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

Mihir Bellare  Joseph Jaeger  Julia Len

UC San Diego
Hash Functions

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

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

any size

\[ M \rightarrow H \rightarrow H(M) \]

fixed size

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

Central usage: Certificates

\[ M \rightarrow H \rightarrow \text{Sign} \rightarrow \text{Signature} \]

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.

<table>
<thead>
<tr>
<th>Generation</th>
<th>(H)</th>
<th>(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>MD4, MD5</td>
<td>128</td>
</tr>
<tr>
<td>2nd</td>
<td>SHA-1, SHA-256,</td>
<td>160, 256, 512</td>
</tr>
<tr>
<td></td>
<td>SHA-512</td>
<td></td>
</tr>
<tr>
<td>3rd</td>
<td>SHA3-224, SHA3-256,</td>
<td>224, 256, 384, 512</td>
</tr>
<tr>
<td></td>
<td>SHA3-384, SHA3-512</td>
<td></td>
</tr>
</tbody>
</table>
Hash Functions

any size $M$ \rightarrow H \rightarrow \text{fixed size} \rightarrow H(M)

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

Central usage: Certificates

$M$ \rightarrow H \rightarrow Sign \rightarrow Signature

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

any size 

\[ M \rightarrow H \rightarrow H(M) \]

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

Central usage: Certificates

\[ M \rightarrow H \rightarrow \text{Sign} \rightarrow \text{Signature} \]

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

any size \( M \) → \( H \) → fixed size \( H(M) \)

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

Central usage: Certificates

\( M \) → \( H \) → Sign → Signature

secret key

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.

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

Central usage: Certificates

\( H(M) \)

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

Collisions in \( H \) lead to certificate forgery. SHA-1 collision leading to browsers no longer accepting SHA-1-based certificates.
How hash functions are built
How hash functions are built

Step 1: Design a compression function $h$

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

<table>
<thead>
<tr>
<th>$H$</th>
<th>$h.ml$</th>
<th>$h.cl$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD5</td>
<td>512</td>
<td>128</td>
</tr>
<tr>
<td>SHA-1</td>
<td>512</td>
<td>160</td>
</tr>
<tr>
<td>SHA-256</td>
<td>512</td>
<td>256</td>
</tr>
<tr>
<td>SHA-512</td>
<td>1024</td>
<td>512</td>
</tr>
</tbody>
</table>
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

\[
\begin{align*}
    m & \rightarrow h \\
    c & \rightarrow h \\
    h : \{0, 1\}^{h.ml + h.cl} & \rightarrow \{0, 1\}^{h.cl}
\end{align*}
\]
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

Classical Theorem: \([\text{Me},\text{Da}]\)  \( h \text{ CR} \Rightarrow H \text{ CR} \)
Step 1: Design a compression function $h$
Step 2: Convert $h$ into a CR hash function $H$ via the MD transform

Classical Theorem: $[\text{Me, Da}]$
$h$ CR $\Rightarrow H$ CR

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

- Corollary of Classical Theorem: $H$ not CR $\Rightarrow h$ 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 $\Rightarrow H$ CR

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

- Corollary of Classical Theorem: $H$ not CR $\Rightarrow h$ 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 $\Rightarrow H$ 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 $\Rightarrow$ $H$ CR

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

- Corollary of Classical Theorem: $H$ not CR $\Rightarrow$ $h$ 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 $\Rightarrow$ $H$ CR

For some choice of X that is WEAKER than CR.

**Our Answer:** YES, $X = CCR$

Constrained Collision-Resistance.
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

Our Theorem 1: $h$ CCR $\Rightarrow H$ CR
Our Theorem 2: There exist $h$ that are CCR but not CR

Classical Theorem: [Me, Da]
$h$ CR $\Rightarrow H$ 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

Our Theorem 1: $h$ CCR $\Rightarrow$ $H$ CR

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

Classical Theorem: [Me, Da] $h$ CR $\Rightarrow$ $H$ 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

Our Theorem 1: $h$ CCR $\Rightarrow H$ CR

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

Classical Theorem: [Me, Da]
$h$ CR $\Rightarrow H$ 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.
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 \( \Rightarrow \) \( H \) 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: \( H = \text{MD}[h, \text{Split}, S] \)
- RS Security framework: Yields both old and new definitions of security for \( h \)
Contributions

Our Theorem 1: $h$ CCR $\Rightarrow H$ 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: $H = \text{MD}[h, \text{Split}, S]$
- 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: \( h \text{ CCR} \Rightarrow H \text{ CR} \)

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

These results are obtained via a **general framework**
- Parameterized version of MD: \( H = \text{MD}[h, \text{Split}, S] \)
- 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
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 $X = \text{CCR}$ 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 …
We don’t design CCR compression functions. But existing candidates include the compression functions of SHA256, SHA512.

