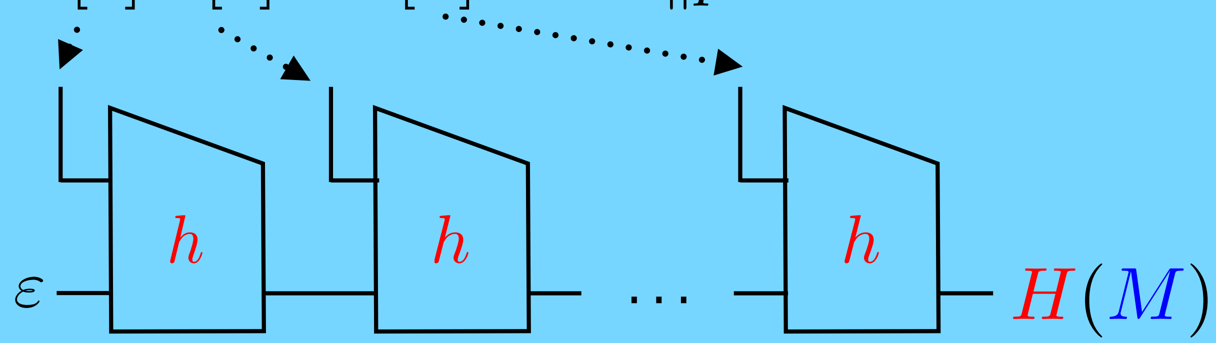
Step 1: Design a compression function h

Step 2: Convert h into a CR hash H function via the MD transform



$$h : \{0,1\}^{h.ml+h.cl} \to \{0,1\}^{h.cl}$$

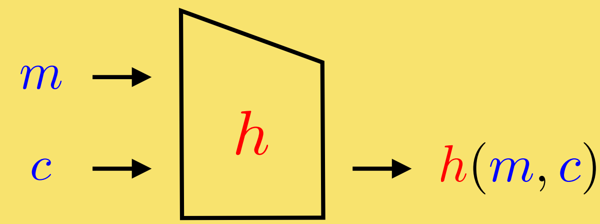
$$m[1]m[2]...m[n] \leftarrow M\|pad$$



How hash functions are built

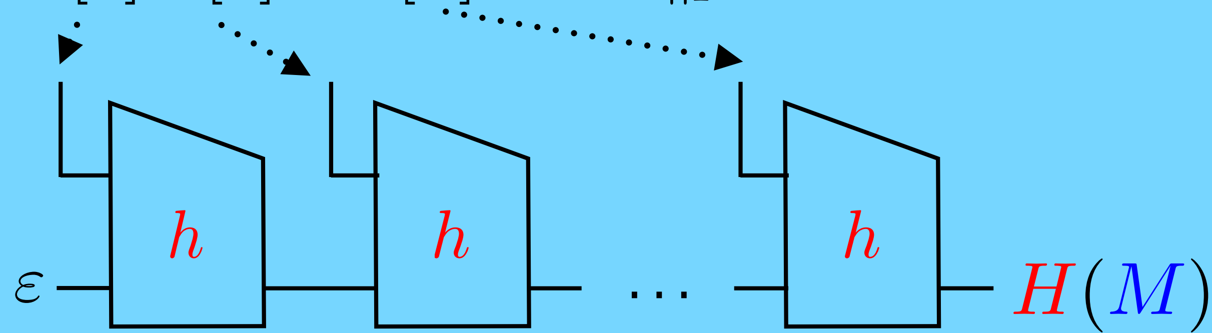
Step 1: Design a compression function h

Step 2: Convert h into a CR hash H function via the MD transform



$$h : \{0,1\}^{h.ml+h.cl} \rightarrow \{0,1\}^{h.cl}$$

$$m[1]m[2]...m[n] \leftarrow M\|pad$$




Merkle


Damgård

Classical Theorem: [Me,Da] h CR => H CR

Step 1: Design a compression function h

Step 2: Convert h into a CR hash function H via the MD transform

Classical Theorem: [Me,Da]
h CR => H CR

Problem: We haven't done so well in designing CR hash functions.

- Corollary of Classical Theorem: H not CR => h not CR
- So compression functions of MD5 and SHA-1 are NOT CR

Step 1: Design a compression function h

Step 2: Convert h into a CR hash function H via the MD transform

Classical Theorem: [Me,Da]
h CR => H CR

Problem: We haven't done so well in designing CR hash functions.

- Corollary of Classical Theorem: H not CR => h not CR
- So compression functions of MD5 and SHA-1 are NOT CR

Question: Can we weaken the assumption on h?

Desired Theorem:
h is X-secure => H CR

For some choice of X that is WEAKER than CR.

Step 1: Design a compression function h

Step 2: Convert h into a CR hash function H via the MD transform

Classical Theorem:[Me,Da]
h CR => H CR

Problem: We haven't done so well in designing CR hash functions.

- Corollary of Classical Theorem: H not CR => h not CR
- So compression functions of MD5 and SHA-1 are NOT CR

Question: Can we weaken the assumption on h ?

Desired Theorem:
h is X-secure => H CR

For some choice of X that is WEAKER than CR.

Our Answer: YES, X = CCR

Constrained **C**ollision-**R**esistance.
We will define this and show it is weaker than CR.

Step 1: Design a compression function h

Step 2: Convert h into a CR hash function H via the MD transform

Classical Theorem:[Me,Da]
h CR => H CR

Our Theorem 1:
h CCR => H CR

Our Theorem 2: There exist h that are CCR but not CR

Assumption-minimization paradigm of theoretical cryptography

But in a practical context

Step 1: Design a compression function h

Step 2: Convert h into a CR hash function H via the MD transform

Classical Theorem:[Me,Da]
h CR => H CR

Our Theorem 1:
h CCR => H CR

Our Theorem 2: There exist h that are CCR but not CR

Assumption-minimization paradigm of theoretical cryptography
But in a practical context

Potential Benefits: CCR may be easier to get right than CR

Step 1: Design a compression function h

Step 2: Convert h into a CR hash function H via the MD transform

Classical Theorem: [Me,Da]
h CR => H CR

Our Theorem 1:
h CCR => H CR

Our Theorem 2: There exist h that are CCR but not CR

Assumption-minimization paradigm of theoretical cryptography

But in a practical context

Potential Benefits: CCR may be easier to get right than CR

Better than Advertised: The MD transform does more than previously understood: It can promote weaker-than-CR compression functions into CR hash functions.

Step 1: Design a compression function h

Step 2: Convert h into a CR hash function H via the MD transform

Classical Theorem:[Me,Da]
h CR => H CR

Our Theorem 1:
h CCR => H CR

Our Theorem 2: There exist h that are CCR but not CR

Assumption-minimization paradigm of theoretical cryptography

But in a practical context

Potential Benefits: CCR may be easier to get right than CR

Better than Advertised: The MD transform does more than previously understood: It can promote weaker-than-CR compression functions into CR hash functions.

Security amplification: The MD transform "amplifies" or "boosts" security by turning a weaker-than-CR compression functions into a CR hash function.

Contributions

Our Theorem 1: h CCR => H CR

Our Theorem 2: There exist h that are CCR but not CR

These results are obtained via a **general framework**

— Parameterized version of MD: $H = \mathbf{MD}[h, \mathsf{Split}, \mathsf{S}]$

— RS Security framework: Yields both old and new definitions of security for h

Contributions

Our Theorem 1:
h CCR => H CR

Our Theorem 2: There exist h that are CCR but not CR

These results are obtained via a **general framework**

— Parameterized version of MD: $H = \mathbf{MD}[h, \mathsf{Split}, \mathsf{S}]$

— RS Security framework: Yields both old and new definitions of security for h

The framework

— Allows us to formalize and prove folklore results

— Is used to prove some new results

— Is pedagogically valuable in unifying results in the area

Contributions

Our Theorem 1: h CCR => H CR

Our Theorem 2: There exist h that are CCR but not CR

These results are obtained via a **general framework**

— Parameterized version of MD: $H = \mathbf{MD}[h, \mathsf{Split}, \mathsf{S}]$

— RS Security framework: Yields both old and new definitions of security for h

The framework

— Allows us to formalize and prove folklore results

— Is used to prove some new results

— Is pedagogically valuable in unifying results in the area

Some of our **other results**

— We give an MD variant that is more efficient than MD

— Memory-efficient reductions

— Various separations and counter-examples

Caveats and FAQ

Caveats and FAQ

1. We don’t design CCR compression functions.
But existing candidates include the compression functions of SHA256, SHA512

Caveats and FAQ

1. We don't design CCR compression functions.
But existing candidates include the compression functions of SHA256, SHA512

2. MD5 and SHA-1 do not have CCR compression functions.
We can't fix broken hash functions.

Caveats and FAQ

1. We don't design CCR compression functions.
   But existing candidates include the compression functions of SHA256, SHA512

2. MD5 and SHA-1 do not have CCR compression functions.
   We can't fix broken hash functions.

3. Our work is **ONLY about CR** of H, not other attributes such as indifferentiability.
   Although hash functions have many usages, CR is central due to certificates.

Caveats and FAQ

1. We don't design CCR compression functions.
But existing candidates include the compression functions of SHA256, SHA512

2. MD5 and SHA-1 do not have CCR compression functions.
We can't fix broken hash functions.

3. Our work is **ONLY about CR** of H, not other attributes such as indifferentiability.
Although hash functions have many usages, CR is central due to certificates.

4. For the result that: h **is X-secure implies** H **is CR** we said that X = CCR suffices
Q: Is there an X **weaker than CCR** for which the result holds?
A: YES, and our framework allows us to define such properties X.
But the gains from further weakening the assumption X are moot …

Caveats and FAQ

1. We don't design CCR compression functions.
But existing candidates include the compression functions of SHA256, SHA512

2. MD5 and SHA-1 do not have CCR compression functions.
We can't fix broken hash functions.

3. Our work is **ONLY about CR** of H, not other attributes such as indifferentiability.
Although hash functions have many usages, CR is central due to certificates.

4. For the result that: h **is X-secure implies** H **is CR** we said that X = CCR suffices
Q: Is there an X **weaker than CCR** for which the result holds?
A: YES, and our framework allows us to define such properties X.
But the gains from further weakening the assumption X are moot …

5. A lot of our work formalizes, extends and unifies folklore or known results.
Nothing we do is technically hard.

The MD Framework

Splitting function $\mathsf{Split} : D \rightarrow (\{0,1\}^{h.ml})^*$

Set of starting points $\mathsf{S} \subseteq \{0,1\}^{h.cl}$

$$H = \mathbf{MD}[h, \mathsf{Split}, \mathsf{S}]$$

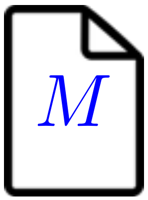
<table>
<tr><th>H</th><th>h</th><th>Split</th><th>S</th></tr>
<tr><td>MD5</td><td>md5</td><td>M || 1 || 0…0 || $\langle |M| \rangle_{64}$</td><td>{0x67452301 || 0xefcdab89 || 0x98badcfe || 0x10325476}</td></tr>
<tr><td>SHA-1</td><td>sha1</td><td>M || 1 || 0…0 || $\langle |M| \rangle_{64}$</td><td>{0x67452301 || 0xefcdab89 || 0x98badcfe ||0x10325476 || 0xc3d2e1f0}</td></tr>
<tr><td>SHA-256</td><td>sha256</td><td>M || 1 || 0…0 || $\langle |M| \rangle_{64}$</td><td>{0x6a09e667 || 0xbb67ae85 || 0x3c6ef372 || 0xa54ff53a || 0x510e527f || 0x9b05688c || 0x1f83d9ab || 0x5be0cd19}</td></tr>
<tr><td>SHA-512</td><td>sha512</td><td>M || 1 || 0…0 || $\langle |M| \rangle_{128}$</td><td>{0x6a09e667f3bcc908 || 0xbb67ae8584caa73b || 0x3c6ef372fe94f82b || 0xa54ff53a5f1d36f1 || 0x510e527fade682d1 || 0x9b05688c2b3e6c1f || 0x1f83d9abfb41bd6b || 0x5be0cd19137e2179}</td></tr>
</table>

The MD Framework

Splitting function $\mathsf{Split} : D \rightarrow (\{0,1\}^{h.ml})^*$

Set of starting points $\mathsf{S} \subseteq \{0,1\}^{h.cl}$

$$H = \mathbf{MD}[h, \mathsf{Split}, \mathsf{S}]$$



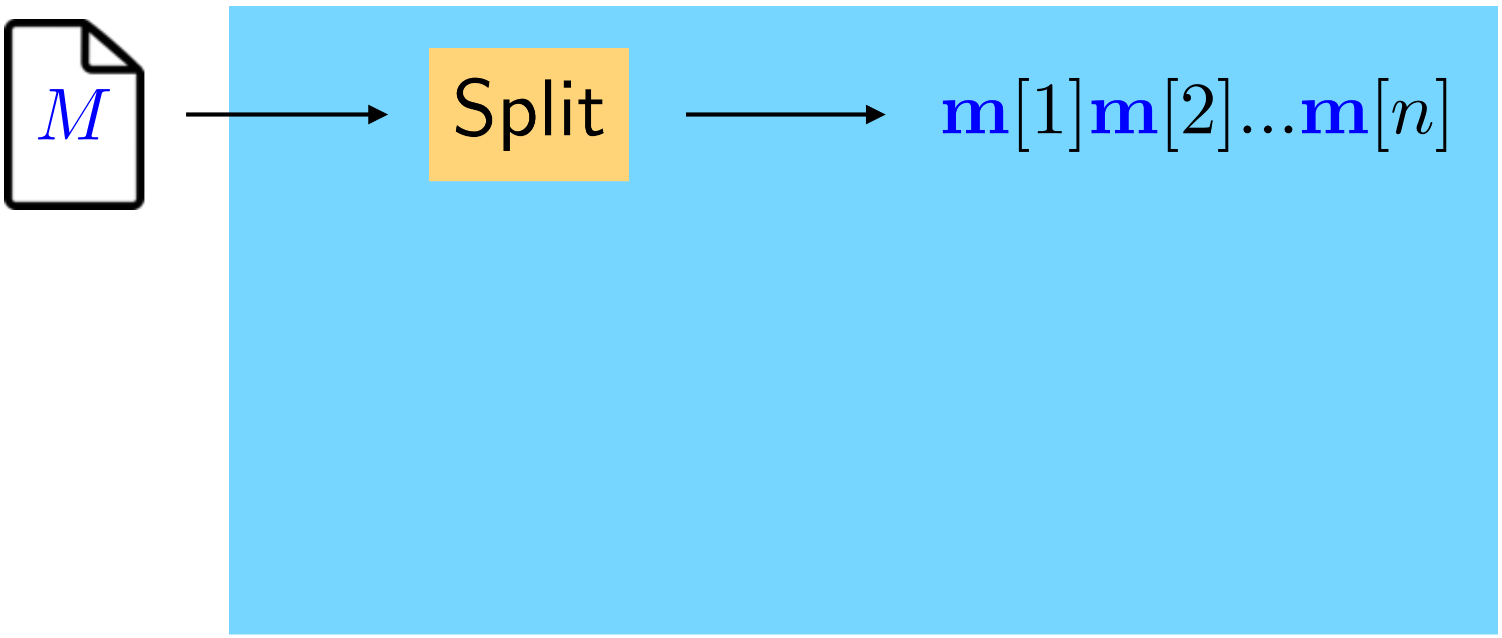
<table>
<tr><th>H</th><th>h</th><th>Split</th><th>S</th></tr>
<tr><td>MD5</td><td>md5</td><td>M || 1 || 0…0 || ⟨|M|⟩$_{64}$</td><td>{0x67452301 || 0xefcdab89 || 0x98badcfe || 0x10325476}</td></tr>
<tr><td>SHA-1</td><td>sha1</td><td>M || 1 || 0…0 || ⟨|M|⟩$_{64}$</td><td>{0x67452301 || 0xefcdab89 || 0x98badcfe ||0x10325476 || 0xc3d2e1f0}</td></tr>
<tr><td>SHA-256</td><td>sha256</td><td>M || 1 || 0…0 || ⟨|M|⟩$_{64}$</td><td>{0x6a09e667 || 0xbb67ae85 || 0x3c6ef372 || 0xa54ff53a || 0x510e527f || 0x9b05688c || 0x1f83d9ab || 0x5be0cd19}</td></tr>
<tr><td>SHA-512</td><td>sha512</td><td>M || 1 || 0…0 || ⟨|M|⟩$_{128}$</td><td>{0x6a09e667f3bcc908 || 0xbb67ae8584caa73b || 0x3c6ef372fe94f82b || 0xa54ff53a5f1d36f1 || 0x510e527fade682d1 || 0x9b05688c2b3e6c1f || 0x1f83d9abfb41bd6b || 0x5be0cd19137e2179}</td></tr>
</table>

The MD Framework

Splitting function $\text{Split} : D \rightarrow (\{0,1\}^{h.ml})^*$

Set of starting points $\text{S} \subseteq \{0,1\}^{h.cl}$

$$H = \mathbf{MD}[h, \text{Split}, \text{S}]$$



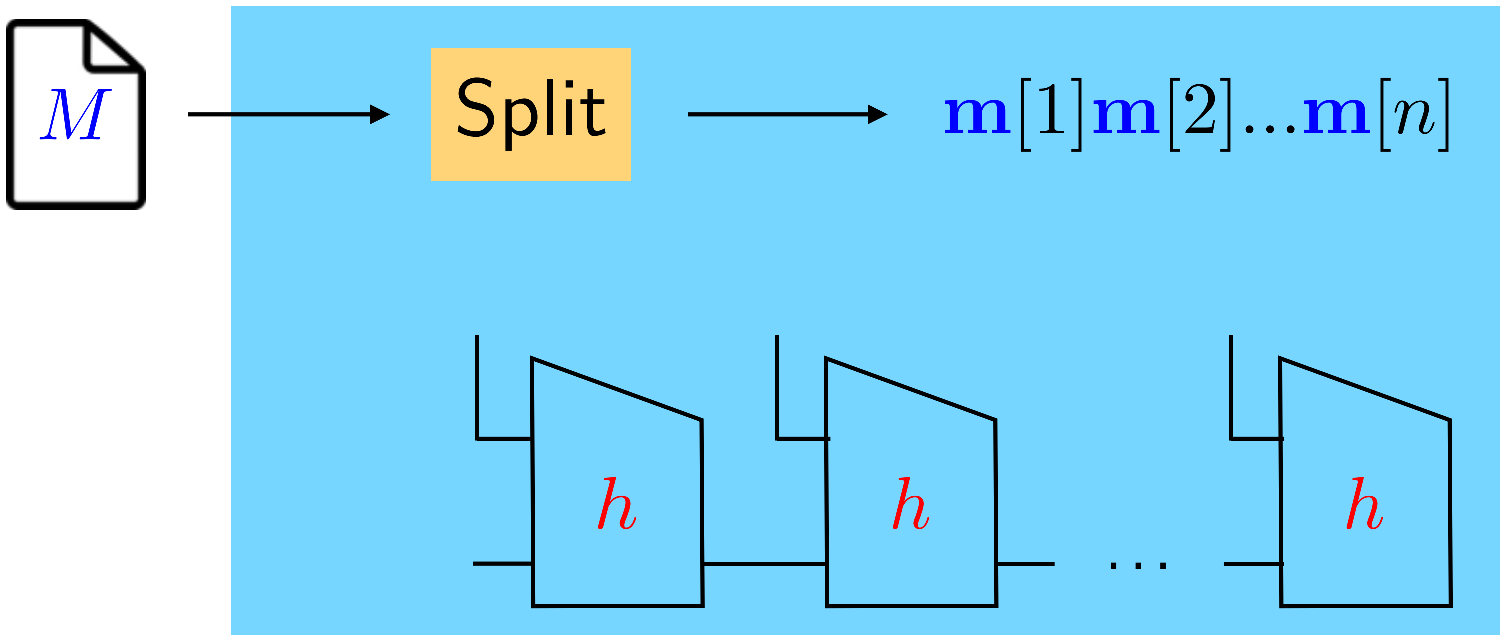
H	h	Split	S
MD5	md5	M ‖ 1 ‖ 0…0 ‖ ⟨|M|⟩$_{64}$	{0x67452301 ‖ 0xefcdab89 ‖ 0x98badcfe ‖ 0x10325476}
SHA-1	sha1	M ‖ 1 ‖ 0…0 ‖ ⟨|M|⟩$_{64}$	{0x67452301 ‖ 0xefcdab89 ‖ 0x98badcfe ‖0x10325476 ‖ 0xc3d2e1f0}
SHA-256	sha256	M ‖ 1 ‖ 0…0 ‖ ⟨|M|⟩$_{64}$	{0x6a09e667 ‖ 0xbb67ae85 ‖ 0x3c6ef372 ‖ 0xa54ff53a ‖ 0x510e527f ‖ 0x9b05688c ‖ 0x1f83d9ab ‖ 0x5be0cd19}
SHA-512	sha512	M ‖ 1 ‖ 0…0 ‖ ⟨|M|⟩$_{128}$	{0x6a09e667f3bcc908 ‖ 0xbb67ae8584caa73b ‖ 0x3c6ef372fe94f82b ‖ 0xa54ff53a5f1d36f1 ‖ 0x510e527fade682d1 ‖ 0x9b05688c2b3e6c1f ‖ 0x1f83d9abfb41bd6b ‖ 0x5be0cd19137e2179}