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

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.

For the result that: \( h \) is X-secure implies \( H \) is CR we said that \( X = \text{CCR} \) 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 …

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

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

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

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

<table>
<thead>
<tr>
<th>$H$</th>
<th>$h$</th>
<th>Split</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD5</td>
<td>md5</td>
<td>$M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M</td>
</tr>
<tr>
<td>SHA-1</td>
<td>sha1</td>
<td>$M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M</td>
</tr>
<tr>
<td>SHA-256</td>
<td>sha256</td>
<td>$M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M</td>
</tr>
<tr>
<td>SHA-512</td>
<td>sha512</td>
<td>$M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M</td>
</tr>
</tbody>
</table>
The MD Framework

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

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

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

<table>
<thead>
<tr>
<th>$H$</th>
<th>$h$</th>
<th>Split</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD5</td>
<td>md5</td>
<td>$M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M\rangle_{64}$</td>
</tr>
<tr>
<td>SHA-1</td>
<td>sha1</td>
<td>$M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M\rangle_{64}$</td>
</tr>
<tr>
<td>SHA-256</td>
<td>sha256</td>
<td>$M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M\rangle_{64}$</td>
</tr>
<tr>
<td>SHA-512</td>
<td>sha512</td>
<td>$M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M\rangle_{128}$</td>
</tr>
</tbody>
</table>
The MD Framework

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

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

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

### Table: Different Hash Functions

<table>
<thead>
<tr>
<th>$H$</th>
<th>$h$</th>
<th>Split</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD5</td>
<td>md5</td>
<td>$M | 1 | 0\ldots0 | \langle</td>
<td>M</td>
</tr>
<tr>
<td>SHA-1</td>
<td>sha1</td>
<td>$M | 1 | 0\ldots0 | \langle</td>
<td>M</td>
</tr>
<tr>
<td>SHA-256</td>
<td>sha256</td>
<td>$M | 1 | 0\ldots0 | \langle</td>
<td>M</td>
</tr>
<tr>
<td>SHA-512</td>
<td>sha512</td>
<td>$M | 1 | 0\ldots0 | \langle</td>
<td>M</td>
</tr>
</tbody>
</table>
### The MD Framework

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

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

**Diagram:**
- \( M \) is input
- \( \text{Split} \) function
- \( m[1] m[2] \ldots m[n] \) output

**Set of starting points:** \( S \subseteq \{0, 1\}^{h\.cl} \)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Split</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( H )</td>
<td>( h )</td>
<td>( \text{Split} )</td>
<td>( S )</td>
</tr>
<tr>
<td>MD5</td>
<td>md5</td>
<td>( M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M</td>
</tr>
<tr>
<td>SHA-1</td>
<td>sha1</td>
<td>( M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M</td>
</tr>
<tr>
<td>SHA-256</td>
<td>sha256</td>
<td>( M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M</td>
</tr>
<tr>
<td>SHA-512</td>
<td>sha512</td>
<td>( M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M</td>
</tr>
</tbody>
</table>
The MD Framework

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

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

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

<table>
<thead>
<tr>
<th>$H$</th>
<th>$h$</th>
<th>Split</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD5</td>
<td>md5</td>
<td>$M \parallel 1 \parallel 0 ... \parallel \langle</td>
<td>M</td>
</tr>
<tr>
<td>SHA-1</td>
<td>sha1</td>
<td>$M \parallel 1 \parallel 0 ... \parallel \langle</td>
<td>M</td>
</tr>
<tr>
<td>SHA-256</td>
<td>sha256</td>
<td>$M \parallel 1 \parallel 0 ... \parallel \langle</td>
<td>M</td>
</tr>
<tr>
<td>SHA-512</td>
<td>sha512</td>
<td>$M \parallel 1 \parallel 0 ... \parallel \langle</td>
<td>M</td>
</tr>
</tbody>
</table>
The MD Framework