The MD Framework

Splitting function $\mathsf{Split} : D \to (\{0,1\}^{h.ml})^*$

Set of starting points $\mathsf{S} \subseteq \{0,1\}^{h.cl}$

$$H = \mathbf{MD}[h, \mathsf{Split}, \mathsf{S}]$$



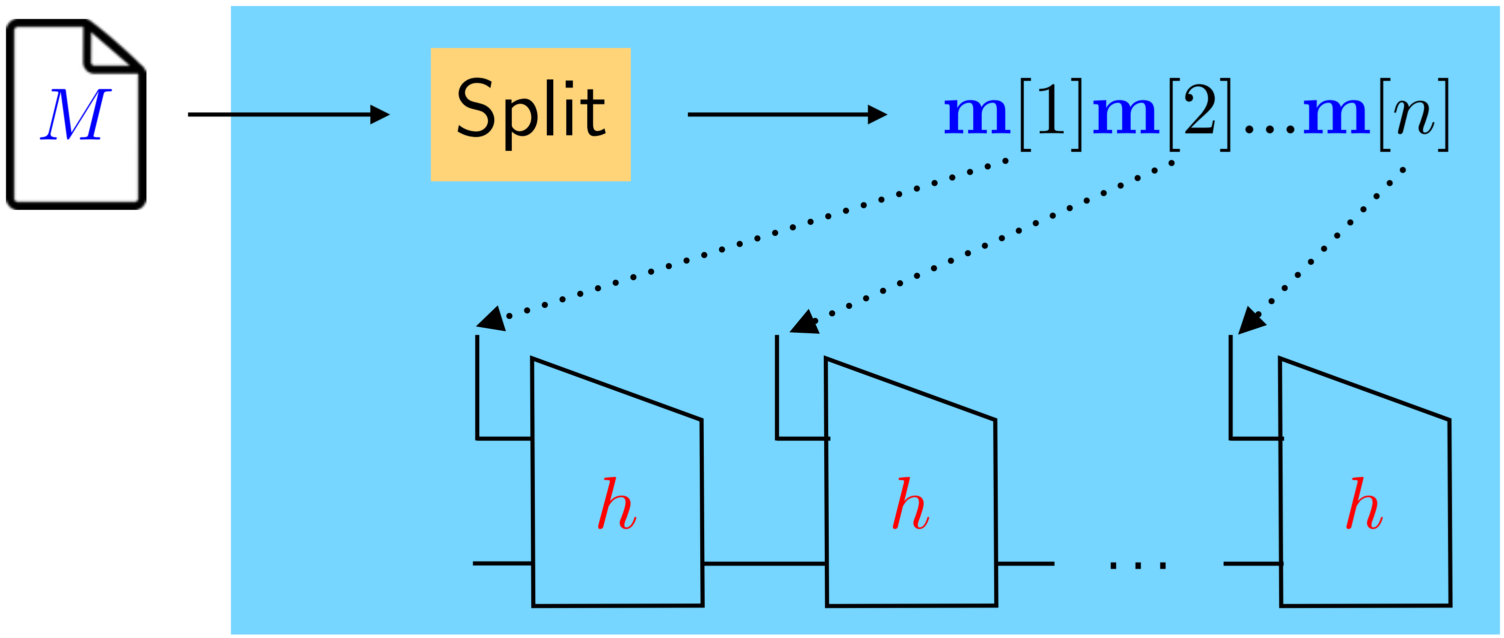
H	h	Split	S
MD5	md5	$M \Vert 1 \Vert 0\ldots0 \Vert \langle\vert M\vert\rangle_{64}$	{0x67452301 $\Vert$ 0xefcdab89 $\Vert$ 0x98badcfe $\Vert$ 0x10325476}
SHA-1	sha1	$M \Vert 1 \Vert 0\ldots0 \Vert \langle\vert M\vert\rangle_{64}$	{0x67452301 $\Vert$ 0xefcdab89 $\Vert$ 0x98badcfe $\Vert$0x10325476 $\Vert$ 0xc3d2e1f0}
SHA-256	sha256	$M \Vert 1 \Vert 0\ldots0 \Vert \langle\vert M\vert\rangle_{64}$	{0x6a09e667 $\Vert$ 0xbb67ae85 $\Vert$ 0x3c6ef372 $\Vert$ 0xa54ff53a $\Vert$ 0x510e527f $\Vert$ 0x9b05688c $\Vert$ 0x1f83d9ab $\Vert$ 0x5be0cd19}
SHA-512	sha512	$M \Vert 1 \Vert 0\ldots0 \Vert \langle\vert M\vert\rangle_{128}$	{0x6a09e667f3bcc908 $\Vert$ 0xbb67ae8584caa73b $\Vert$ 0x3c6ef372fe94f82b $\Vert$ 0xa54ff53a5f1d36f1 $\Vert$ 0x510e527fade682d1 $\Vert$ 0x9b05688c2b3e6c1f $\Vert$ 0x1f83d9abfb41bd6b $\Vert$ 0x5be0cd19137e2179}

The MD Framework

Splitting function $\mathsf{Split} : D \to (\{0,1\}^{h.ml})^*$

Set of starting points $\mathsf{S} \subseteq \{0,1\}^{h.cl}$

$$H = \mathbf{MD}[h, \mathsf{Split}, \mathsf{S}]$$



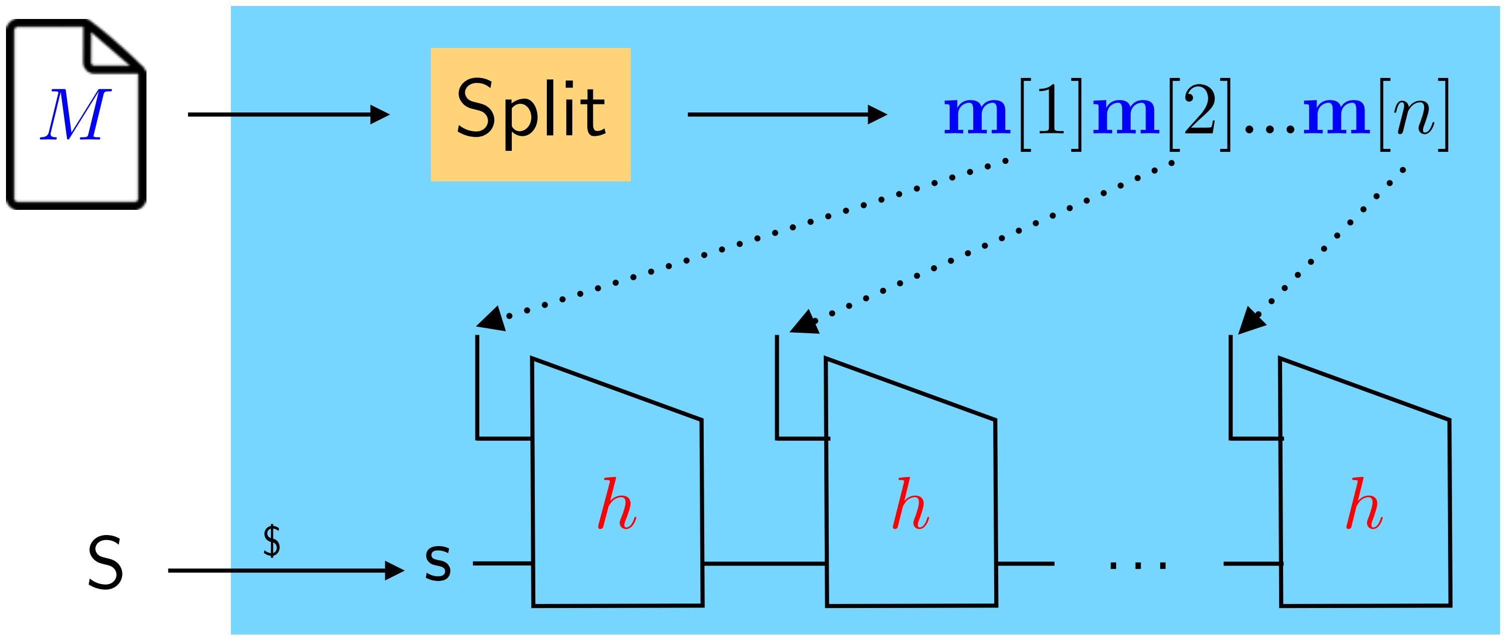
H	h	Split	S
MD5	md5	M ‖ 1 ‖ 0…0 ‖ $\langle|M|\rangle_{64}$	{0x67452301 ‖ 0xefcdab89 ‖ 0x98badcfe ‖ 0x10325476}
SHA-1	sha1	M ‖ 1 ‖ 0…0 ‖ $\langle|M|\rangle_{64}$	{0x67452301 ‖ 0xefcdab89 ‖ 0x98badcfe ‖0x10325476 ‖ 0xc3d2e1f0}
SHA-256	sha256	M ‖ 1 ‖ 0…0 ‖ $\langle|M|\rangle_{64}$	{0x6a09e667 ‖ 0xbb67ae85 ‖ 0x3c6ef372 ‖ 0xa54ff53a ‖ 0x510e527f ‖ 0x9b05688c ‖ 0x1f83d9ab ‖ 0x5be0cd19}
SHA-512	sha512	M ‖ 1 ‖ 0…0 ‖ $\langle|M|\rangle_{128}$	{0x6a09e667f3bcc908 ‖ 0xbb67ae8584caa73b ‖ 0x3c6ef372fe94f82b ‖ 0xa54ff53a5f1d36f1 ‖ 0x510e527fade682d1 ‖ 0x9b05688c2b3e6c1f ‖ 0x1f83d9abfb41bd6b ‖ 0x5be0cd19137e2179}

The MD Framework

Splitting function $\text{Split} : D \to (\{0,1\}^{h.ml})^*$

Set of starting points $\text{S} \subseteq \{0,1\}^{h.cl}$

$$H = \mathbf{MD}[h, \text{Split}, \text{S}]$$



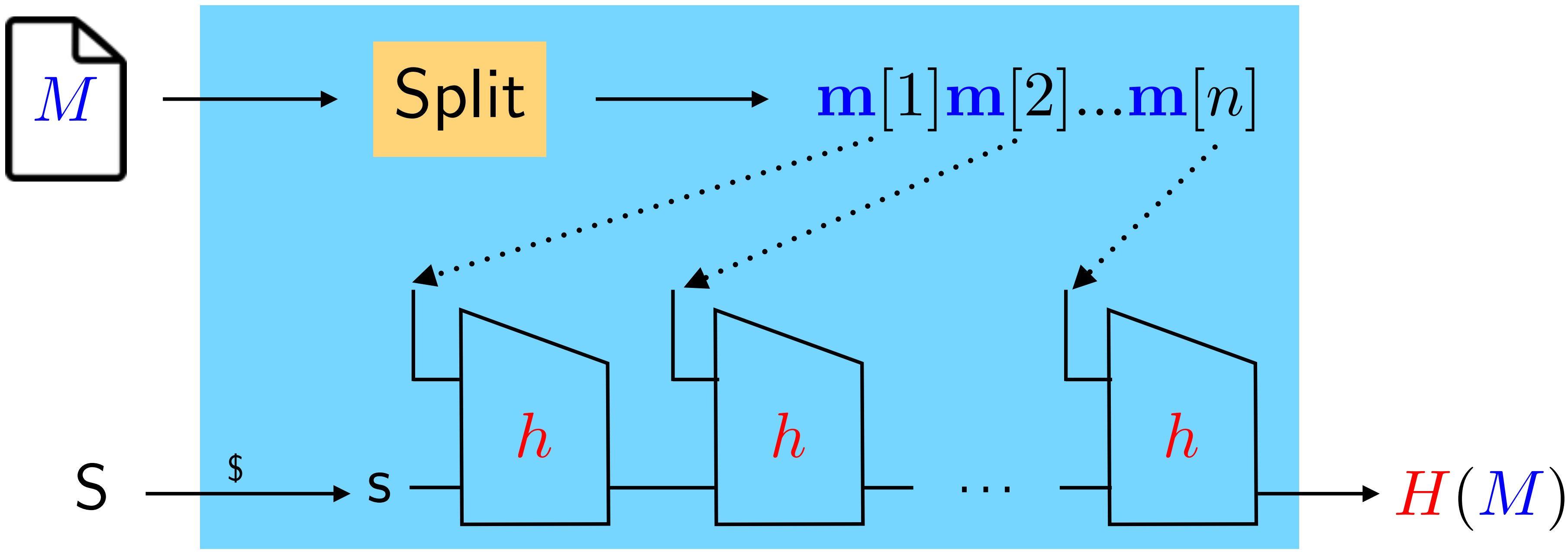
H	h	Split	S
MD5	md5	M ‖ 1 ‖ 0…0 ‖ ⟨|M|⟩$_{64}$	{0x67452301 ‖ 0xefcdab89 ‖ 0x98badcfe ‖ 0x10325476}
SHA-1	sha1	M ‖ 1 ‖ 0…0 ‖ ⟨|M|⟩$_{64}$	{0x67452301 ‖ 0xefcdab89 ‖ 0x98badcfe ‖0x10325476 ‖ 0xc3d2e1f0}
SHA-256	sha256	M ‖ 1 ‖ 0…0 ‖ ⟨|M|⟩$_{64}$	{0x6a09e667 ‖ 0xbb67ae85 ‖ 0x3c6ef372 ‖ 0xa54ff53a ‖ 0x510e527f ‖ 0x9b05688c ‖ 0x1f83d9ab ‖ 0x5be0cd19}
SHA-512	sha512	M ‖ 1 ‖ 0…0 ‖ ⟨|M|⟩$_{128}$	{0x6a09e667f3bcc908 ‖ 0xbb67ae8584caa73b ‖ 0x3c6ef372fe94f82b ‖ 0xa54ff53a5f1d36f1 ‖ 0x510e527fade682d1 ‖ 0x9b05688c2b3e6c1f ‖ 0x1f83d9abfb41bd6b ‖ 0x5be0cd19137e2179}

The MD Framework

Splitting function Split : $D \to (\{0,1\}^{h.ml})^*$

Set of starting points $\mathsf{S} \subseteq \{0,1\}^{h.cl}$

$$H = \mathbf{MD}[h, \mathsf{Split}, \mathsf{S}]$$



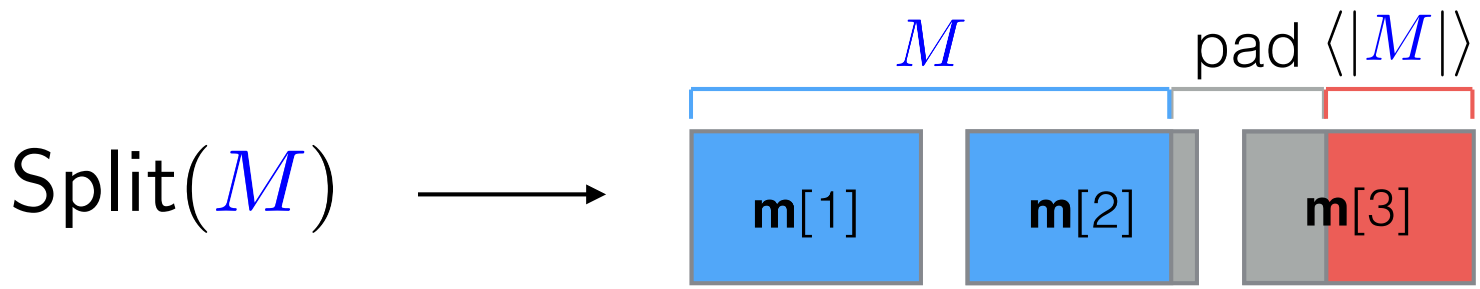
<table>
<tr><th>H</th><th>h</th><th>Split</th><th>S</th></tr>
<tr><td>MD5</td><td>md5</td><td>M || 1 || 0…0 || ⟨|M|⟩$_{64}$</td><td>{0x67452301 || 0xefcdab89 || 0x98badcfe || 0x10325476}</td></tr>
<tr><td>SHA-1</td><td>sha1</td><td>M || 1 || 0…0 || ⟨|M|⟩$_{64}$</td><td>{0x67452301 || 0xefcdab89 || 0x98badcfe ||0x10325476 || 0xc3d2e1f0}</td></tr>
<tr><td>SHA-256</td><td>sha256</td><td>M || 1 || 0…0 || ⟨|M|⟩$_{64}$</td><td>{0x6a09e667 || 0xbb67ae85 || 0x3c6ef372 || 0xa54ff53a || 0x510e527f || 0x9b05688c || 0x1f83d9ab || 0x5be0cd19}</td></tr>
<tr><td>SHA-512</td><td>sha512</td><td>M || 1 || 0…0 || ⟨|M|⟩$_{128}$</td><td>{0x6a09e667f3bcc908 || 0xbb67ae8584caa73b || 0x3c6ef372fe94f82b || 0xa54ff53a5f1d36f1 || 0x510e527fade682d1 || 0x9b05688c2b3e6c1f || 0x1f83d9abfb41bd6b || 0x5be0cd19137e2179}</td></tr>
</table>

Possible conditions on Split

Suffix-free

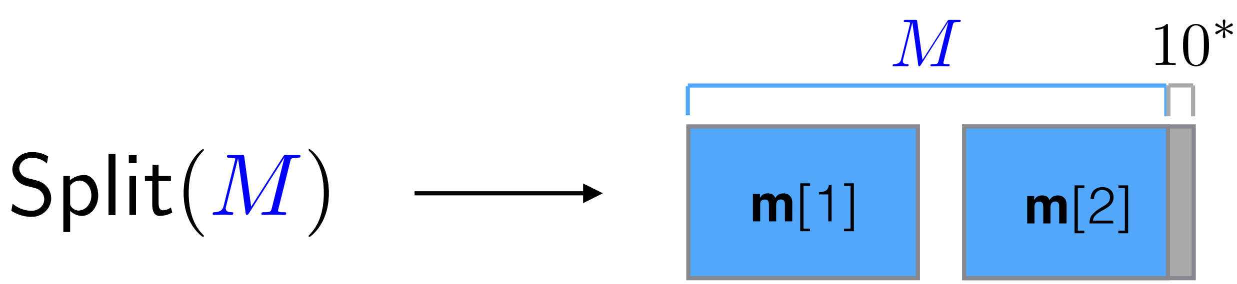
After you apply Split on two distinct messages, neither resulting vector is a suffix of the other.