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

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

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

<table>
<thead>
<tr>
<th>$H$</th>
<th>$h$</th>
<th>Split</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD5</td>
<td>md5</td>
<td>$M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M</td>
</tr>
<tr>
<td>SHA-1</td>
<td>sha1</td>
<td>$M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M</td>
</tr>
<tr>
<td>SHA-256</td>
<td>sha256</td>
<td>$M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M</td>
</tr>
<tr>
<td>SHA-512</td>
<td>sha512</td>
<td>$M \parallel 1 \parallel 0\ldots0 \parallel \langle</td>
<td>M</td>
</tr>
</tbody>
</table>
The MD Framework

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

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

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

<table>
<thead>
<tr>
<th>( H )</th>
<th>( h )</th>
<th>( \text{Split} )</th>
<th>( S )</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD5</td>
<td>md5</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>SHA-1</td>
<td>sha1</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>SHA-256</td>
<td>sha256</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>SHA-512</td>
<td>sha512</td>
<td>M</td>
<td></td>
</tr>
</tbody>
</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):

$$\text{Split}(M) \rightarrow m[1] \quad m[2] \quad m[3]$$

**Injective**

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

$$\text{Split}(M) \rightarrow m[1] \quad m[2]$$

Split $(M)$ is one block shorter, so hashing uses one less call to the compression function. **Faster!**
<table>
<thead>
<tr>
<th></th>
<th>To win, $A$ must find</th>
<th>such that</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR</td>
<td>$(m_1, c_1) \neq (m_2, c_2)$</td>
<td>$h(m_1, c_1) = h(m_2, c_2)$</td>
</tr>
<tr>
<td>CCR</td>
<td>$(m_1, c_1) \neq (m_2, c_2)$</td>
<td>$h(m_1, c_1) = h(m_2, c_2)$</td>
</tr>
<tr>
<td></td>
<td>$(m'_1, c'_1), (m'_2, c'_2)$</td>
<td>$c_1 \in {s, h(m'_1, c'_1)}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$c_2 \in {s, h(m'_2, c'_2)}$</td>
</tr>
<tr>
<td>Pre</td>
<td>$(m, c)$</td>
<td>$h(m, c) = s$</td>
</tr>
<tr>
<td></td>
<td><strong>To win, $\mathcal{A}$ must find</strong></td>
<td><strong>such that</strong></td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td><strong>CR</strong></td>
<td>$(m_1, c_1) \neq (m_2, c_2)$</td>
<td>$h(m_1, c_1) = h(m_2, c_2)$</td>
</tr>
<tr>
<td><strong>CCR</strong></td>
<td>$(m_1, c_1) \neq (m_2, c_2)$</td>
<td>$h(m_1, c_1) = h(m_2, c_2)$</td>
</tr>
<tr>
<td></td>
<td>$(m'_1, c'_1), (m'_2, c'_2)$</td>
<td>$c_1 \in {s, h(m'_1, c'_1)}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$c_2 \in {s, h(m'_2, c'_2)}$</td>
</tr>
<tr>
<td><strong>Pre</strong></td>
<td>$(m, c)$</td>
<td>$h(m, c) = s$</td>
</tr>
</tbody>
</table>

![Diagram](attachment:image.png)
<table>
<thead>
<tr>
<th></th>
<th>To win, $\mathcal{A}$ must find</th>
<th>such that</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CR</strong></td>
<td>$(m_1, c_1) \neq (m_2, c_2)$</td>
<td>$h(m_1, c_1) = h(m_2, c_2)$</td>
</tr>
<tr>
<td><strong>CCR</strong></td>
<td>$(m_1, c_1) \neq (m_2, c_2)$</td>
<td>$h(m_1, c_1) = h(m_2, c_2)$</td>
</tr>
<tr>
<td></td>
<td>$(m_1', c_1'), (m_2', c_2')$</td>
<td>$c_1 \in {s, h(m_1', c_1')}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$c_2 \in {s, h(m_2', c_2')}$</td>
</tr>
<tr>
<td><strong>Pre</strong></td>
<td>$(m, c)$</td>
<td>$h(m, c) = s$</td>
</tr>
</tbody>
</table>