Typical suffix-free encoding of M (such as in SHA-256):



Injective

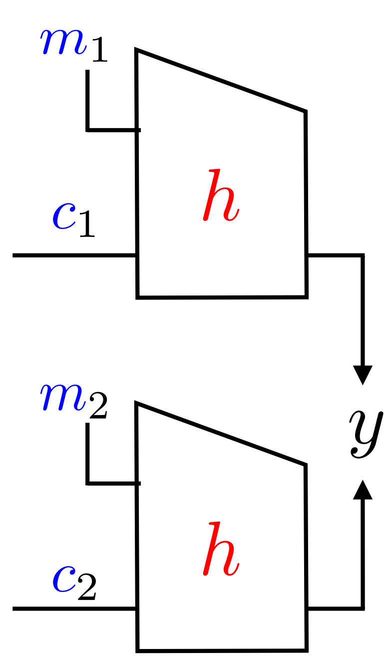
After you apply Split on two distinct messages, you get two distinct vectors.



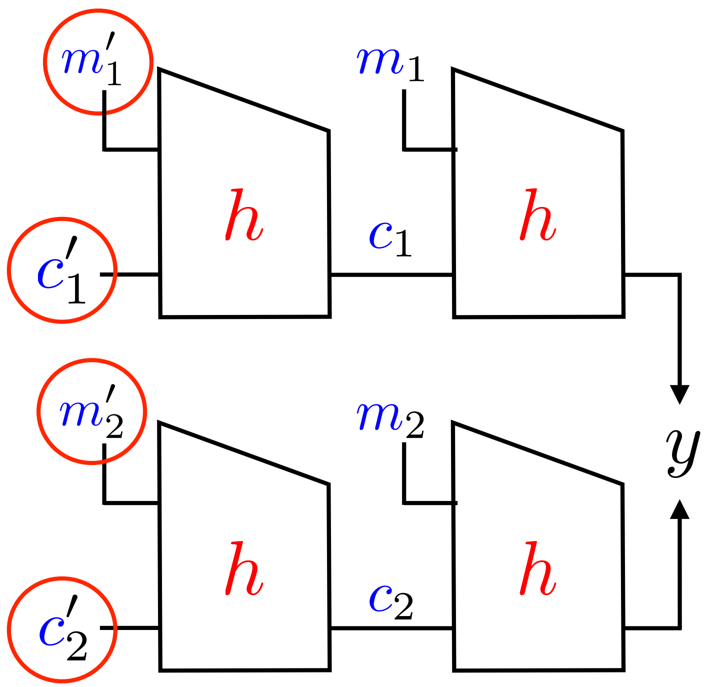
Split(M) is one block shorter, so hashing uses one less call to the compression function. Faster!

	To win, $\mathcal{A}$ must find	such that
CR	$(m_1, c_1) \neq (m_2, c_2)$	$h(m_1, c_1) = h(m_2, c_2)$
CCR	$(m_1, c_1) \neq (m_2, c_2)$ $(m_1', c_1'), (m_2', c_2')$	$h(m_1, c_1) = h(m_2, c_2)$ $c_1 \in \{\mathsf{s}, h(m_1', c_1')\}$ $c_2 \in \{\mathsf{s}, h(m_2', c_2')\}$
Pre	(m, c)	$h(m, c) = \mathsf{s}$

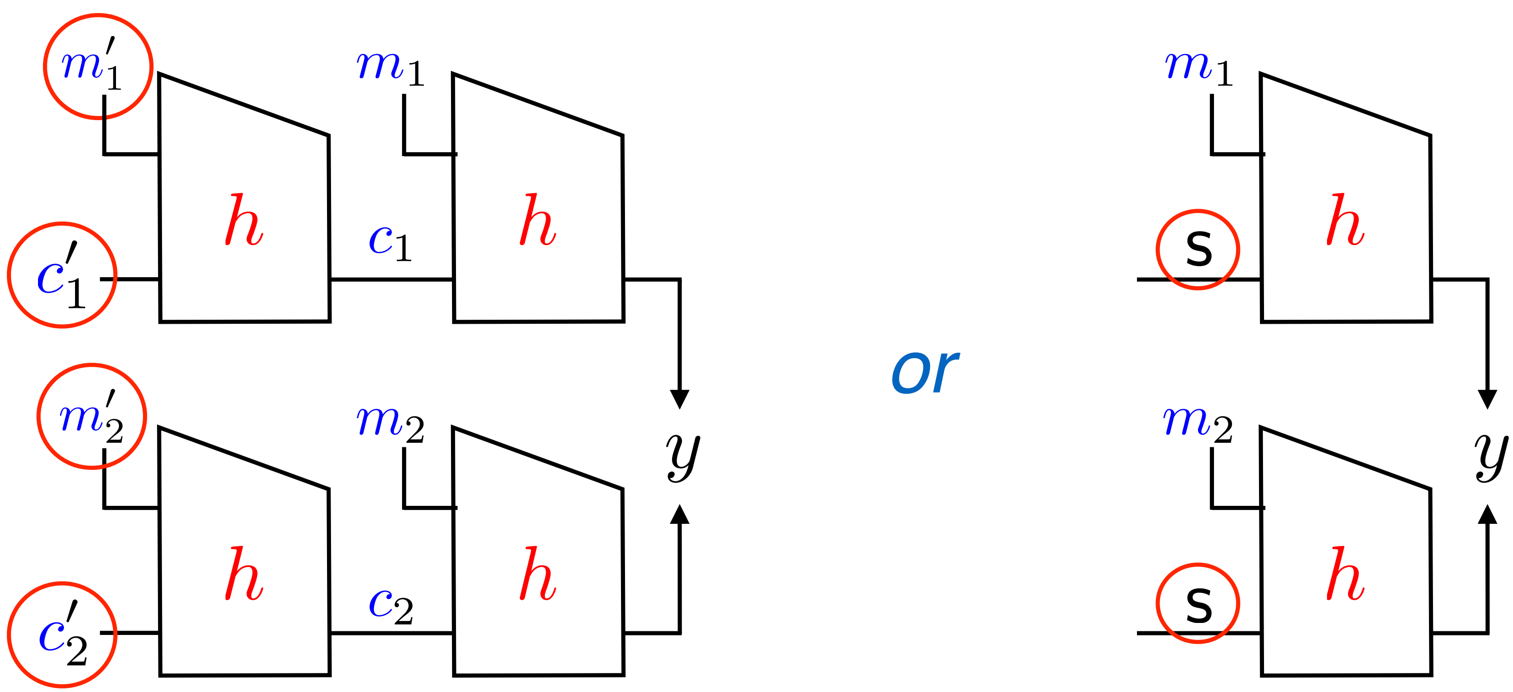
	To win, $\mathcal{A}$ must find	such that
CR	$(m_1, c_1) \neq (m_2, c_2)$	$h(m_1, c_1) = h(m_2, c_2)$
CCR	$(m_1, c_1) \neq (m_2, c_2)$ $(m'_1, c'_1), (m'_2, c'_2)$	$h(m_1, c_1) = h(m_2, c_2)$ $c_1 \in \{\mathsf{s}, h(m'_1, c'_1)\}$ $c_2 \in \{\mathsf{s}, h(m'_2, c'_2)\}$
Pre	(m, c)	$h(m, c) = \mathsf{s}$



	To win, $\mathcal{A}$ must find	such that
CR	$(m_1, c_1) \neq (m_2, c_2)$	$h(m_1, c_1) = h(m_2, c_2)$
CCR	$(m_1, c_1) \neq (m_2, c_2)$ $(m_1', c_1'), (m_2', c_2')$	$h(m_1, c_1) = h(m_2, c_2)$ $c_1 \in \{\mathsf{s}, h(m_1', c_1')\}$ $c_2 \in \{\mathsf{s}, h(m_2', c_2')\}$
Pre	(m, c)	$h(m, c) = \mathsf{s}$

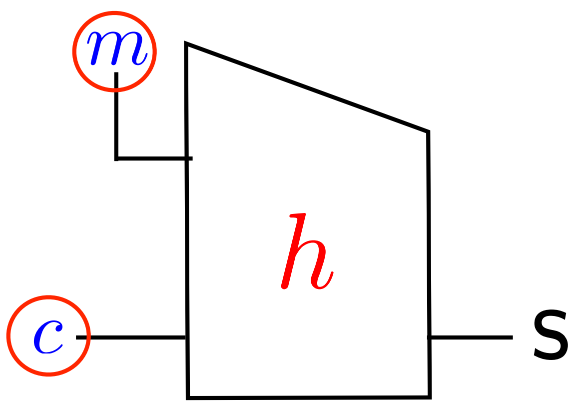


	To win, $\mathcal{A}$ must find	such that
CR	$(m_1, c_1) \neq (m_2, c_2)$	$h(m_1, c_1) = h(m_2, c_2)$
CCR	$(m_1, c_1) \neq (m_2, c_2)$ $(m_1', c_1'), (m_2', c_2')$	$h(m_1, c_1) = h(m_2, c_2)$ $c_1 \in \{\mathsf{s}, h(m_1', c_1')\}$ $c_2 \in \{\mathsf{s}, h(m_2', c_2')\}$
Pre	(m, c)	$h(m, c) = \mathsf{s}$



	To win, $\mathcal{A}$ must find	such that
CR	$(m_1, c_1) \neq (m_2, c_2)$	$h(m_1, c_1) = h(m_2, c_2)$
CCR	$(m_1, c_1) \neq (m_2, c_2)$ $(m_1', c_1'), (m_2', c_2')$	$h(m_1, c_1) = h(m_2, c_2)$ $c_1 \in \{\mathsf{s}, h(m_1', c_1')\}$ $c_2 \in \{\mathsf{s}, h(m_2', c_2')\}$
Pre	(m, c)	$h(m, c) = \mathsf{s}$

Pre



The RS Security Framework

In the previous slide we defined **CR**, **CCR**, and **Pre**.
We give a general definitional framework that yields these and other definitions.

Our definition of security for a compression function h is parameterized by a relation

$$\mathsf{R} : \{0,1\}^* \times \{0,1\}^* \to \{\mathsf{true}, \mathsf{false}\}$$

and a set $\mathsf{S} \subseteq \{0,1\}^*$

Game $\mathrm{G}^{\mathrm{RS}}_{h}(\mathcal{A})$

$\mathrm{s} \leftarrow\!\!\$\; \mathrm{S}\ ;\ out \leftarrow\!\!\$\; \mathcal{A}(\mathrm{s})$

Return $\mathrm{R}(\mathrm{s}, out)$

$\mathsf{R}(\mathsf{s}, out)$

- s: starting value
- out: string that adversary outputs

For $\mathrm{R_{cr}}$ we have $\mathsf{s} = \varepsilon$.

The RS Security Framework

In the previous slide we defined **CR**, **CCR**, and **Pre**.
We give a general definitional framework that yields these and other definitions.

Our definition of security for a compression function h is parameterized by a relation

$$\mathsf{R} : \{0,1\}^* \times \{0,1\}^* \to \{\mathsf{true}, \mathsf{false}\}$$

and a set $\mathsf{S} \subseteq \{0,1\}^*$

Game $\mathrm{G}_h^{\mathrm{RS}}(\mathcal{A})$

$\mathsf{s} \leftarrow\!\!\$\ \mathsf{S}$; $out \leftarrow\!\!\$\ \mathcal{A}(\mathsf{s})$

Return $\mathsf{R}(\mathsf{s}, out)$

$\mathsf{R}(\mathsf{s}, out)$

s: starting value

out: string that adversary outputs

For R_{cr} we have $\mathsf{s} = \varepsilon$.

R	out	$\mathsf{R}(\mathsf{s}, out)$ **returns true iff**	**Property**
R_{cr}	$((m_1, c_1), (m_2, c_2))$	$h(m_1, c_1) = h(m_2, c_2)$	Collision resistance
R_{ccr}	$((m_1, c_1), (m_2, c_2)),$ $((m_1', c_1'), (m_2', c_2')))$	$\mathsf{R}_{cr}(\varepsilon, ((m_1, c_1), (m_2, c_2)))\wedge$ $(c_1 \in \{\mathsf{s}, h(m_1', c_1')\})\wedge$ $(c_2 \in \{\mathsf{s}, h(m_2', c_2')\})$	Constrained CR
R_{pre}	(m, c)	$h(m, c) = \mathsf{s}$	Pre-image resistance

Results

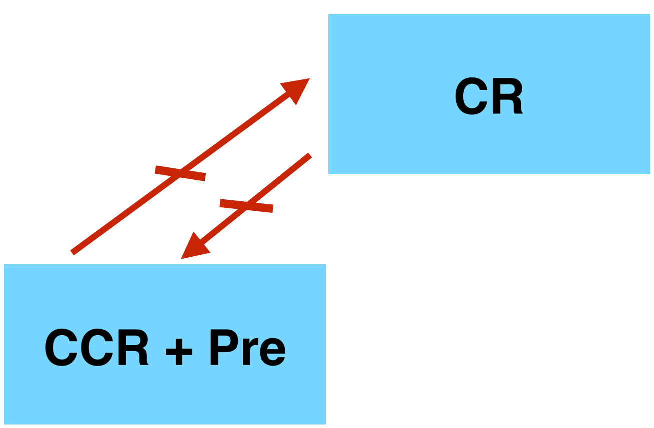
Typically, $S = \{s\}$ is a singleton set.

	If Split is	and h is	then H = MD[h,Split,S] is	Notes
1	Suffix-free	CR	CR	Known [Me,Da], reproved
2	Suffix-free	CCR	CR	
3	Injective	CCR and Pre	CR	Folklore for CR and Pre [AnSt11]

Results

Typically, $S = \{s\}$ is a singleton set.

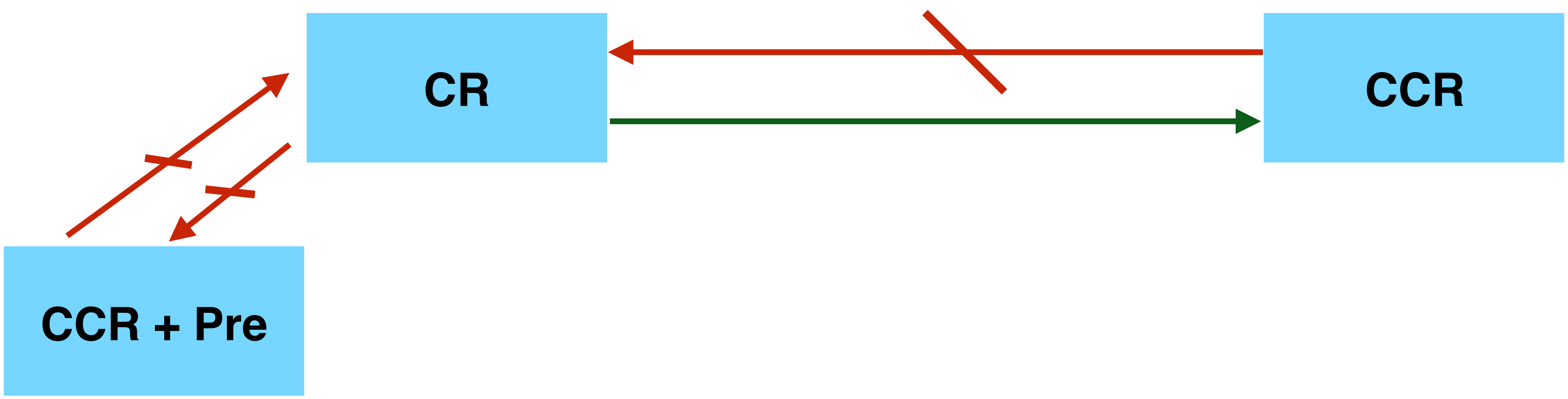
	If Split is	and h is	then H = MD[h,Split,S] is	Notes
1	Suffix-free	CR	CR	Known [Me,Da], reproved
2	Suffix-free	CCR	CR	
3	Injective	CCR and Pre	CR	Folklore for CR and Pre [AnSt11]



Results

Typically, $S = \{s\}$ is a singleton set.

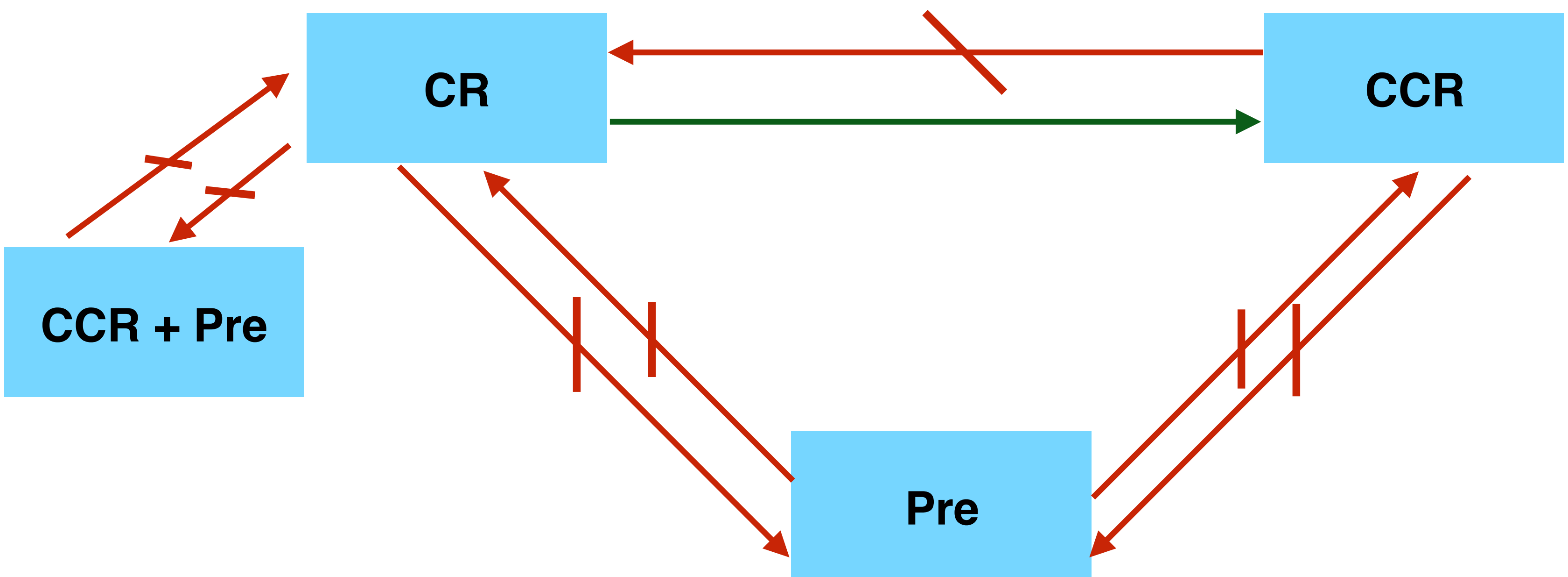
	If Split is	and h is	then H = MD[h,Split,S] is	Notes
1	Suffix-free	CR	CR	Known [Me,Da], reproved
2	Suffix-free	CCR	CR	
3	Injective	CCR and Pre	CR	Folklore for CR and Pre [AnSt11]



Results

Typically, $S = \{s\}$ is a singleton set.