Diagram:

```
\begin{align*}
& m_1' \\
& h \\
& c_1' \\
& m_2' \\
& h \\
& c_2' \\
& y \\
& m_1 \\
& h \\
& c_1 \\
& m_2 \\
& h \\
\end{align*}
```
<table>
<thead>
<tr>
<th></th>
<th>To win, $\mathcal{A}$ must find</th>
<th>such that</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CR</strong></td>
<td>$(m_1, c_1) \neq (m_2, c_2)$</td>
<td>$h(m_1, c_1) = h(m_2, c_2)$</td>
</tr>
<tr>
<td><strong>CCR</strong></td>
<td>$(m_1, c_1) \neq (m_2, c_2)$</td>
<td>$h(m_1, c_1) = h(m_2, c_2)$</td>
</tr>
<tr>
<td></td>
<td>$(m'_1, c'_1), (m'_2, c'_2)$</td>
<td>$c_1 \in {s, h(m'_1, c'_1)}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$c_2 \in {s, h(m'_2, c'_2)}$</td>
</tr>
<tr>
<td><strong>Pre</strong></td>
<td>$(m, c)$</td>
<td>$h(m, c) = s$</td>
</tr>
</tbody>
</table>

![Diagram](image)
<table>
<thead>
<tr>
<th></th>
<th>To win, $\mathcal{A}$ must find</th>
<th>such that</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CR</strong></td>
<td>$(m_1, c_1) \neq (m_2, c_2)$</td>
<td>$h(m_1, c_1) = h(m_2, c_2)$</td>
</tr>
<tr>
<td><strong>CCR</strong></td>
<td>$(m_1, c_1) \neq (m_2, c_2)$</td>
<td>$h(m_1, c_1) = h(m_2, c_2)$</td>
</tr>
<tr>
<td></td>
<td>$(m'_1, c'_1), (m'_2, c'_2)$</td>
<td>$c_1 \in {s, h(m'_1, c'_1)}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$c_2 \in {s, h(m'_2, c'_2)}$</td>
</tr>
<tr>
<td><strong>Pre</strong></td>
<td>$(m, c)$</td>
<td>$h(m, c) = s$</td>
</tr>
</tbody>
</table>

**Pre**

![Pre Diagram](image-url)
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

$$R : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{\text{true, false}\}$$

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

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

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

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

Return $R(s, out)$

$R(s, out)$

starting value

string that adversary outputs
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

$$R : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{\text{true, false}\}$$

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

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

<table>
<thead>
<tr>
<th>$R$</th>
<th>$out$</th>
<th>$R(s, out)$ returns true iff</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{cr}$</td>
<td>$((m_1, c_1), (m_2, c_2))$</td>
<td>$h(m_1, c_1) = h(m_2, c_2)$</td>
<td>Collision resistance</td>
</tr>
<tr>
<td>$R_{ccc}$</td>
<td>$((m_1, c_1), (m_2, c_2), ((m'_1, c'_1), (m'_2, c'_2)))$</td>
<td>$R_{cr}(\varepsilon, ((m_1, c_1), (m_2, c_2))) \wedge (c_1 \in {s, h(m'_1, c'_1)}) \wedge (c_2 \in {s, h(m'_2, c'_2)})$</td>
<td>Constrained CR</td>
</tr>
<tr>
<td>$R_{pre}$</td>
<td>$(m, c)$</td>
<td>$h(m, c) = s$</td>
<td>Pre-image resistance</td>
</tr>
</tbody>
</table>
Results

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

<table>
<thead>
<tr>
<th>If Split is</th>
<th>and ( h ) is</th>
<th>then ( H = MD[h,\text{Split},S] ) is</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Suffix-free</td>
<td>CR</td>
<td>CR</td>
<td>Known [Me,Da], reproved</td>
</tr>
<tr>
<td>2 Suffix-free</td>
<td>CCR</td>
<td>CR</td>
<td></td>
</tr>
<tr>
<td>3 Injective</td>
<td>CCR and Pre</td>
<td>CR</td>
<td>Folklore for CR and Pre [AnSt11]</td>
</tr>
</tbody>
</table>
Typically, $S = \{s\}$ is a singleton set.