	If Split is	and h is	then H = MD[h,Split,S] is	Notes
1	Suffix-free	CR	CR	Known [Me,Da], reproved
2	Suffix-free	CCR	CR	
3	Injective	CCR and Pre	CR	Folklore for CR and Pre [AnSt11]

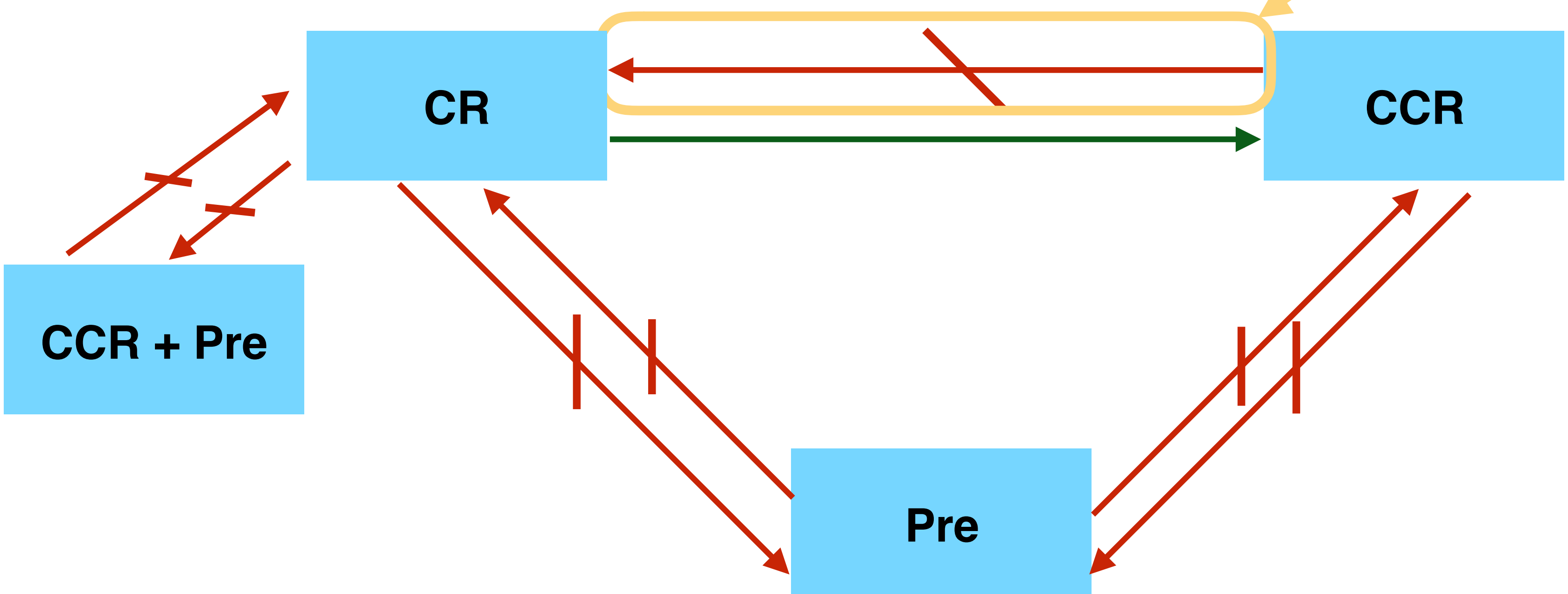


Results

Typically, $S = \{s\}$ is a singleton set.

	If Split is	and h is	then H = MD[h,Split,S] is	Notes
1	Suffix-free	CR	CR	Known [Me,Da], reproved
2	Suffix-free	CCR	CR	
3	Injective	CCR and Pre	CR	Fo

Discussed in the rest of this talk



Theorem
Let Split be a suffix-free splitting function. Given an adversary $\mathcal{A}_H$, we define $\mathcal{A}_h$ such that

$$\mathbf{Adv}_H^{\mathsf{cr}}(\mathcal{A}_H) \leq \mathbf{Adv}_h^{\mathsf{R_{ccr}S}}(\mathcal{A}_h)$$

The time complexity of $\mathcal{A}_h$ is approximately that of $\mathcal{A}_H$ plus the time to compute H. The memory complexity of $\mathcal{A}_h$ is the maximum of the memory complexity of $\mathcal{A}_H$ and term linear in the length of the output of $\mathcal{A}_H$.

```
adversary A_h(s)
(M_1, M_2) ← A_H(s, ε)
m_1 ← Split(M_1) ; m_2 ← Split(M_2) ; n_1 ← |m_1| ; n_2 ← |m_2|
c_1[1] ← s ; c_2[1] ← s
For i = 1, ..., n_1 do c_1[i + 1] ← h(m_1[i], c_1[i])
For i = 1, ..., n_2 do c_2[i + 1] ← h(m_2[i], c_2[i])
b ← argmin_d(n_d)
For i = 0, ..., n_b − 2 do
    (m_1, c_1) ← (m_1[n_1 − i], c_1[n_1 − i])
    (m_2, c_2) ← (m_2[n_2 − i], c_2[n_2 − i])
    a_1 ← (m_1[n_1 − i − 1], c_1[n_1 − i − 1])
    a_2 ← (m_2[n_2 − i − 1], c_2[n_2 − i − 1])
    If (m_1, c_1) ≠ (m_2, c_2) then
        Return ((m_1, c_1), (m_2, c_2), a_1, a_2)
If n_1 = n_2 then
    (m_1, c_1) ← (m_1[1], c_1[1]) ; (m_2, c_2) ← (m_2[1], c_2[1])
    a_1 ← 1; a_2 ← 2
    Return ((m_1, c_1), (m_2, c_2), a_1, a_2)
(m_1, c_1) ← (m_1[n_1 − n_b + 1], c_1[n_1 − n_b + 1])
(m_2, c_2) ← (m_2[n_2 − n_b + 1], c_2[n_2 − n_b + 1])
a_{3−b} ← (m_{3−b}[n_{3−b} − n_b], c_{3−b}[n_{3−b} − n_b])
a_b ← a_{3−b}
Return ((m_1, c_1), (m_2, c_2), a_1, a_2)
```

Proof uses the back-tracking paradigm of [Me,Da] but constructs a CCR-violating adversary rather than a CR-violating one.

Theorem

Let Split be a suffix-free splitting function. Given an adversary $\mathcal{A}_H$, we define $\mathcal{A}_h$ such that

$$\mathbf{Adv}_H^{\mathsf{cr}}(\mathcal{A}_H) \leq \mathbf{Adv}_h^{\mathsf{R_{ccr}S}}(\mathcal{A}_h)$$

The time complexity of $\mathcal{A}_h$ is approximately that of $\mathcal{A}_H$ plus the time to compute H. The memory complexity of $\mathcal{A}_h$ is the maximum of the memory complexity of $\mathcal{A}_H$ and term linear in the length of the output of $\mathcal{A}_H$.

adversary $\mathcal{A}_h(\mathrm{s})$

$(M_1, M_2) \leftarrow \mathcal{A}_H(\mathrm{s}, \varepsilon)$

$\mathbf{m}_1 \leftarrow \mathrm{Split}(M_1)$; $\mathbf{m}_2 \leftarrow \mathrm{Split}(M_2)$; $n_1 \leftarrow |\mathbf{m}_1|$; $n_2 \leftarrow |\mathbf{m}_2|$

$\mathbf{c}_1[1] \leftarrow \mathrm{s}$; $\mathbf{c}_2[1] \leftarrow \mathrm{s}$

For $i = 1, \ldots, n_1$ do $\mathbf{c}_1[i+1] \leftarrow h(\mathbf{m}_1[i], \mathbf{c}_1[i])$

For $i = 1, \ldots, n_2$ do $\mathbf{c}_2[i+1] \leftarrow h(\mathbf{m}_2[i], \mathbf{c}_2[i])$

$b \leftarrow \mathrm{argmin}_d(n_d)$

For $i = 0, \ldots, n_b - 2$ do

    $(m_1, c_1) \leftarrow (\mathbf{m}_1[n_1 - i], \mathbf{c}_1[n_1 - i])$

    $(m_2, c_2) \leftarrow (\mathbf{m}_2[n_2 - i], \mathbf{c}_2[n_2 - i])$

    $a_1 \leftarrow (\mathbf{m}_1[n_1 - i - 1], \mathbf{c}_1[n_1 - i - 1])$

    $a_2 \leftarrow (\mathbf{m}_2[n_2 - i - 1], \mathbf{c}_2[n_2 - i - 1])$

    If $(m_1, c_1) \neq (m_2, c_2)$ then

        Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$

If $n_1 = n_2$ then

    $(m_1, c_1) \leftarrow (\mathbf{m}_1[1], \mathbf{c}_1[1])$; $(m_2, c_2) \leftarrow (\mathbf{m}_2[1], \mathbf{c}_2[1])$

    $a_1 \leftarrow 1$; $a_2 \leftarrow 2$

    Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$

$(m_1, c_1) \leftarrow (\mathbf{m}_1[n_1 - n_b + 1], \mathbf{c}_1[n_1 - n_b + 1])$

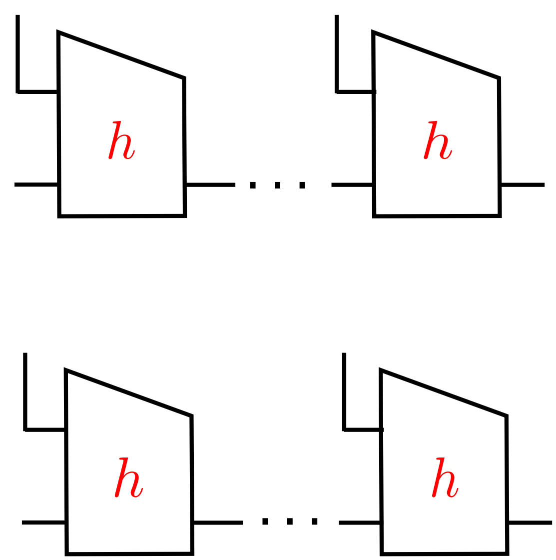
$(m_2, c_2) \leftarrow (\mathbf{m}_2[n_2 - n_b + 1], \mathbf{c}_2[n_2 - n_b + 1])$

$a_{3-b} \leftarrow (\mathbf{m}_{3-b}[n_{3-b} - n_b], \mathbf{c}_{3-b}[n_{3-b} - n_b])$

$a_b \leftarrow a_{3-b}$

Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$

Proof uses the back-tracking paradigm of [Me,Da] but constructs a CCR-violating adversary rather than a CR-violating one.



Theorem

Let Split be a suffix-free splitting function. Given an adversary $\mathcal{A}_H$, we define $\mathcal{A}_h$ such that

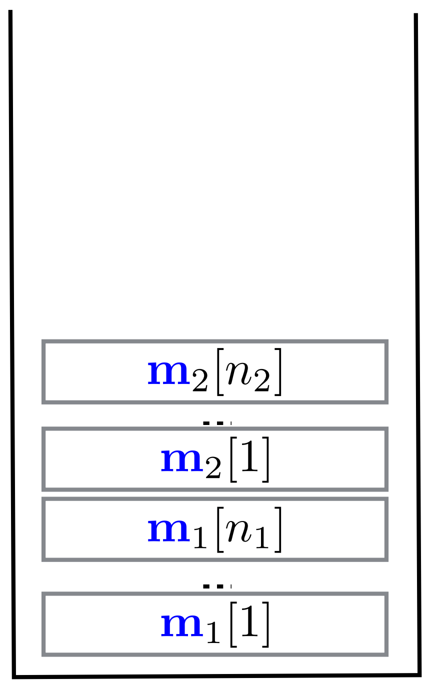
$$\mathbf{Adv}^{\mathsf{cr}}_{H}(\mathcal{A}_H) \leq \mathbf{Adv}^{\mathsf{R_{ccr}S}}_{h}(\mathcal{A}_h)$$

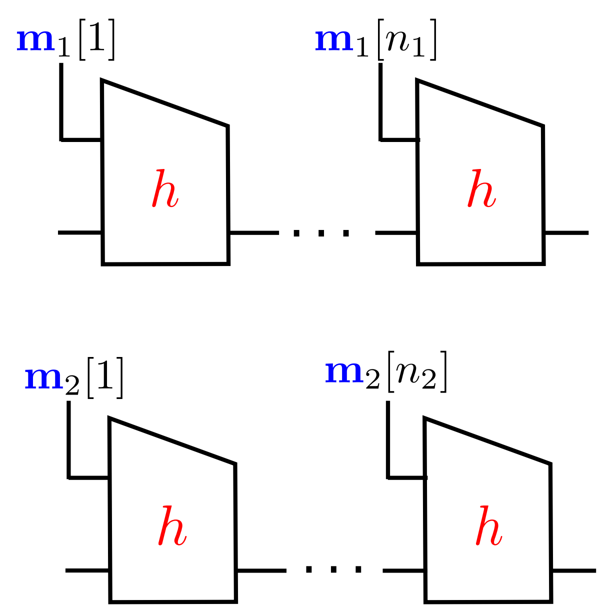
The time complexity of $\mathcal{A}_h$ is approximately that of $\mathcal{A}_H$ plus the time to compute H. The memory complexity of $\mathcal{A}_h$ is the maximum of the memory complexity of $\mathcal{A}_H$ and term linear in the length of the output of $\mathcal{A}_H$.

adversary $\mathcal{A}_h(s)$

$(M_1, M_2) \leftarrow \mathcal{A}_H(s, \varepsilon)$
$\mathbf{m}_1 \leftarrow \mathsf{Split}(M_1)$; $\mathbf{m}_2 \leftarrow \mathsf{Split}(M_2)$; $n_1 \leftarrow |\mathbf{m}_1|$; $n_2 \leftarrow |\mathbf{m}_2|$
$\mathbf{c}_1[1] \leftarrow s$; $\mathbf{c}_2[1] \leftarrow s$
For $i = 1, \ldots, n_1$ do $\mathbf{c}_1[i+1] \leftarrow h(\mathbf{m}_1[i], \mathbf{c}_1[i])$
For $i = 1, \ldots, n_2$ do $\mathbf{c}_2[i+1] \leftarrow h(\mathbf{m}_2[i], \mathbf{c}_2[i])$
$b \leftarrow \mathrm{argmin}_d(n_d)$
For $i = 0, \ldots, n_b - 2$ do
    $(m_1, c_1) \leftarrow (\mathbf{m}_1[n_1 - i], \mathbf{c}_1[n_1 - i])$
    $(m_2, c_2) \leftarrow (\mathbf{m}_2[n_2 - i], \mathbf{c}_2[n_2 - i])$
    $a_1 \leftarrow (\mathbf{m}_1[n_1 - i - 1], \mathbf{c}_1[n_1 - i - 1])$
    $a_2 \leftarrow (\mathbf{m}_2[n_2 - i - 1], \mathbf{c}_2[n_2 - i - 1])$
    If $(m_1, c_1) \neq (m_2, c_2)$ then
        Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$
If $n_1 = n_2$ then
    $(m_1, c_1) \leftarrow (\mathbf{m}_1[1], \mathbf{c}_1[1])$; $(m_2, c_2) \leftarrow (\mathbf{m}_2[1], \mathbf{c}_2[1])$
    $a_1 \leftarrow 1$; $a_2 \leftarrow 2$
    Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$
$(m_1, c_1) \leftarrow (\mathbf{m}_1[n_1 - n_b + 1], \mathbf{c}_1[n_1 - n_b + 1])$
$(m_2, c_2) \leftarrow (\mathbf{m}_2[n_2 - n_b + 1], \mathbf{c}_2[n_2 - n_b + 1])$
$a_{3-b} \leftarrow (\mathbf{m}_{3-b}[n_{3-b} - n_b], \mathbf{c}_{3-b}[n_{3-b} - n_b])$
$a_b \leftarrow a_{3-b}$
Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$

Proof uses the back-tracking paradigm of [Me,Da] but constructs a CCR-violating adversary rather than a CR-violating one.