<table>
<thead>
<tr>
<th>If Split is</th>
<th>and $h$ is</th>
<th>then $H = MD[h, \text{Split}, S]$ is</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Suffix-free</td>
<td>CR</td>
<td>CR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Known [Me, Da], reproved</td>
</tr>
<tr>
<td>2</td>
<td>Suffix-free</td>
<td>CCR</td>
<td>CR</td>
</tr>
<tr>
<td>3</td>
<td>Injective</td>
<td>CCR and Pre</td>
<td>CR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Folklore for CR and Pre [AnSt11]</td>
</tr>
</tbody>
</table>
Typically, $S = \{s\}$ is a singleton set.

<table>
<thead>
<tr>
<th></th>
<th>If Split is</th>
<th>and $h$ is</th>
<th>then $H = MD[h, \text{Split}, S]$ is</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Suffix-free</td>
<td>CR</td>
<td>CR</td>
<td>Known [Me,Da], reproofed</td>
</tr>
<tr>
<td>2</td>
<td>Suffix-free</td>
<td>CCR</td>
<td>CR</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Injective</td>
<td>CCR and Pre</td>
<td>CR</td>
<td>Folklore for CR and Pre [AnSt11]</td>
</tr>
</tbody>
</table>

![Diagram showing relationships between CR, CCR, and CCR + Pre](image-url)
Results

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

<table>
<thead>
<tr>
<th>If Split is</th>
<th>and h is</th>
<th>then $H = MD[h, \text{Split}, S]$ is</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Suffix-free</td>
<td>CR</td>
<td>CR</td>
<td>Known [Me,Da], reproved</td>
</tr>
<tr>
<td>2 Suffix-free</td>
<td>CCR</td>
<td>CR</td>
<td></td>
</tr>
<tr>
<td>3 Injective</td>
<td>CCR and Pre</td>
<td>CR</td>
<td>Folklore for CR and Pre [AnSt11]</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>If Split is</th>
<th>and ( h ) is</th>
<th>then ( H = \text{MD}[h,\text{Split},S] ) is</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Suffix-free</td>
<td>CR</td>
<td>CR</td>
<td>Known [Me, Da], reproved</td>
</tr>
<tr>
<td>2 Suffix-free</td>
<td>CCR</td>
<td>CR</td>
<td></td>
</tr>
<tr>
<td>3 Injective</td>
<td>CCR and Pre</td>
<td>CR</td>
<td>Discussed in the rest of this talk</td>
</tr>
</tbody>
</table>

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

\[ S = \{s\} \]

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

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

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 $\mathcal{A}_H$.

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

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

$$\text{Adv}^{\text{cr}}_H(\mathcal{A}_H) \leq \text{Adv}^{\text{RccrS}}_h(\mathcal{A}_h)$$

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 $\mathcal{A}_H$.

---

**Proof** uses the back-tracking paradigm of [Me,Da] but constructs a $\text{CCR}$-violating adversary rather than a $\text{CR}$-violating one.
**Theorem**

Let Split be a suffix-free splitting function. Given an adversary $A_H$, we define $A_h$ such that

$$\text{Adv}^\text{cr}_H(A_H) \leq \text{Adv}^R_{\text{cr}}(A_h)$$

The time complexity of $A_h$ is approximately that of $A_H$ plus the time to compute $H$. The memory complexity of $A_h$ is the maximum of the memory complexity of $A_H$ and term linear in the length of the output of $A_H$.

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

Let \( \text{Split} \) be a suffix-free splitting function. Given an adversary \( A_H \), we define \( A_h \) such that

\[
\text{Adv}^\text{cr}_H(A_H) \leq \text{Adv}^\text{RccrS}_h(A_h)
\]

The time complexity of \( A_h \) is approximately that of \( A_H \) plus the time to compute \( H \). The memory complexity of \( A_h \) is the maximum of the memory complexity of \( A_H \) and term linear in the length of the output of \( A_H \).

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

---

**Diagram**

The diagram illustrates the back-tracking paradigm described in the proof. It shows the process of constructing \( A_h \) from \( A_H \), highlighting the states and transitions involved in the splitting process.

---

Julia Len

14

UCSD
**Theorem**

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

$$\text{Adv}_H^{cr}(\mathcal{A}_H) \leq \text{Adv}_h^{R_{ccrS}}(\mathcal{A}_h)$$

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 $\mathcal{A}_H$.

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

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

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

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 $\mathcal{A}_H$.

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 $A_H$, we define $A_h$ such that

$$\text{Adv}^{cr}(A_H) \leq \text{Adv}^{R_{ccr}}_h(A_h)$$

The time complexity of $A_h$ is approximately that of $A_H$ plus the time to compute $H$. The memory complexity of $A_h$ is the maximum of the memory complexity of $A_H$ and term linear in the length of the output of $A_H$.

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 $A_h$ is the maximum of the memory complexity of $A_H$ and
a small constant.

---

adversary $A_h(s)$

\[
\begin{align*}
(M_1, M_2) & \leftarrow A_H(s, \epsilon) \\
 m_1 & \leftarrow \text{Split}(M_1) ; m_2 \leftarrow \text{Split}(M_2) ; n_1 \leftarrow |m_1| ; n_2 \leftarrow |m_2| \\
c_1[1] & \leftarrow s ; c_2[1] \leftarrow s; n \leftarrow \min(n_1, n_2) \\
\end{align*}
\]

If $(n_1 > n_2)$ then

For $i = 1, \ldots, n_1 - n_2$ do $c_1[i + 1] \leftarrow h(m_1[i], c_1[i])$

If $(n_2 > n_1)$ then

For $i = 1, \ldots, n_2 - n_1$ do $c_2[i + 1] \leftarrow h(m_2[i], c_2[i])$

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

\[
\begin{align*}
m_1 & \leftarrow m_1[n_1 - n + i]; c_1 \leftarrow c_1[n_1 - n + i] \\
m_2 & \leftarrow m_2[n_2 - n + i]; c_2 \leftarrow c_2[n_2 - n + i] \\
c'_1 & \leftarrow h(m_1, c_1) \\
c'_2 & \leftarrow h(m_2, c_2) \\
\end{align*}
\]

If $(c'_1 = c'_2)$ and $(m_1, c_1) \neq (m_2, c_2)$ then

\[
\begin{align*}
a_1 & \leftarrow (m_1[n_1 - n + i - 1], c_1[n_1 - n + i - 1]) \\
a_2 & \leftarrow (m_2[n_2 - n + i - 1], c_2[n_2 - n + i - 1]) \\
\text{Return } ((m_1, c_1), (m_2, c_2), a_1, a_2) \\
\end{align*}
\]

\[
\begin{align*}
c_1[n_1 - n + i + 1] & \leftarrow c'_1 \\
c_2[n_2 - n + i + 1] & \leftarrow c'_2 \\
\end{align*}
\]

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 $A_h$ is the maximum of the memory complexity of $A_H$ and a small constant.

---

**adversary** $A_h(s)$

$(M_1, M_2) \leftarrow A_H(s, \varepsilon)$

$m_1 \leftarrow \text{Split}(M_1)$; $m_2 \leftarrow \text{Split}(M_2)$; $n_1 \leftarrow |m_1|$; $n_2 \leftarrow |m_2|$

$c_1[1] \leftarrow s$; $c_2[1] \leftarrow s$; $n \leftarrow \min(n_1, n_2)$

If $(n_1 > n_2)$ then

For $i = 1, \ldots, n_1 - n_2$ do $c_1[i + 1] \leftarrow h(m_1[i], c_1[i])$

If $(n_2 > n_1)$ then

For $i = 1, \ldots, n_2 - n_1$ do $c_2[i + 1] \leftarrow h(m_2[i], c_2[i])$

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

$m_1 \leftarrow m_1[n_1 - n + i]$; $c_1 \leftarrow c_1[n_1 - n + i]$

$m_2 \leftarrow m_2[n_2 - n + i]$; $c_2 \leftarrow c_2[n_2 - n + i]$

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

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

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

$a_1 \leftarrow (m_1[n_1 - n + i - 1], c_1[n_1 - n + i - 1])$

$a_2 \leftarrow (m_2[n_2 - n + i - 1], c_2[n_2 - n + i - 1])$

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

$c_1[n_1 - n + i + 1] \leftarrow c_1'$

$c_2[n_2 - n + i + 1] \leftarrow c_2'$

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 $A_h$ is the maximum of the memory complexity of $A_H$ and a small constant.