Theorem

Let Split be a suffix-free splitting function. Given an adversary $\mathcal{A}_H$, we define $\mathcal{A}_h$ such that

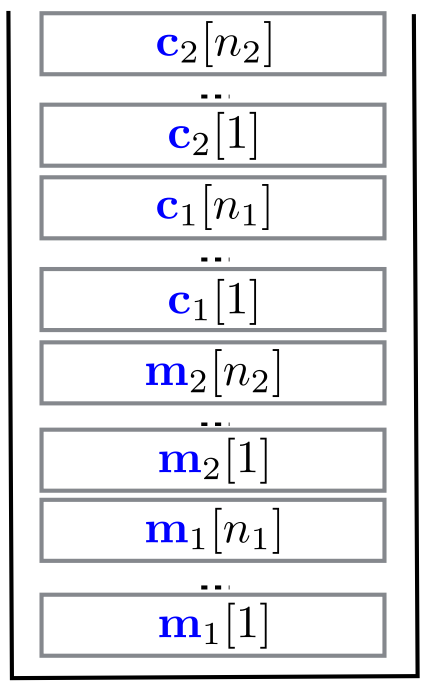
$$\mathbf{Adv}_H^{\mathsf{cr}}(\mathcal{A}_H) \leq \mathbf{Adv}_h^{\mathsf{R_{ccr}S}}(\mathcal{A}_h)$$

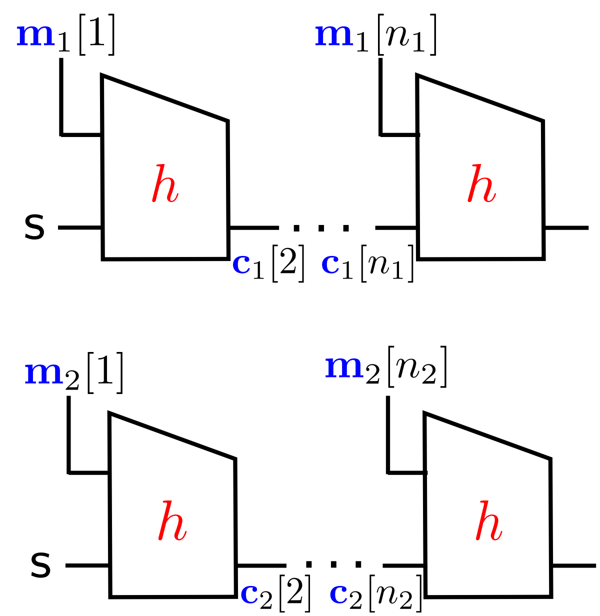
The time complexity of $\mathcal{A}_h$ is approximately that of $\mathcal{A}_H$ plus the time to compute H. The memory complexity of $\mathcal{A}_h$ is the maximum of the memory complexity of $\mathcal{A}_H$ and term linear in the length of the output of $\mathcal{A}_H$.

adversary $\mathcal{A}_h(\mathsf{s})$

$(M_1, M_2) \leftarrow \mathcal{A}_H(\mathsf{s}, \varepsilon)$
$\mathbf{m}_1 \leftarrow \mathsf{Split}(M_1)$; $\mathbf{m}_2 \leftarrow \mathsf{Split}(M_2)$; $n_1 \leftarrow |\mathbf{m}_1|$; $n_2 \leftarrow |\mathbf{m}_2|$
$\mathbf{c}_1[1] \leftarrow \mathsf{s}$; $\mathbf{c}_2[1] \leftarrow \mathsf{s}$
For $i = 1, \ldots, n_1$ do $\mathbf{c}_1[i+1] \leftarrow h(\mathbf{m}_1[i], \mathbf{c}_1[i])$
For $i = 1, \ldots, n_2$ do $\mathbf{c}_2[i+1] \leftarrow h(\mathbf{m}_2[i], \mathbf{c}_2[i])$
$b \leftarrow \mathrm{argmin}_d(n_d)$
For $i = 0, \ldots, n_b - 2$ do
  $(m_1, c_1) \leftarrow (\mathbf{m}_1[n_1 - i], \mathbf{c}_1[n_1 - i])$
  $(m_2, c_2) \leftarrow (\mathbf{m}_2[n_2 - i], \mathbf{c}_2[n_2 - i])$
  $a_1 \leftarrow (\mathbf{m}_1[n_1 - i - 1], \mathbf{c}_1[n_1 - i - 1])$
  $a_2 \leftarrow (\mathbf{m}_2[n_2 - i - 1], \mathbf{c}_2[n_2 - i - 1])$
  If $(m_1, c_1) \neq (m_2, c_2)$ then
    Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$
If $n_1 = n_2$ then
  $(m_1, c_1) \leftarrow (\mathbf{m}_1[1], \mathbf{c}_1[1])$; $(m_2, c_2) \leftarrow (\mathbf{m}_2[1], \mathbf{c}_2[1])$
  $a_1 \leftarrow 1$; $a_2 \leftarrow 2$
  Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$
$(m_1, c_1) \leftarrow (\mathbf{m}_1[n_1 - n_b + 1], \mathbf{c}_1[n_1 - n_b + 1])$
$(m_2, c_2) \leftarrow (\mathbf{m}_2[n_2 - n_b + 1], \mathbf{c}_2[n_2 - n_b + 1])$
$a_{3-b} \leftarrow (\mathbf{m}_{3-b}[n_{3-b} - n_b], \mathbf{c}_{3-b}[n_{3-b} - n_b])$
$a_b \leftarrow a_{3-b}$
Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$

Proof uses the back-tracking paradigm of [Me,Da] but constructs a CCR-violating adversary rather than a CR-violating one.





Theorem

Let Split be a suffix-free splitting function. Given an adversary $\mathcal{A}_H$, we define $\mathcal{A}_h$ such that

$$\mathbf{Adv}^{\mathsf{cr}}_{H}(\mathcal{A}_H) \leq \mathbf{Adv}^{\mathsf{R_{ccr}S}}_{h}(\mathcal{A}_h)$$

The time complexity of $\mathcal{A}_h$ is approximately that of $\mathcal{A}_H$ plus the time to compute H. The memory complexity of $\mathcal{A}_h$ is the maximum of the memory complexity of $\mathcal{A}_H$ and term linear in the length of the output of $\mathcal{A}_H$.

adversary $\mathcal{A}_h(\mathrm{s})$

$(M_1, M_2) \leftarrow \mathcal{A}_H(\mathrm{s}, \varepsilon)$

$\mathbf{m}_1 \leftarrow \mathrm{Split}(M_1)$; $\mathbf{m}_2 \leftarrow \mathrm{Split}(M_2)$; $n_1 \leftarrow |\mathbf{m}_1|$; $n_2 \leftarrow |\mathbf{m}_2|$

$\mathbf{c}_1[1] \leftarrow \mathrm{s}$; $\mathbf{c}_2[1] \leftarrow \mathrm{s}$

For $i = 1, \ldots, n_1$ do $\mathbf{c}_1[i+1] \leftarrow h(\mathbf{m}_1[i], \mathbf{c}_1[i])$

For $i = 1, \ldots, n_2$ do $\mathbf{c}_2[i+1] \leftarrow h(\mathbf{m}_2[i], \mathbf{c}_2[i])$

$b \leftarrow \mathrm{argmin}_d(n_d)$

For $i = 0, \ldots, n_b - 2$ do

  $(m_1, c_1) \leftarrow (\mathbf{m}_1[n_1 - i], \mathbf{c}_1[n_1 - i])$

  $(m_2, c_2) \leftarrow (\mathbf{m}_2[n_2 - i], \mathbf{c}_2[n_2 - i])$

  $a_1 \leftarrow (\mathbf{m}_1[n_1 - i - 1], \mathbf{c}_1[n_1 - i - 1])$

  $a_2 \leftarrow (\mathbf{m}_2[n_2 - i - 1], \mathbf{c}_2[n_2 - i - 1])$

  If $(m_1, c_1) \neq (m_2, c_2)$ then

    Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$

If $n_1 = n_2$ then

  $(m_1, c_1) \leftarrow (\mathbf{m}_1[1], \mathbf{c}_1[1])$; $(m_2, c_2) \leftarrow (\mathbf{m}_2[1], \mathbf{c}_2[1])$

  $a_1 \leftarrow 1$; $a_2 \leftarrow 2$

  Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$

$(m_1, c_1) \leftarrow (\mathbf{m}_1[n_1 - n_b + 1], \mathbf{c}_1[n_1 - n_b + 1])$

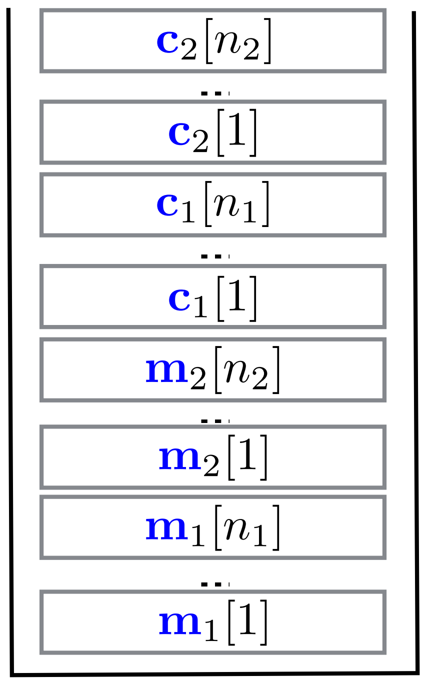
$(m_2, c_2) \leftarrow (\mathbf{m}_2[n_2 - n_b + 1], \mathbf{c}_2[n_2 - n_b + 1])$

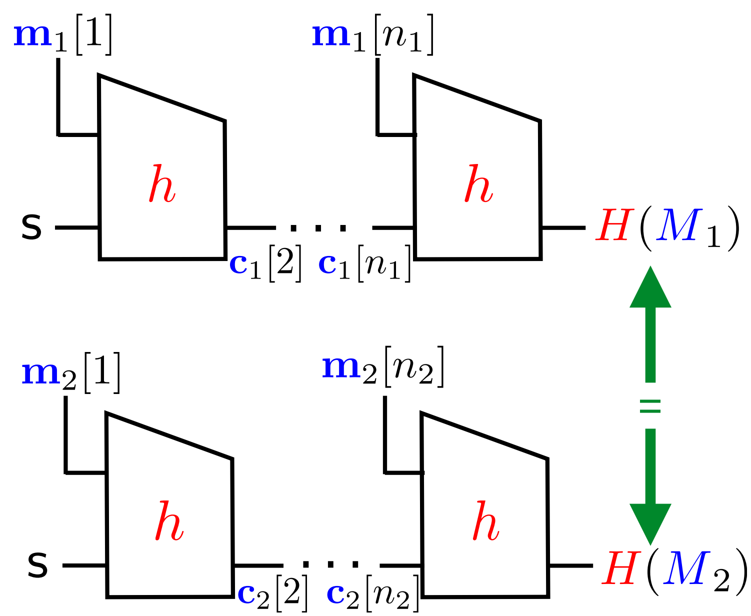
$a_{3-b} \leftarrow (\mathbf{m}_{3-b}[n_{3-b} - n_b], \mathbf{c}_{3-b}[n_{3-b} - n_b])$

$a_b \leftarrow a_{3-b}$

Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$

Proof uses the back-tracking paradigm of [Me,Da] but constructs a CCR-violating adversary rather than a CR-violating one.





Theorem

Let Split be a suffix-free splitting function. Given an adversary $\mathcal{A}_H$, we define $\mathcal{A}_h$ such that

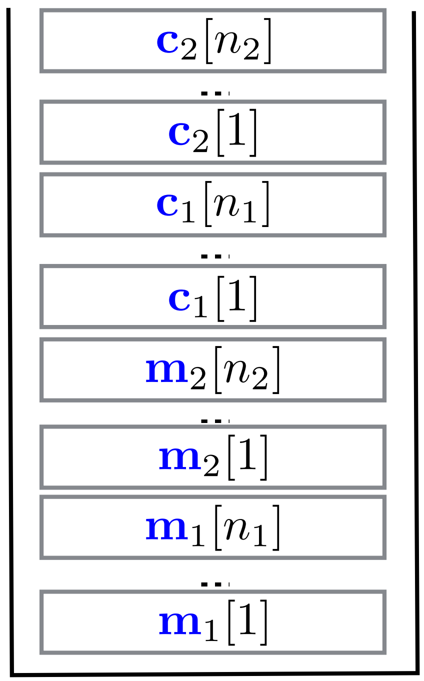
$$\mathbf{Adv}_H^{\mathsf{cr}}(\mathcal{A}_H) \leq \mathbf{Adv}_h^{\mathsf{R}_{\mathsf{ccr}}\mathsf{S}}(\mathcal{A}_h)$$

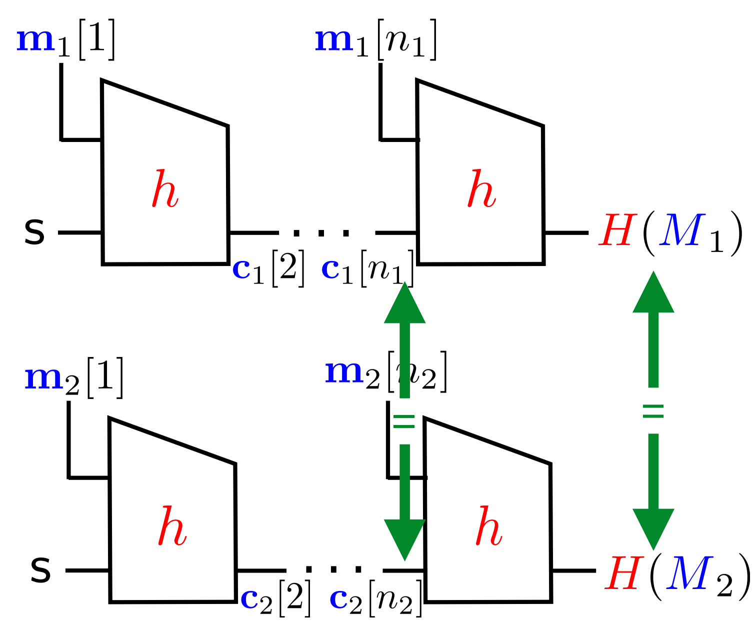
The time complexity of $\mathcal{A}_h$ is approximately that of $\mathcal{A}_H$ plus the time to compute H. The memory complexity of $\mathcal{A}_h$ is the maximum of the memory complexity of $\mathcal{A}_H$ and term linear in the length of the output of $\mathcal{A}_H$.

adversary $\mathcal{A}_h(\mathsf{s})$

$(M_1, M_2) \leftarrow \mathcal{A}_H(\mathsf{s}, \varepsilon)$
$\mathbf{m}_1 \leftarrow \mathsf{Split}(M_1)$; $\mathbf{m}_2 \leftarrow \mathsf{Split}(M_2)$; $n_1 \leftarrow |\mathbf{m}_1|$; $n_2 \leftarrow |\mathbf{m}_2|$
$\mathbf{c}_1[1] \leftarrow \mathsf{s}$; $\mathbf{c}_2[1] \leftarrow \mathsf{s}$
For $i = 1, \ldots, n_1$ do $\mathbf{c}_1[i+1] \leftarrow h(\mathbf{m}_1[i], \mathbf{c}_1[i])$
For $i = 1, \ldots, n_2$ do $\mathbf{c}_2[i+1] \leftarrow h(\mathbf{m}_2[i], \mathbf{c}_2[i])$
$b \leftarrow \mathrm{argmin}_d(n_d)$
For $i = 0, \ldots, n_b - 2$ do
  $(m_1, c_1) \leftarrow (\mathbf{m}_1[n_1 - i], \mathbf{c}_1[n_1 - i])$
  $(m_2, c_2) \leftarrow (\mathbf{m}_2[n_2 - i], \mathbf{c}_2[n_2 - i])$
  $a_1 \leftarrow (\mathbf{m}_1[n_1 - i - 1], \mathbf{c}_1[n_1 - i - 1])$
  $a_2 \leftarrow (\mathbf{m}_2[n_2 - i - 1], \mathbf{c}_2[n_2 - i - 1])$
  If $(m_1, c_1) \neq (m_2, c_2)$ then
    Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$
If $n_1 = n_2$ then
  $(m_1, c_1) \leftarrow (\mathbf{m}_1[1], \mathbf{c}_1[1])$; $(m_2, c_2) \leftarrow (\mathbf{m}_2[1], \mathbf{c}_2[1])$
  $a_1 \leftarrow 1$; $a_2 \leftarrow 2$
  Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$
$(m_1, c_1) \leftarrow (\mathbf{m}_1[n_1 - n_b + 1], \mathbf{c}_1[n_1 - n_b + 1])$
$(m_2, c_2) \leftarrow (\mathbf{m}_2[n_2 - n_b + 1], \mathbf{c}_2[n_2 - n_b + 1])$
$a_{3-b} \leftarrow (\mathbf{m}_{3-b}[n_{3-b} - n_b], \mathbf{c}_{3-b}[n_{3-b} - n_b])$
$a_b \leftarrow a_{3-b}$
Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$

Proof uses the back-tracking paradigm of [Me,Da] but constructs a CCR-violating adversary rather than a CR-violating one.





Theorem
Let Split be a suffix-free splitting function. Given an adversary $\mathcal{A}_H$, we define $\mathcal{A}_h$ such that

$$\mathbf{Adv}_H^{\mathsf{cr}}(\mathcal{A}_H) \leq \mathbf{Adv}_h^{\mathsf{R_{ccr}S}}(\mathcal{A}_h)$$

The time complexity of $\mathcal{A}_h$ is approximately that of $\mathcal{A}_H$ plus the time to compute H. The memory complexity of $\mathcal{A}_h$ is the maximum of the memory complexity of $\mathcal{A}_H$ and term linear in the length of the output of $\mathcal{A}_H$.

adversary $\mathcal{A}_h(\mathsf{s})$

$(M_1, M_2) \leftarrow \mathcal{A}_H(\mathsf{s}, \varepsilon)$

$\mathbf{m}_1 \leftarrow \mathsf{Split}(M_1)$; $\mathbf{m}_2 \leftarrow \mathsf{Split}(M_2)$; $n_1 \leftarrow |\mathbf{m}_1|$; $n_2 \leftarrow |\mathbf{m}_2|$

$\mathbf{c}_1[1] \leftarrow \mathsf{s}$; $\mathbf{c}_2[1] \leftarrow \mathsf{s}$

For $i = 1, \ldots, n_1$ do $\mathbf{c}_1[i+1] \leftarrow h(\mathbf{m}_1[i], \mathbf{c}_1[i])$

For $i = 1, \ldots, n_2$ do $\mathbf{c}_2[i+1] \leftarrow h(\mathbf{m}_2[i], \mathbf{c}_2[i])$

$b \leftarrow \mathrm{argmin}_d(n_d)$

For $i = 0, \ldots, n_b - 2$ do

  $(m_1, c_1) \leftarrow (\mathbf{m}_1[n_1 - i], \mathbf{c}_1[n_1 - i])$

  $(m_2, c_2) \leftarrow (\mathbf{m}_2[n_2 - i], \mathbf{c}_2[n_2 - i])$

  $a_1 \leftarrow (\mathbf{m}_1[n_1 - i - 1], \mathbf{c}_1[n_1 - i - 1])$

  $a_2 \leftarrow (\mathbf{m}_2[n_2 - i - 1], \mathbf{c}_2[n_2 - i - 1])$

  If $(m_1, c_1) \neq (m_2, c_2)$ then

    Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$

If $n_1 = n_2$ then

  $(m_1, c_1) \leftarrow (\mathbf{m}_1[1], \mathbf{c}_1[1])$; $(m_2, c_2) \leftarrow (\mathbf{m}_2[1], \mathbf{c}_2[1])$

  $a_1 \leftarrow 1; a_2 \leftarrow 2$

  Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$

$(m_1, c_1) \leftarrow (\mathbf{m}_1[n_1 - n_b + 1], \mathbf{c}_1[n_1 - n_b + 1])$

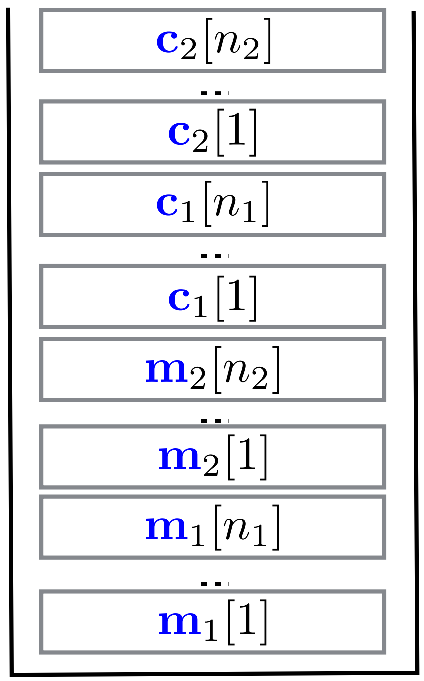
$(m_2, c_2) \leftarrow (\mathbf{m}_2[n_2 - n_b + 1], \mathbf{c}_2[n_2 - n_b + 1])$

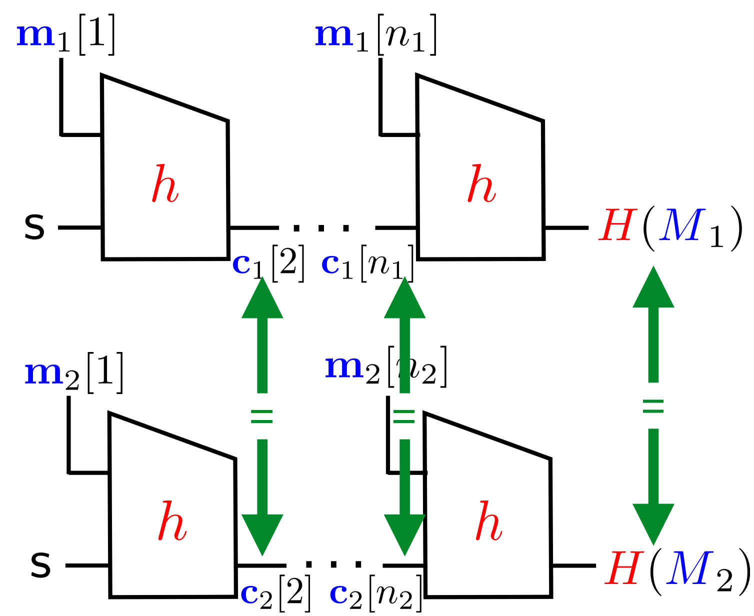
$a_{3-b} \leftarrow (\mathbf{m}_{3-b}[n_{3-b} - n_b], \mathbf{c}_{3-b}[n_{3-b} - n_b])$

$a_b \leftarrow a_{3-b}$

Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$

Proof uses the back-tracking paradigm of [Me,Da] but constructs a CCR-violating adversary rather than a CR-violating one.