\[
\begin{align*}
&\text{adversary } A_h(s) \\
&(M_1, M_2) \leftarrow A_H(s, \epsilon) \\
&m_1 \leftarrow \text{Split}(M_1); m_2 \leftarrow \text{Split}(M_2); n_1 \leftarrow |m_1|; n_2 \leftarrow |m_2| \\
&c_1[1] \leftarrow s; c_2[1] \leftarrow s; n \leftarrow \min(n_1, n_2) \\
&\text{If } (n_1 > n_2) \text{ then} \\
&\quad \text{For } i = 1, \ldots, n_1 - n_2 \text{ do } c_1[i + 1] \leftarrow h(m_1[i], c_1[i]) \\
&\text{If } (n_2 > n_1) \text{ then} \\
&\quad \text{For } i = 1, \ldots, n_2 - n_1 \text{ do } c_2[i + 1] \leftarrow h(m_2[i], c_2[i]) \\
&\text{For } i = 1, \ldots, n \text{ do} \\
&\quad m_1 \leftarrow m_1[n_1 - n + i]; c_1 \leftarrow c_1[n_1 - n + i] \\
&\quad m_2 \leftarrow m_2[n_2 - n + i]; c_2 \leftarrow c_2[n_2 - n + i] \\
&\quad c'_1 \leftarrow h(m_1, c_1) \\
&\quad c'_2 \leftarrow h(m_2, c_2) \\
&\text{If } (c'_1 = c'_2) \text{ and } (m_1, c_1) \neq (m_2, c_2) \text{ then} \\
&\quad a_1 \leftarrow (m_1[n_1 - n + i - 1], c_1[n_1 - n + i - 1]) \\
&\quad a_2 \leftarrow (m_2[n_2 - n + i - 1], c_2[n_2 - n + i - 1]) \\
&\quad \text{Return } ((m_1, c_1), (m_2, c_2), a_1, a_2) \\
&c_1[n_1 - n + i + 1] \leftarrow c'_1 \\
&c_2[n_2 - n + i + 1] \leftarrow c'_2 \\
&\text{Return } \bot
\end{align*}
\]

ACFK17: “memory tightness is important”

Natural reduction was *not* memory tight.
Theorem  Same as above, except:
The memory complexity of $A_h$ is the maximum of the memory complexity of $A_H$ and a small constant.

ACFK17: “memory tightness is important”

Natural reduction was not memory tight.
**Theorem**  Same as above, except:
The memory complexity of $A_h$ is the maximum of the memory complexity of $A_H$ and a small constant.

ACFK17: “memory tightness is important”

Natural reduction was *not* memory tight.
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.

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:

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

**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. \( S = \{0, 1\}^{h.cl} \setminus \{1\|0^{h.cl-1}, 1^2\|0^{h.cl-2}\} \)

**Claims**
1. \( h \) is CCR
2. \( h \) is **not** CR
3. \( H = MD[h, \text{Split}, S] \) **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. \( S = \{0, 1\}^{h.cl} \setminus \{1^{h.cl-1}, 1^2 \| 0^{h.cl-2}\} \)

**Claims**
1. \( h \) is CCR
2. \( h \) is not CR
3. \( H = \text{MD}[h, \text{Split}, S] \) 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. \( S = \{0, 1\}^{h.cl} \setminus \{1||0^{h.cl-1}, 1^2||0^{h.cl-2}\} \)

**Claims**
1. \( h \) is CCR
2. \( h \) is not CR
3. \( H = MD[h, Split, S] \) is CR

\[
\begin{align*}
  h(m, c) \\
  \text{If } (m, c) \in \{(0^{h.ml}, 1||0^{h.cl-1}), (1^{h.ml}, 1^2||0^{h.cl-2})\} \\
  \quad \text{Return } 1^{h.cl} \\
  \text{Return } 0 || h'(m, c)
\end{align*}
\]
### 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. $S = \{0, 1\}^{h.cl} \setminus \{1\|0^{h.cl-1}, 1^2\|0^{h.cl-2}\}$

#### Claims

1. $h$ is CCR
2. $h$ is not CR
3. $H = \text{MD}[h, \text{Split}, S]$ 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 $0\| h'(m, c)$

---

Diagram:

```
S  ↦  h  ↦  h  ↦  ...  ↦  h
  ↘   ↘   ↘   ↘   ↘
  m[1] m[2] m[n] h'
  ↗   ↗   ↗   ↗   ↗
  0\| h'(m[1], c[1])
```
**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. \( S = \{0, 1\}^{h.cl} \setminus \{1\}^{h.cl-1} \cup \{2\}^{h.cl-2} \)