Closer look at memory complexity

Theorem Same as above, except:
The memory complexity of $\mathcal{A}_h$ is the maximum of the memory complexity of $\mathcal{A}_H$ and a **small constant**.

```
adversary A_h(s)
(M_1, M_2) ← A_H(s, ε)
m_1 ← Split(M_1) ; m_2 ← Split(M_2) ; n_1 ← |m_1| ; n_2 ← |m_2|
c_1[1] ← s ; c_2[1] ← s; n ← min(n_1, n_2)
If (n_1 > n_2) then
   For i = 1, ..., n_1 − n_2 do c_1[i + 1] ← h(m_1[i], c_1[i])
If (n_2 > n_1) then
   For i = 1, ..., n_2 − n_1 do c_2[i + 1] ← h(m_2[i], c_2[i])
For i = 1, ..., n do
   m_1 ← m_1[n_1 − n + i]; c_1 ← c_1[n_1 − n + i]
   m_2 ← m_2[n_2 − n + i]; c_2 ← c_2[n_2 − n + i]
   c'_1 ← h(m_1, c_1)
   c'_2 ← h(m_2, c_2)
   If (c'_1 = c'_2) and (m_1, c_1) ≠ (m_2, c_2) then
      a_1 ← (m_1[n_1 − n + i − 1], c_1[n_1 − n + i − 1])
      a_2 ← (m_2[n_2 − n + i − 1], c_2[n_2 − n + i − 1])
      Return ((m_1, c_1), (m_2, c_2), a_1, a_2)
   c_1[n_1 − n + i + 1] ← c'_1
   c_2[n_2 − n + i + 1] ← c'_2
Return ⊥
```

ACFK17: “memory tightness is important”

Natural reduction was *not* memory tight.

Closer look at memory complexity

Theorem Same as above, except:
The memory complexity of $\mathcal{A}_h$ is the maximum of the memory complexity of $\mathcal{A}_H$ and a **small constant**.

adversary $\mathcal{A}_h(s)$

$(M_1, M_2) \leftarrow \mathcal{A}_H(s, \epsilon)$

$\mathbf{m}_1 \leftarrow \mathsf{Split}(M_1)$; $\mathbf{m}_2 \leftarrow \mathsf{Split}(M_2)$; $n_1 \leftarrow |\mathbf{m}_1|$; $n_2 \leftarrow |\mathbf{m}_2|$

$\mathbf{c}_1[1] \leftarrow s$; $\mathbf{c}_2[1] \leftarrow s$; $n \leftarrow \min(n_1, n_2)$

If $(n_1 > n_2)$ then

  For $i = 1, \ldots, n_1 - n_2$ do $\mathbf{c}_1[i+1] \leftarrow h(\mathbf{m}_1[i], \mathbf{c}_1[i])$

If $(n_2 > n_1)$ then

  For $i = 1, \ldots, n_2 - n_1$ do $\mathbf{c}_2[i+1] \leftarrow h(\mathbf{m}_2[i], \mathbf{c}_2[i])$

For $i = 1, \ldots, n$ do

  $m_1 \leftarrow \mathbf{m}_1[n_1 - n + i]$; $c_1 \leftarrow \mathbf{c}_1[n_1 - n + i]$

  $m_2 \leftarrow \mathbf{m}_2[n_2 - n + i]$; $c_2 \leftarrow \mathbf{c}_2[n_2 - n + i]$

  $c_1' \leftarrow h(m_1, c_1)$

  $c_2' \leftarrow h(m_2, c_2)$

  If $(c_1' = c_2')$ and $(m_1, c_1) \neq (m_2, c_2)$ then

    $a_1 \leftarrow (\mathbf{m}_1[n_1 - n + i - 1], \mathbf{c}_1[n_1 - n + i - 1])$

    $a_2 \leftarrow (\mathbf{m}_2[n_2 - n + i - 1], \mathbf{c}_2[n_2 - n + i - 1])$

    Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$

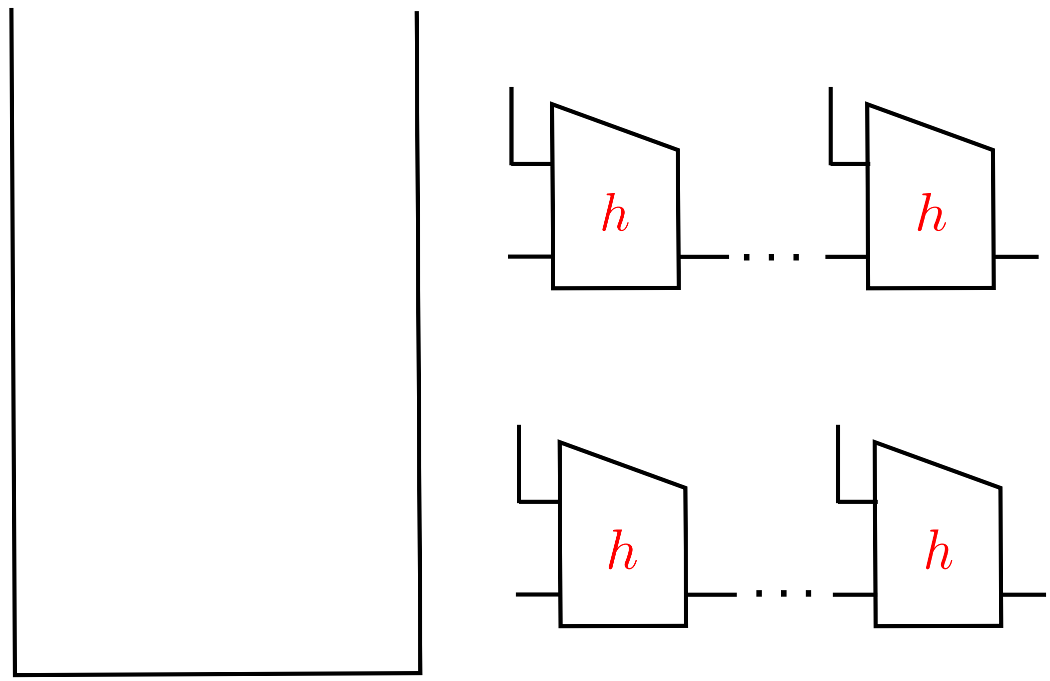
  $\mathbf{c}_1[n_1 - n + i + 1] \leftarrow c_1'$

  $\mathbf{c}_2[n_2 - n + i + 1] \leftarrow c_2'$

Return $\perp$

ACFK17: “memory tightness is important”

Natural reduction was *not* memory tight.



Closer look at memory complexity

<u>Theorem</u> Same as above, except:
The memory complexity of $\mathcal{A}_h$ is the maximum of the memory complexity of $\mathcal{A}_H$ and a **small constant**.

adversary $\mathcal{A}_h(s)$

$(M_1, M_2) \leftarrow \mathcal{A}_H(s, \epsilon)$

$\mathbf{m}_1 \leftarrow \mathsf{Split}(M_1)$; $\mathbf{m}_2 \leftarrow \mathsf{Split}(M_2)$; $n_1 \leftarrow |\mathbf{m}_1|$; $n_2 \leftarrow |\mathbf{m}_2|$

$\mathbf{c}_1[1] \leftarrow s$; $\mathbf{c}_2[1] \leftarrow s$; $n \leftarrow \min(n_1, n_2)$

If $(n_1 > n_2)$ then

  For $i = 1, \ldots, n_1 - n_2$ do $\mathbf{c}_1[i+1] \leftarrow h(\mathbf{m}_1[i], \mathbf{c}_1[i])$

If $(n_2 > n_1)$ then

  For $i = 1, \ldots, n_2 - n_1$ do $\mathbf{c}_2[i+1] \leftarrow h(\mathbf{m}_2[i], \mathbf{c}_2[i])$

For $i = 1, \ldots, n$ do

  $m_1 \leftarrow \mathbf{m}_1[n_1 - n + i]$; $c_1 \leftarrow \mathbf{c}_1[n_1 - n + i]$

  $m_2 \leftarrow \mathbf{m}_2[n_2 - n + i]$; $c_2 \leftarrow \mathbf{c}_2[n_2 - n + i]$

  $c_1' \leftarrow h(m_1, c_1)$

  $c_2' \leftarrow h(m_2, c_2)$

  If $(c_1' = c_2')$ and $(m_1, c_1) \neq (m_2, c_2)$ then

    $a_1 \leftarrow (\mathbf{m}_1[n_1 - n + i - 1], \mathbf{c}_1[n_1 - n + i - 1])$

    $a_2 \leftarrow (\mathbf{m}_2[n_2 - n + i - 1], \mathbf{c}_2[n_2 - n + i - 1])$

    Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$

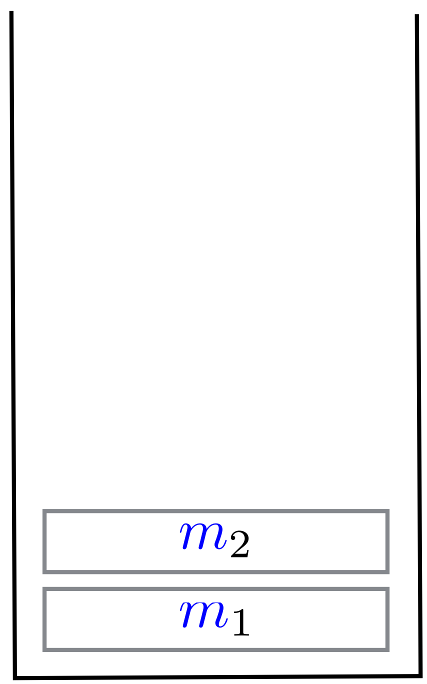
  $\mathbf{c}_1[n_1 - n + i + 1] \leftarrow c_1'$

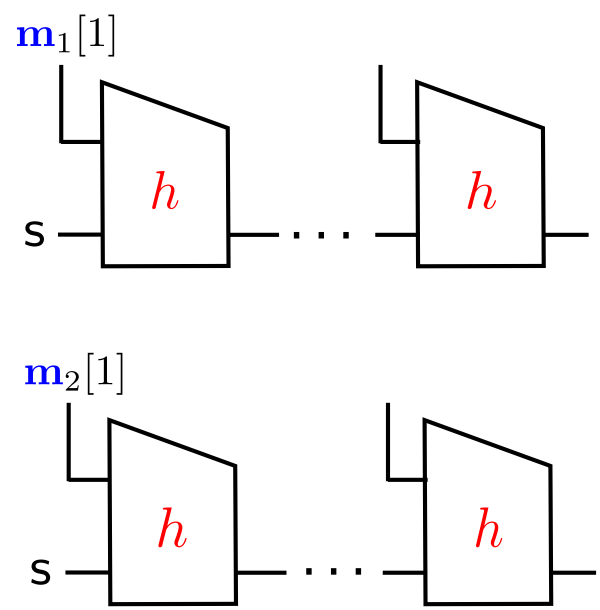
  $\mathbf{c}_2[n_2 - n + i + 1] \leftarrow c_2'$

Return $\perp$

ACFK17: "memory tightness is important"

Natural reduction was *not* memory tight.





Closer look at memory complexity

Theorem Same as above, except:
The memory complexity of $\mathcal{A}_h$ is the maximum of the memory complexity of $\mathcal{A}_H$ and a **small constant**.

adversary $\mathcal{A}_h(s)$

$(M_1, M_2) \leftarrow \mathcal{A}_H(s, \epsilon)$

$\mathbf{m}_1 \leftarrow \mathsf{Split}(M_1)$; $\mathbf{m}_2 \leftarrow \mathsf{Split}(M_2)$; $n_1 \leftarrow |\mathbf{m}_1|$; $n_2 \leftarrow |\mathbf{m}_2|$

$\mathbf{c}_1[1] \leftarrow s$; $\mathbf{c}_2[1] \leftarrow s$; $n \leftarrow \min(n_1, n_2)$

If $(n_1 > n_2)$ then

  For $i = 1, \ldots, n_1 - n_2$ do $\mathbf{c}_1[i+1] \leftarrow h(\mathbf{m}_1[i], \mathbf{c}_1[i])$

If $(n_2 > n_1)$ then

  For $i = 1, \ldots, n_2 - n_1$ do $\mathbf{c}_2[i+1] \leftarrow h(\mathbf{m}_2[i], \mathbf{c}_2[i])$

For $i = 1, \ldots, n$ do

  $m_1 \leftarrow \mathbf{m}_1[n_1 - n + i]$; $c_1 \leftarrow \mathbf{c}_1[n_1 - n + i]$

  $m_2 \leftarrow \mathbf{m}_2[n_2 - n + i]$; $c_2 \leftarrow \mathbf{c}_2[n_2 - n + i]$

  $c_1' \leftarrow h(m_1, c_1)$

  $c_2' \leftarrow h(m_2, c_2)$

  If $(c_1' = c_2')$ and $(m_1, c_1) \neq (m_2, c_2)$ then

    $a_1 \leftarrow (\mathbf{m}_1[n_1 - n + i - 1], \mathbf{c}_1[n_1 - n + i - 1])$

    $a_2 \leftarrow (\mathbf{m}_2[n_2 - n + i - 1], \mathbf{c}_2[n_2 - n + i - 1])$

    Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$

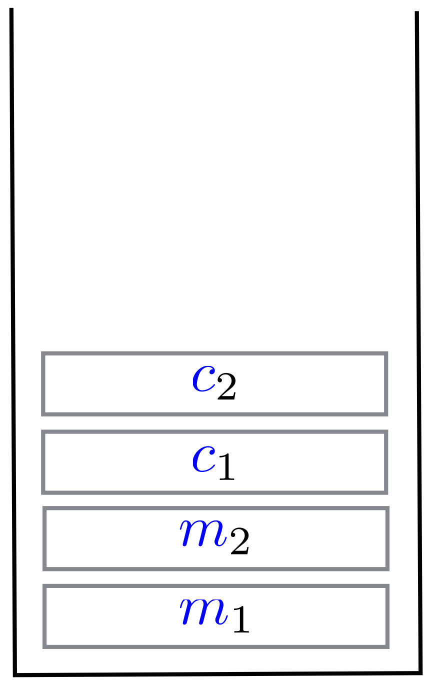
  $\mathbf{c}_1[n_1 - n + i + 1] \leftarrow c_1'$

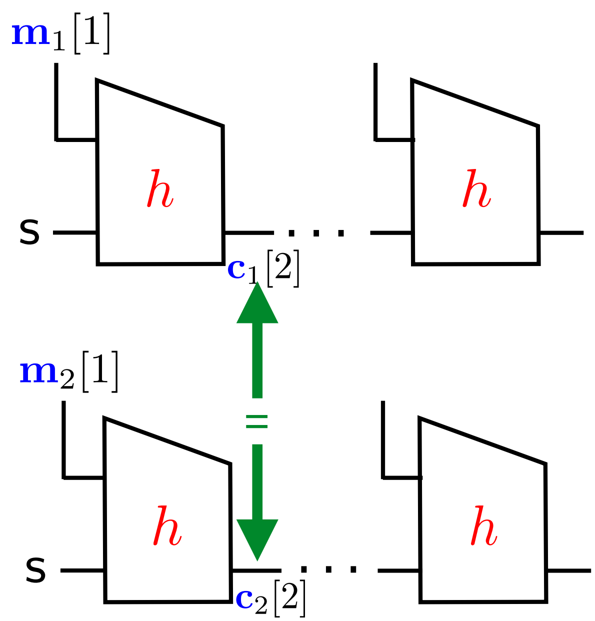
  $\mathbf{c}_2[n_2 - n + i + 1] \leftarrow c_2'$

Return $\perp$

ACFK17: "memory tightness is important"

Natural reduction was *not* memory tight.





Closer look at memory complexity

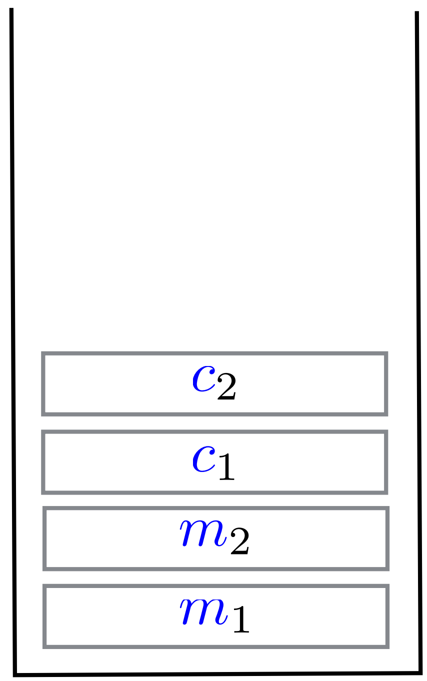
Theorem Same as above, except:
The memory complexity of $\mathcal{A}_h$ is the maximum of the memory complexity of $\mathcal{A}_H$ and a **small constant**.