**Claims**

1. \( h \) is CCR
2. \( h \) is **not** CR
3. \( H = \text{MD}[h, \text{Split, } S] \) is CR

\[
\begin{align*}
h(m, c) \\
\text{If } (m, c) \in \{(0^{h.ml}, 1 \parallel 0^{h.cl-1}), (1^{h.ml}, 1^2 \parallel 0^{h.cl-2})\} \\
\text{Return } 1^{h.cl} \\
\text{Return } 0 \parallel h'(m, c)
\end{align*}
\]
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. \( S = \{0, 1\}^{h.cl} \setminus \{1||0^{h.cl-1}, 1^2||0^{h.cl-2}\} \)

**Claims**

1. \( h \) is CCR
2. \( h \) is **not** CR
3. \( H = \text{MD}[h, \text{Split}, S] \) 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 \(0||h'(m, c)\)

\[ S \xrightarrow{\$} s \xrightarrow{\bullet} s \xrightarrow{\bullet} \ldots \rightarrow s \]

\[ 0||h'(m[1], c[1]) \quad 0||h'(m[2], c[2]) \quad 0||h'(m[n-1], c[n-1]) \]
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 CCR function \( h' : \{0, 1\}^{h.ml} \times \{0, 1\}^{h.cl} \rightarrow \{0, 1\}^{h.cl-1} \)
3. \( S = \{0, 1\}^{h.cl} \setminus \{1||0^{h.cl-1}, 1^2||0^{h.cl-2}\} \)

**Claims**
1. \( h \) is CCR
2. \( h \) is not CR
3. \( H = \text{MD}[h, \text{Split}, S] \) is CR

\[
\begin{align*}
&h(m, c) \\
&\text{If } (m, c) \in \{(0^{h.ml}, 1||0^{h.cl-1}), (1^{h.ml}, 1^2||0^{h.cl-2})\} \\
&\quad \text{Return } 1^{h.cl} \\
&\text{Return } 0||h'(m, c)
\end{align*}
\]
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 \( \text{Split} \) be an injective splitting function. Given an adversary \( \mathcal{A}_H \) we define adversaries \( \mathcal{A}_h \) and \( \mathcal{B}_h \) such that

\[
\text{Adv}^{\text{cr}}_H(\mathcal{A}_H) \leq \text{Adv}^{\text{R}}_{\text{ccr}}(\mathcal{A}_h) + \text{Adv}^{\text{R}}_{\text{pre}}(\mathcal{B}_h)
\]

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 $\text{Split}$ be an injective splitting function. Given an adversary $A_H$ we define adversaries $A_h$ and $B_h$ such that

$$\text{Adv}^{\text{cr}}_H(A_H) \leq \text{Adv}^{R_{ccrS}}_h(A_h) + \text{Adv}^{R_{preS}}_h(B_h)$$

The time complexities of $A_h$ and $B_h$ are that of $A_H$ plus the time to compute $H$ on its output. The memory complexities of $A_h$ and $B_h$ are the maximum of that of $A_H$ and a small constant.

- \[ H(M) \]
- \[ H(M) \]
- \[ H(M) \]
- \[ H(M) \]
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 $\text{Split}$ be an injective splitting function. Given an adversary $A_H$ we define adversaries $A_h$ and $B_h$ such that

$$\text{Adv}^\text{cr}_H(A_H) \leq \text{Adv}^\text{Rccr}_h(A_h) + \text{Adv}^\text{Rpre}_h(B_h)$$

The time complexities of $A_h$ and $B_h$ are that of $A_H$ plus the time to compute $H$ on its output. The memory complexities of $A_h$ and $B_h$ are the maximum of that of $A_H$ and a small constant.

**Case 1:** This is $s$. 

![Diagram](image)
**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 $\text{Split}$ be an injective splitting function. Given an adversary $A_H$ we define adversaries $A_h$ and $B_h$ such that

$$\text{Adv}_{H}^{cr}(A_H) \leq \text{Adv}_{h}^{R_{ccr}}(A_h) + \text{Adv}_{h}^{R_{pre}}(B_h)$$

The time complexities of $A_h$ and $B_h$ are that of $A_H$ plus the time to compute $H$ on its output. The memory complexities of $A_h$ and $B_h$ are the maximum of that of $A_H$ 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 \textit{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 \textit{more easily address memory complexity}.

• We showed how the MD transform can be made \textit{more efficient} by using an \textit{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.