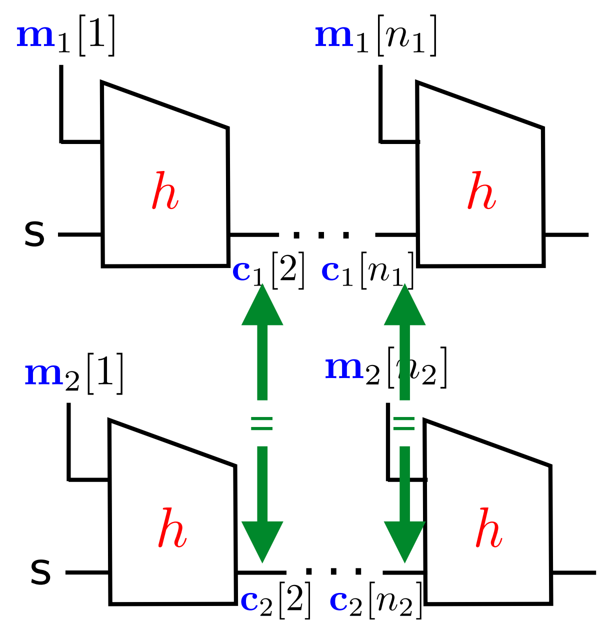
adversary $\mathcal{A}_h(s)$

$(M_1, M_2) \leftarrow \mathcal{A}_H(s, \epsilon)$
$\mathbf{m}_1 \leftarrow \mathsf{Split}(M_1)$; $\mathbf{m}_2 \leftarrow \mathsf{Split}(M_2)$; $n_1 \leftarrow |\mathbf{m}_1|$; $n_2 \leftarrow |\mathbf{m}_2|$
$\mathbf{c}_1[1] \leftarrow s$; $\mathbf{c}_2[1] \leftarrow s$; $n \leftarrow \min(n_1, n_2)$
If $(n_1 > n_2)$ then
  For $i = 1, \ldots, n_1 - n_2$ do $\mathbf{c}_1[i+1] \leftarrow h(\mathbf{m}_1[i], \mathbf{c}_1[i])$
If $(n_2 > n_1)$ then
  For $i = 1, \ldots, n_2 - n_1$ do $\mathbf{c}_2[i+1] \leftarrow h(\mathbf{m}_2[i], \mathbf{c}_2[i])$
For $i = 1, \ldots, n$ do
  $m_1 \leftarrow \mathbf{m}_1[n_1 - n + i]$; $c_1 \leftarrow \mathbf{c}_1[n_1 - n + i]$
  $m_2 \leftarrow \mathbf{m}_2[n_2 - n + i]$; $c_2 \leftarrow \mathbf{c}_2[n_2 - n + i]$
  $c_1' \leftarrow h(m_1, c_1)$
  $c_2' \leftarrow h(m_2, c_2)$
  If $(c_1' = c_2')$ and $(m_1, c_1) \neq (m_2, c_2)$ then
    $a_1 \leftarrow (\mathbf{m}_1[n_1 - n + i - 1], \mathbf{c}_1[n_1 - n + i - 1])$
    $a_2 \leftarrow (\mathbf{m}_2[n_2 - n + i - 1], \mathbf{c}_2[n_2 - n + i - 1])$
    Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$
  $\mathbf{c}_1[n_1 - n + i + 1] \leftarrow c_1'$
  $\mathbf{c}_2[n_2 - n + i + 1] \leftarrow c_2'$
Return $\perp$

ACFK17: "memory tightness is important"

Natural reduction was *not* memory tight.





Closer look at memory complexity

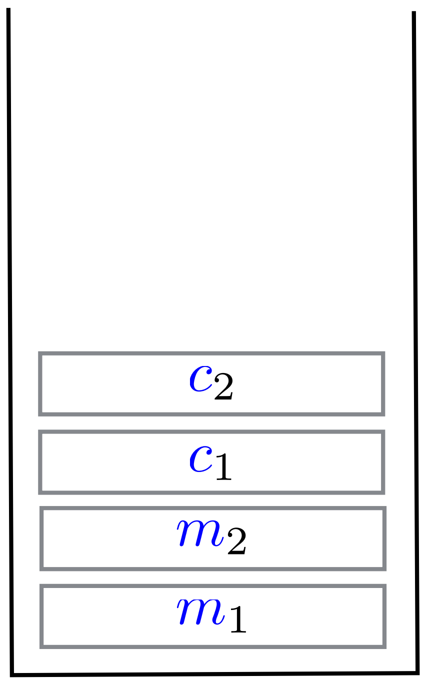
Theorem Same as above, except:
The memory complexity of $\mathcal{A}_h$ is the maximum of the memory complexity of $\mathcal{A}_H$ and a **small constant**.

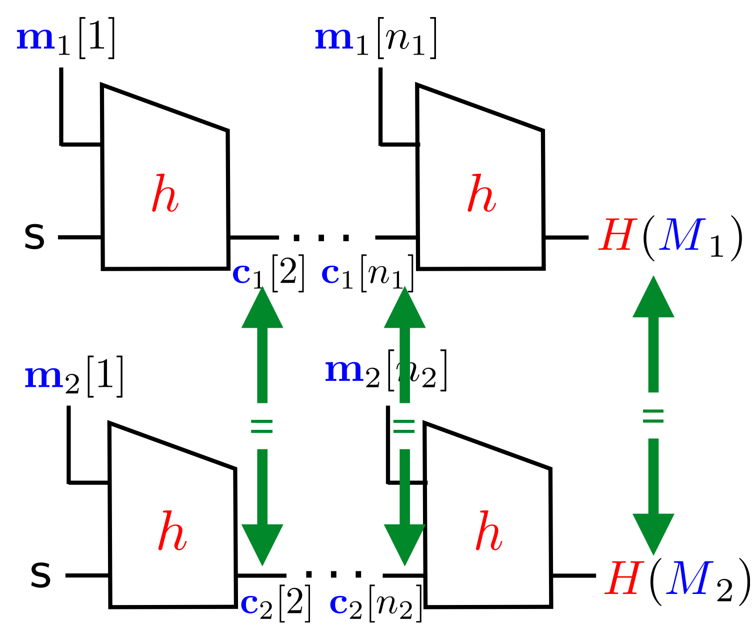
adversary $\mathcal{A}_h(s)$

$(M_1, M_2) \leftarrow \mathcal{A}_H(s, \epsilon)$
$\mathbf{m}_1 \leftarrow \mathsf{Split}(M_1)$; $\mathbf{m}_2 \leftarrow \mathsf{Split}(M_2)$; $n_1 \leftarrow |\mathbf{m}_1|$; $n_2 \leftarrow |\mathbf{m}_2|$
$\mathbf{c}_1[1] \leftarrow s$; $\mathbf{c}_2[1] \leftarrow s$; $n \leftarrow \min(n_1, n_2)$
If $(n_1 > n_2)$ then
  For $i = 1, \ldots, n_1 - n_2$ do $\mathbf{c}_1[i+1] \leftarrow h(\mathbf{m}_1[i], \mathbf{c}_1[i])$
If $(n_2 > n_1)$ then
  For $i = 1, \ldots, n_2 - n_1$ do $\mathbf{c}_2[i+1] \leftarrow h(\mathbf{m}_2[i], \mathbf{c}_2[i])$
For $i = 1, \ldots, n$ do
  $m_1 \leftarrow \mathbf{m}_1[n_1 - n + i]$; $c_1 \leftarrow \mathbf{c}_1[n_1 - n + i]$
  $m_2 \leftarrow \mathbf{m}_2[n_2 - n + i]$; $c_2 \leftarrow \mathbf{c}_2[n_2 - n + i]$
  $c_1' \leftarrow h(m_1, c_1)$
  $c_2' \leftarrow h(m_2, c_2)$
  If $(c_1' = c_2')$ and $(m_1, c_1) \neq (m_2, c_2)$ then
    $a_1 \leftarrow (\mathbf{m}_1[n_1 - n + i - 1], \mathbf{c}_1[n_1 - n + i - 1])$
    $a_2 \leftarrow (\mathbf{m}_2[n_2 - n + i - 1], \mathbf{c}_2[n_2 - n + i - 1])$
    Return $((m_1, c_1), (m_2, c_2), a_1, a_2)$
  $\mathbf{c}_1[n_1 - n + i + 1] \leftarrow c_1'$
  $\mathbf{c}_2[n_2 - n + i + 1] \leftarrow c_2'$
Return $\perp$

ACFK17: "memory tightness is important"

Natural reduction was *not* memory tight.





CCR is strictly weaker than CR

We show this by defining a CCR but not CR secure compression function:

$$h : \{0,1\}^{h.ml} \times \{0,1\}^{h.cl} \to \{0,1\}^{h.cl}$$

Assumptions

1. Split is suffix-free
2. h has access to a CR function $h' : \{0,1\}^{h.ml} \times \{0,1\}^{h.cl} \to \{0,1\}^{h.cl-1}$
3. $\mathsf{S} = \{0,1\}^{h.cl} \setminus \{1\|0^{h.cl-1}, 1^2\|0^{h.cl-2}\}$

Claims

1. h is CCR
2. h is **not** CR
3. $H = \mathrm{MD}[h, \mathsf{Split}, \mathsf{S}]$ **is** CR

$h(m, c)$

If $(m, c) \in \{(0^{h.ml}, 1 \| 0^{h.cl-1}), (1^{h.ml}, 1^2 \| 0^{h.cl-2})\}$
    Return $1^{h.cl}$
Return $0 \| h'(m, c)$

CCR is strictly weaker than CR

We show this by defining a CCR but not CR secure compression function:

$$h : \{0,1\}^{h.ml} \times \{0,1\}^{h.cl} \rightarrow \{0,1\}^{h.cl}$$

Assumptions

1. Split is suffix-free
2. h has access to a CR function $h' : \{0,1\}^{h.ml} \times \{0,1\}^{h.cl} \rightarrow \{0,1\}^{h.cl-1}$
3. $\mathsf{S} = \{0,1\}^{h.cl} \setminus \{1\|0^{h.cl-1}, 1^2\|0^{h.cl-2}\}$

Claims

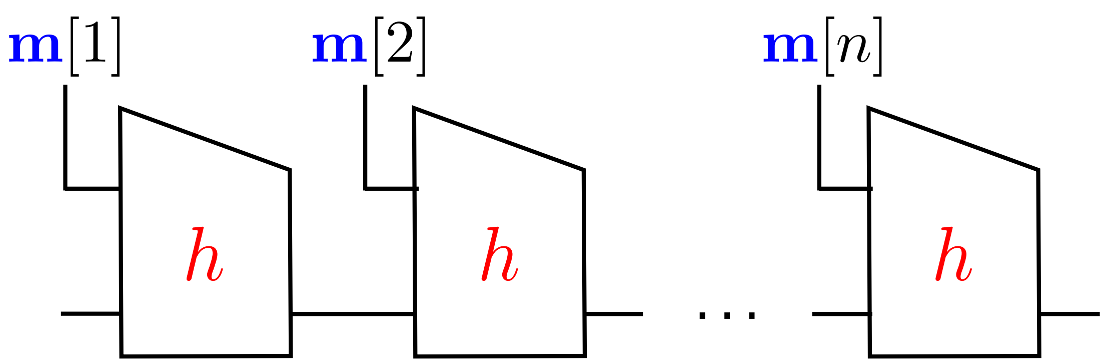
1. h is CCR
2. h is **not** CR
3. $H = \mathbf{MD}[h, \mathsf{Split}, \mathsf{S}]$ **is** CR

$h(m, c)$

If $(m, c) \in \{(0^{h.ml}, 1 \| 0^{h.cl-1}), (1^{h.ml}, 1^2 \| 0^{h.cl-2})\}$

  Return $1^{h.cl}$

Return $0 \| h'(m, c)$



CCR is strictly weaker than CR

We show this by defining a CCR but not CR secure compression function:

$$h : \{0,1\}^{h.ml} \times \{0,1\}^{h.cl} \rightarrow \{0,1\}^{h.cl}$$

Assumptions

1. Split is suffix-free
2. h has access to a CR function $h' : \{0,1\}^{h.ml} \times \{0,1\}^{h.cl} \rightarrow \{0,1\}^{h.cl-1}$
3. $\mathsf{S} = \{0,1\}^{h.cl} \setminus \{1 \| 0^{h.cl-1}, 1^2 \| 0^{h.cl-2}\}$

Claims

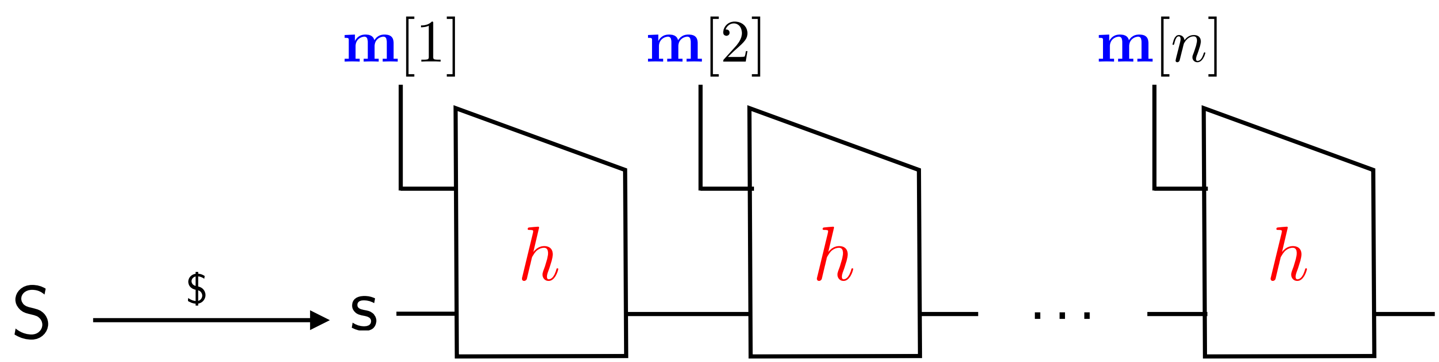
1. h is CCR
2. h is **not** CR
3. $H = \mathbf{MD}[h, \mathsf{Split}, \mathsf{S}]$ **is** CR

$h(m, c)$

If $(m, c) \in \{(0^{h.ml}, 1 \,\|\, 0^{h.cl-1}), (1^{h.ml}, 1^2 \,\|\, 0^{h.cl-2})\}$

  Return $1^{h.cl}$

Return $0 \,\|\, h'(m, c)$



CCR is strictly weaker than CR

We show this by defining a CCR but not CR secure compression function:

$$h : \{0,1\}^{h.ml} \times \{0,1\}^{h.cl} \rightarrow \{0,1\}^{h.cl}$$

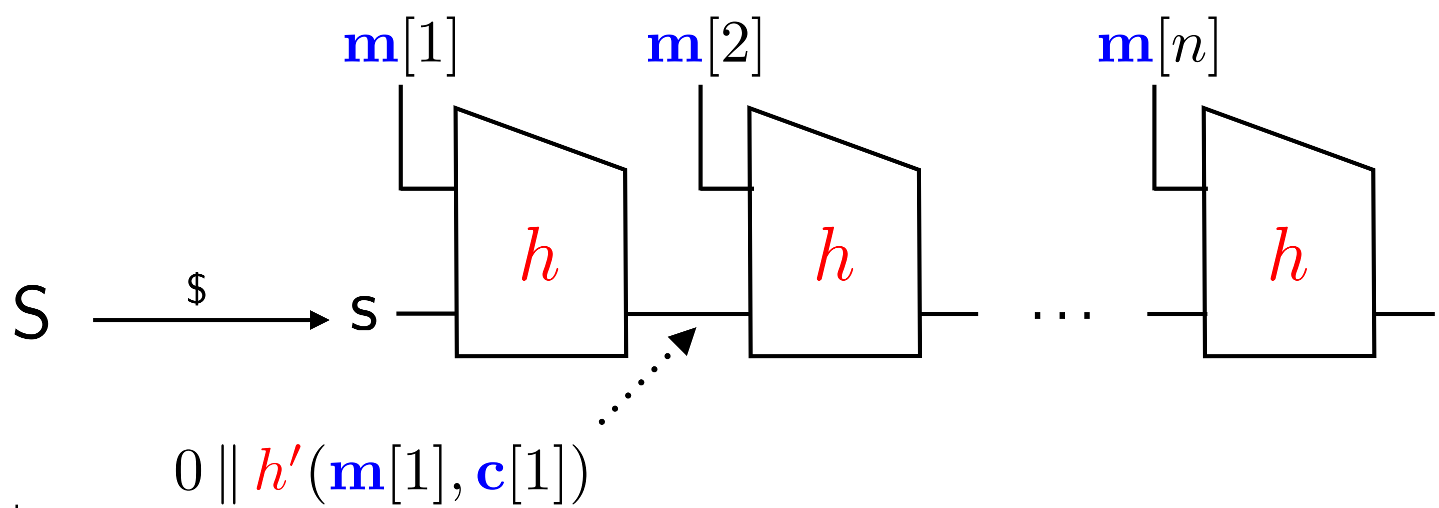
Assumptions

1. Split is suffix-free
2. h has access to a CR function $h' : \{0,1\}^{h.ml} \times \{0,1\}^{h.cl} \rightarrow \{0,1\}^{h.cl-1}$
3. $\mathsf{S} = \{0,1\}^{h.cl} \setminus \{1\|0^{h.cl-1}, 1^2\|0^{h.cl-2}\}$

Claims

1. h is CCR
2. h is **not** CR
3. $H = \mathbf{MD}[h, \mathsf{Split}, \mathsf{S}]$ **is** CR

$\underline{h(m, c)}$
If $(m, c) \in \{(0^{h.ml}, 1 \,\|\, 0^{h.cl-1}), (1^{h.ml}, 1^2 \,\|\, 0^{h.cl-2})\}$
  Return $1^{h.cl}$
Return $0 \,\|\, h'(m, c)$



CCR is strictly weaker than CR

We show this by defining a CCR but not CR secure compression function:

$$h : \{0,1\}^{h.ml} \times \{0,1\}^{h.cl} \rightarrow \{0,1\}^{h.cl}$$

Assumptions

1. Split is suffix-free
2. h has access to a CR function $h' : \{0,1\}^{h.ml} \times \{0,1\}^{h.cl} \rightarrow \{0,1\}^{h.cl-1}$
3. $\mathsf{S} = \{0,1\}^{h.cl} \setminus \{1\|0^{h.cl-1}, 1^2\|0^{h.cl-2}\}$

Claims

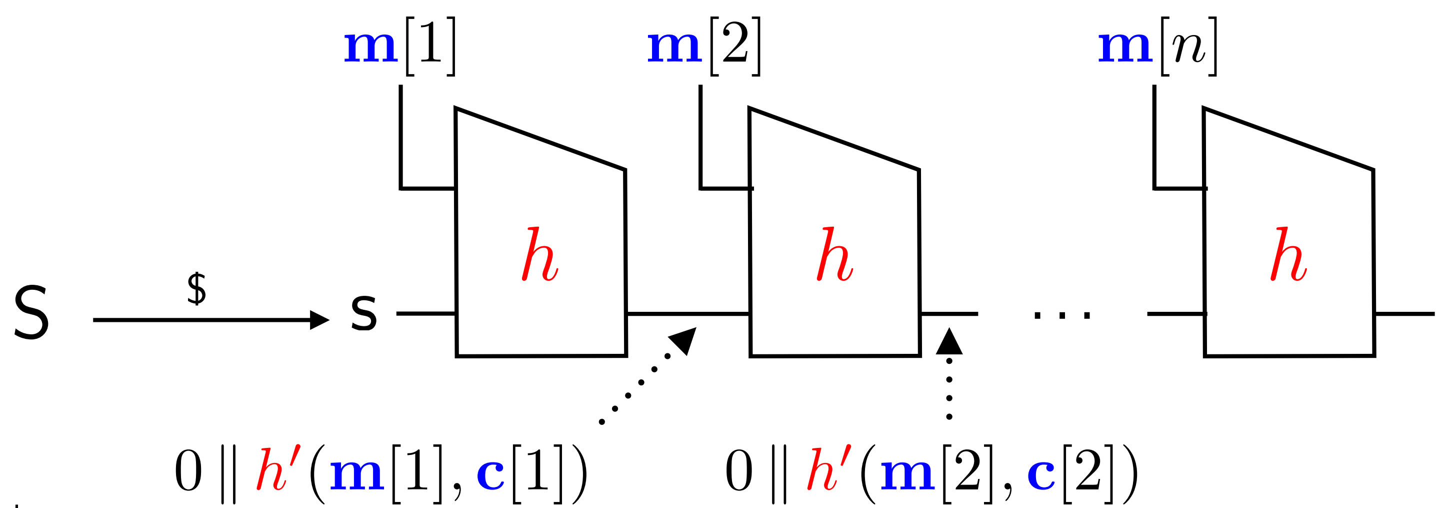
1. h is CCR
2. h is **not** CR
3. $H = \mathbf{MD}[h, \mathsf{Split}, \mathsf{S}]$ **is** CR

$h(m, c)$

If $(m, c) \in \{(0^{h.ml}, 1 \| 0^{h.cl-1}), (1^{h.ml}, 1^2 \| 0^{h.cl-2})\}$

  Return $1^{h.cl}$

Return $0 \| h'(m, c)$



CCR is strictly weaker than CR

We show this by defining a CCR but not CR secure compression function:

$$h : \{0,1\}^{h.ml} \times \{0,1\}^{h.cl} \to \{0,1\}^{h.cl}$$

Assumptions

1. Split is suffix-free
2. h has access to a CR function $h' : \{0,1\}^{h.ml} \times \{0,1\}^{h.cl} \to \{0,1\}^{h.cl-1}$
3. $\mathsf{S} = \{0,1\}^{h.cl} \setminus \{1\|0^{h.cl-1}, 1^2\|0^{h.cl-2}\}$

Claims

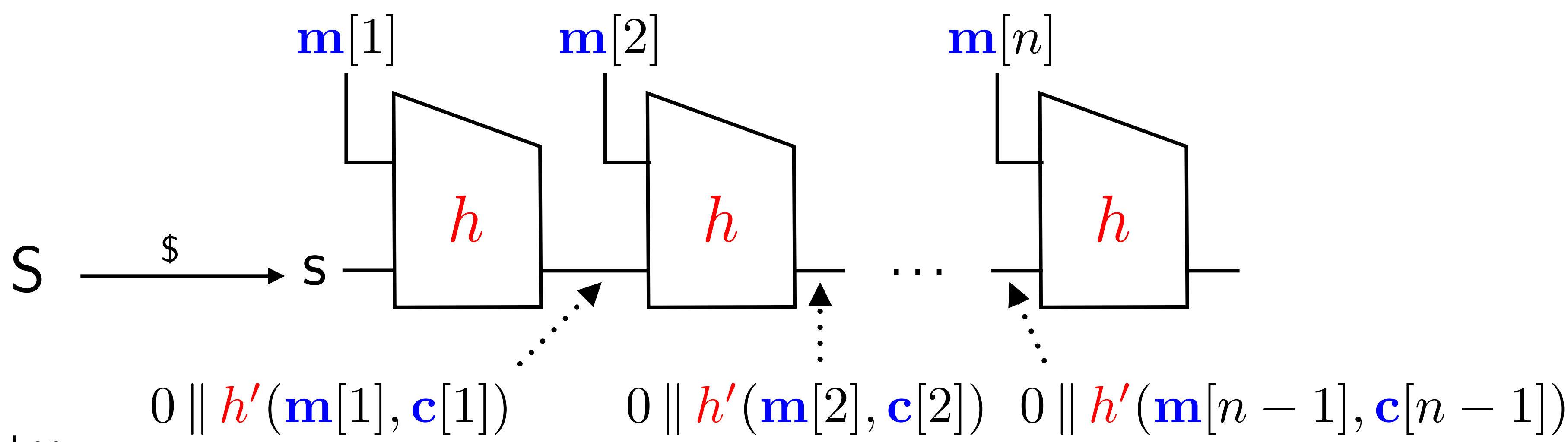
1. h is CCR
2. h is **not** CR
3. $H = \mathbf{MD}[h, \mathsf{Split}, \mathsf{S}]$ **is** CR

$\underline{h(m, c)}$

If $(m, c) \in \{(0^{h.ml}, 1 \,\|\, 0^{h.cl-1}), (1^{h.ml}, 1^2 \,\|\, 0^{h.cl-2})\}$

  Return $1^{h.cl}$

Return $0 \,\|\, h'(m, c)$



CCR is strictly weaker than CR

We show this by defining a CCR but not CR secure compression function:

$$h : \{0,1\}^{h.ml} \times \{0,1\}^{h.cl} \to \{0,1\}^{h.cl}$$

Assumptions

1. Split is suffix-free
2. h has access to a CR function $h' : \{0,1\}^{h.ml} \times \{0,1\}^{h.cl} \to \{0,1\}^{h.cl-1}$
3. $\mathsf{S} = \{0,1\}^{h.cl} \setminus \{1\|0^{h.cl-1}, 1^2\|0^{h.cl-2}\}$

Claims

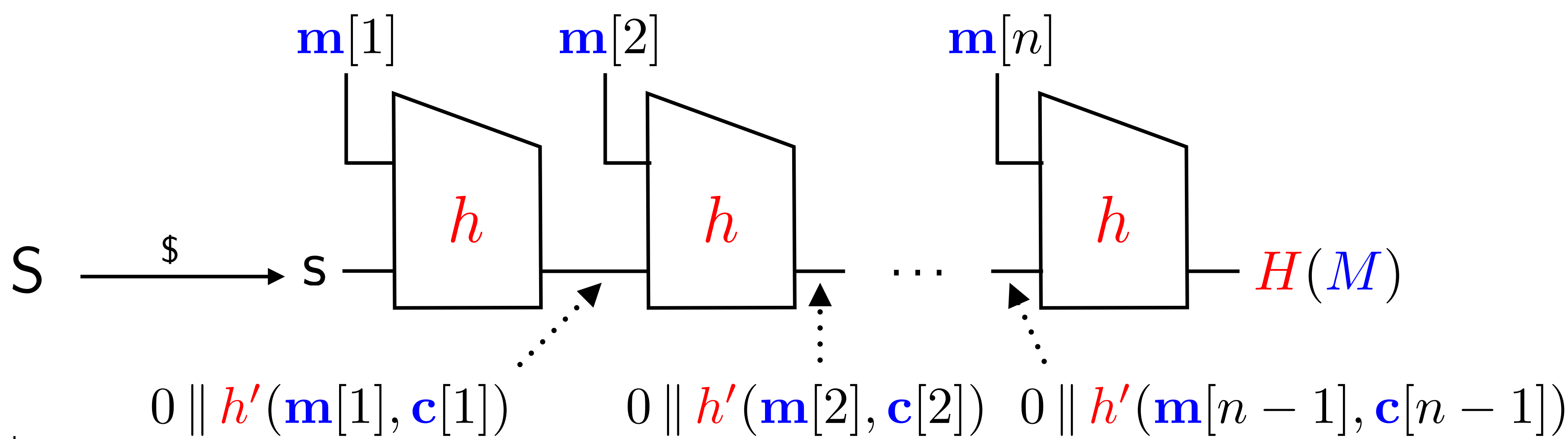
1. h is CCR
2. h is **not** CR
3. $H = \mathbf{MD}[h, \mathsf{Split}, \mathsf{S}]$ **is** CR

$h(m, c)$

If $(m, c) \in \{(0^{h.ml}, 1 \| 0^{h.cl-1}), (1^{h.ml}, 1^2 \| 0^{h.cl-2})\}$

  Return $1^{h.cl}$

Return $0 \| h'(m, c)$



Speeding up MD

Recall: using an injective splitting function could potentially save an extra call to h. This could lead to efficiency gains in the performance of the MD transform.

Theorem

Let Split be an injective splitting function. Given an adversary $\mathcal{A}_H$ we define adversaries $\mathcal{A}_h$ and $\mathcal{B}_h$ such that

$$\mathbf{Adv}_H^{\mathsf{cr}}(\mathcal{A}_H) \leq \mathbf{Adv}_h^{\mathsf{R_{ccr}S}}(\mathcal{A}_h) + \mathbf{Adv}_h^{\mathsf{R_{pre}S}}(\mathcal{B}_h)$$

The time complexities of $\mathcal{A}_h$ and $\mathcal{B}_h$ are that of $\mathcal{A}_H$ plus the time to compute H on its output. The memory complexities of $\mathcal{A}_h$ and $\mathcal{B}_h$ are the maximum of that of $\mathcal{A}_H$ and a small constant.

[AnSt11] informally state similar result for CR.

Speeding up MD

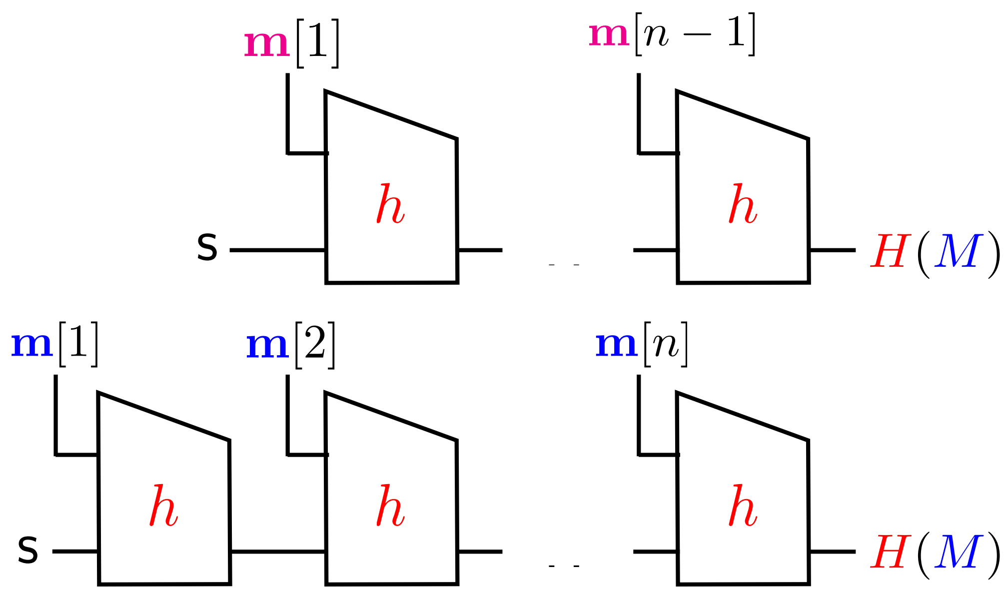
Recall: using an injective splitting function could potentially save an extra call to h. This could lead to efficiency gains in the performance of the MD transform.

Theorem

Let Split be an injective splitting function. Given an adversary $\mathcal{A}_H$ we define adversaries $\mathcal{A}_h$ and $\mathcal{B}_h$ such that

$$\mathbf{Adv}_H^{\mathsf{cr}}(\mathcal{A}_H) \leq \mathbf{Adv}_h^{\mathsf{R_{ccr}S}}(\mathcal{A}_h) + \mathbf{Adv}_h^{\mathsf{R_{pre}S}}(\mathcal{B}_h)$$

The time complexities of $\mathcal{A}_h$ and $\mathcal{B}_h$ are that of $\mathcal{A}_H$ plus the time to compute H on its output. The memory complexities of $\mathcal{A}_h$ and $\mathcal{B}_h$ are the maximum of that of $\mathcal{A}_H$ and a small constant.



Speeding up MD

Recall: using an injective splitting function could potentially save an extra call to h. This could lead to efficiency gains in the performance of the MD transform.

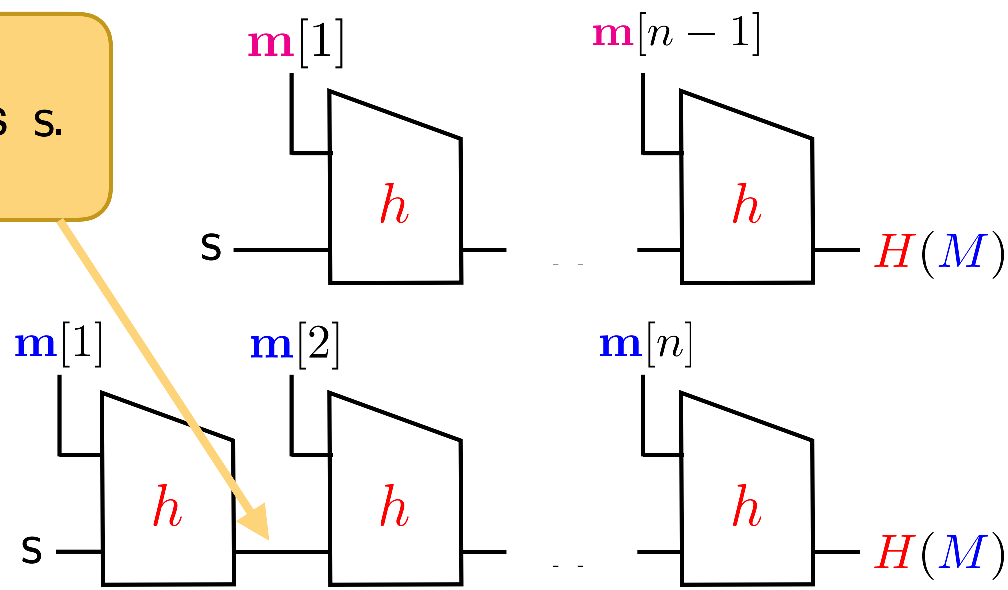
Theorem

Let Split be an injective splitting function. Given an adversary $\mathcal{A}_H$ we define adversaries $\mathcal{A}_h$ and $\mathcal{B}_h$ such that

$$\mathbf{Adv}_H^{\mathrm{cr}}(\mathcal{A}_H) \le \mathbf{Adv}_h^{\mathrm{R_{ccr}S}}(\mathcal{A}_h) + \mathbf{Adv}_h^{\mathrm{R_{pre}S}}(\mathcal{B}_h)$$

The time complexities of $\mathcal{A}_h$ and $\mathcal{B}_h$ are that of $\mathcal{A}_H$ plus the time to compute H on its output. The memory complexities of $\mathcal{A}_h$ and $\mathcal{B}_h$ are the maximum of that of $\mathcal{A}_H$ and a small constant.

Case 1: This is s.



Speeding up MD

Recall: using an injective splitting function could potentially save an extra call to h. This could lead to <u>efficiency gains</u> in the performance of the MD transform.

Theorem

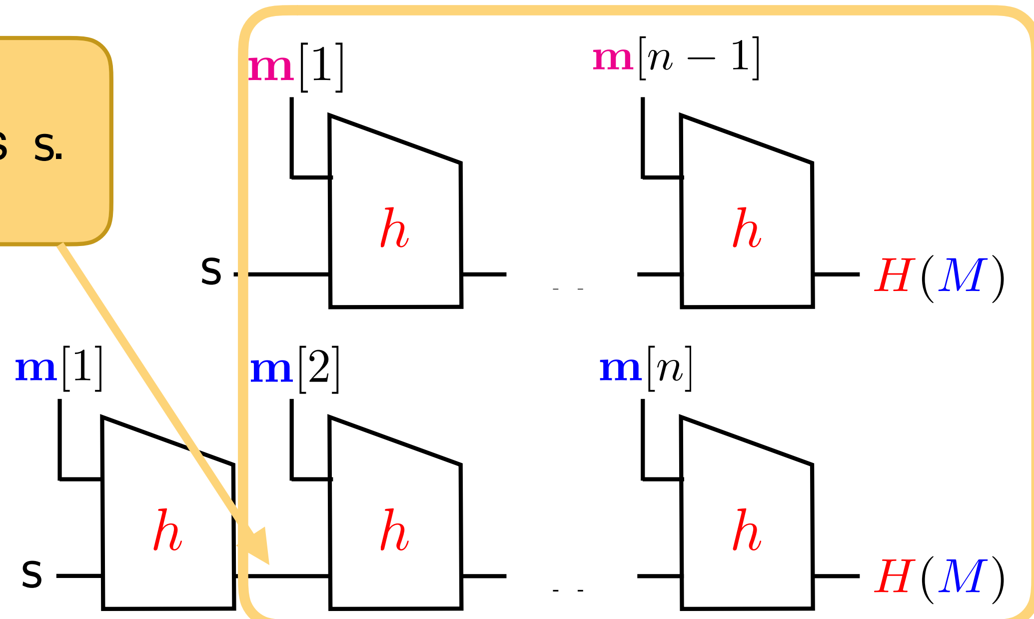
Let Split be an injective splitting function. Given an adversary $\mathcal{A}_H$ we define adversaries $\mathcal{A}_h$ and $\mathcal{B}_h$ such that

$$\mathbf{Adv}_H^{\mathsf{cr}}(\mathcal{A}_H) \leq \mathbf{Adv}_h^{\mathsf{R_{ccr}S}}(\mathcal{A}_h) + \mathbf{Adv}_h^{\mathsf{R_{pre}S}}(\mathcal{B}_h)$$

The time complexities of $\mathcal{A}_h$ and $\mathcal{B}_h$ are that of $\mathcal{A}_H$ plus the time to compute H on its output. The memory complexities of $\mathcal{A}_h$ and $\mathcal{B}_h$ are the maximum of that of $\mathcal{A}_H$ and a small constant.

Case 1: This is s.

Case 2: This is a collision in h somewhere here.



Summary

Summary

- We defined a framework for the MD transform that allows us to formalize results and unify and simplify the area.

Summary

- We defined a framework for the MD transform that allows us to formalize results and unify and simplify the area.
- We defined a new security property for compression functions called **constrained collision resistance** (CCR) and showed that a CCR compression function will result in a CR hash function.

Summary

- We defined a framework for the MD transform that allows us to formalize results and unify and simplify the area.
- We defined a new security property for compression functions called **constrained collision resistance** (CCR) and showed that a CCR compression function will result in a CR hash function.
- We defined the RS-security framework in order to describe classical definitions and specify new variants of definitions.

Summary

- We defined a framework for the MD transform that allows us to formalize results and unify and simplify the area.
- We defined a new security property for compression functions called **constrained collision resistance** (CCR) and showed that a CCR compression function will result in a CR hash function.
- We defined the RS-security framework in order to describe classical definitions and specify new variants of definitions.
- We looked at memory complexity by explicitly giving reductions. In addition, we gave alternate reduction algorithms that were more memory tight. This allows us to **more easily address memory complexity**.

Summary

- We defined a framework for the MD transform that allows us to formalize results and unify and simplify the area.
- We defined a new security property for compression functions called **constrained collision resistance** (CCR) and showed that a CCR compression function will result in a CR hash function.
- We defined the RS-security framework in order to describe classical definitions and specify new variants of definitions.
- We looked at memory complexity by explicitly giving reductions. In addition, we gave alternate reduction algorithms that were more memory tight. This allows us to **more easily address memory complexity**.
- We showed how the MD transform can be made **more efficient** by using an **injective splitting function**. In particular, if the splitting function is injective, the compression function is CCR, and it is hard to find a pre-image for s, then the hash function will be CR.