Failure-Tolerance: A Design Principle for Iterated Hashfunctions

Krakow, June 2005

Stefan Lucks

University of Mannheim
Germany

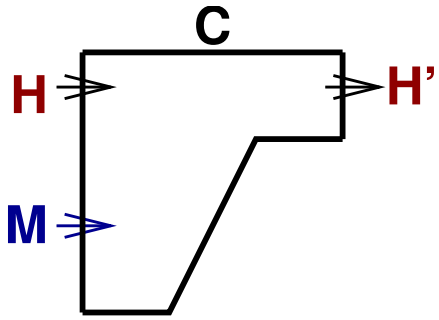
Roadmap

Current Iterated Hashfunctions (Merkle-Damgård)

Iterate a given compression function

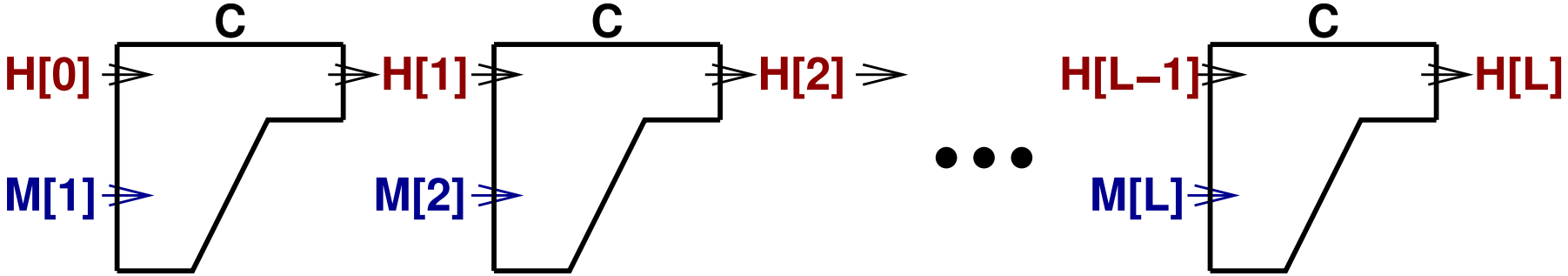
$$\mathbf{C} : \{0,1\}^{\mathbf{n}} \times \{0,1\}^{\mathbf{m}} \to \{0,1\}^{\mathbf{n}}$$

(input size $n + m$)



as much as needed to hash a given message of arbitrary size:

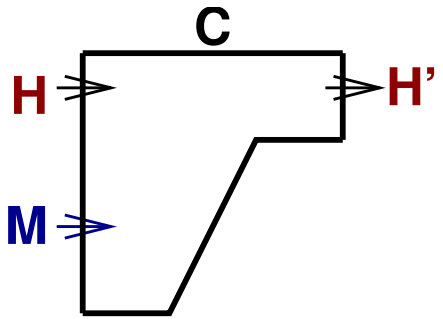
$\mathbf{H[i]} = \mathbf{C}(\mathbf{H[i-1]}, \mathbf{M[i]})$:



The Bright Side: A Formal Proof of Security

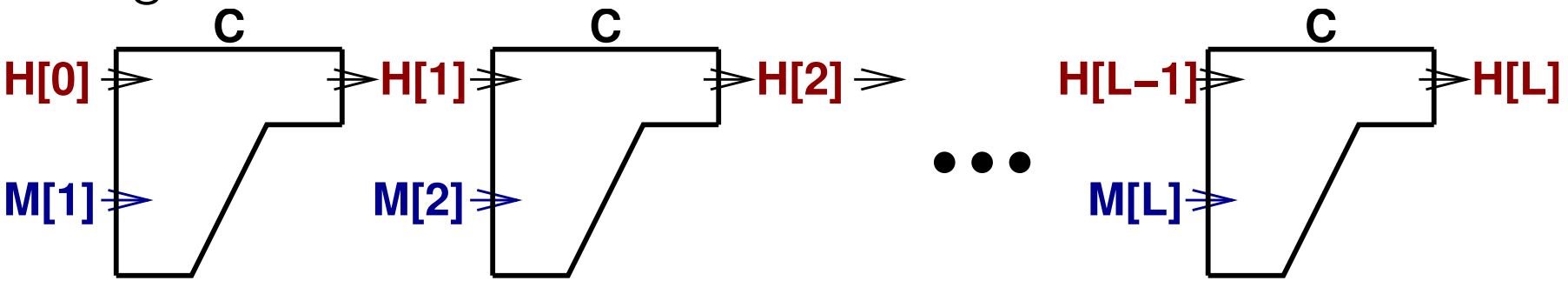
If

finding collisions for **C** is infeasible,



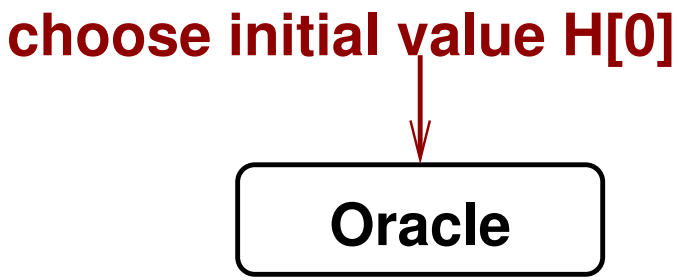
then

finding collisions for H is infeasible as well.



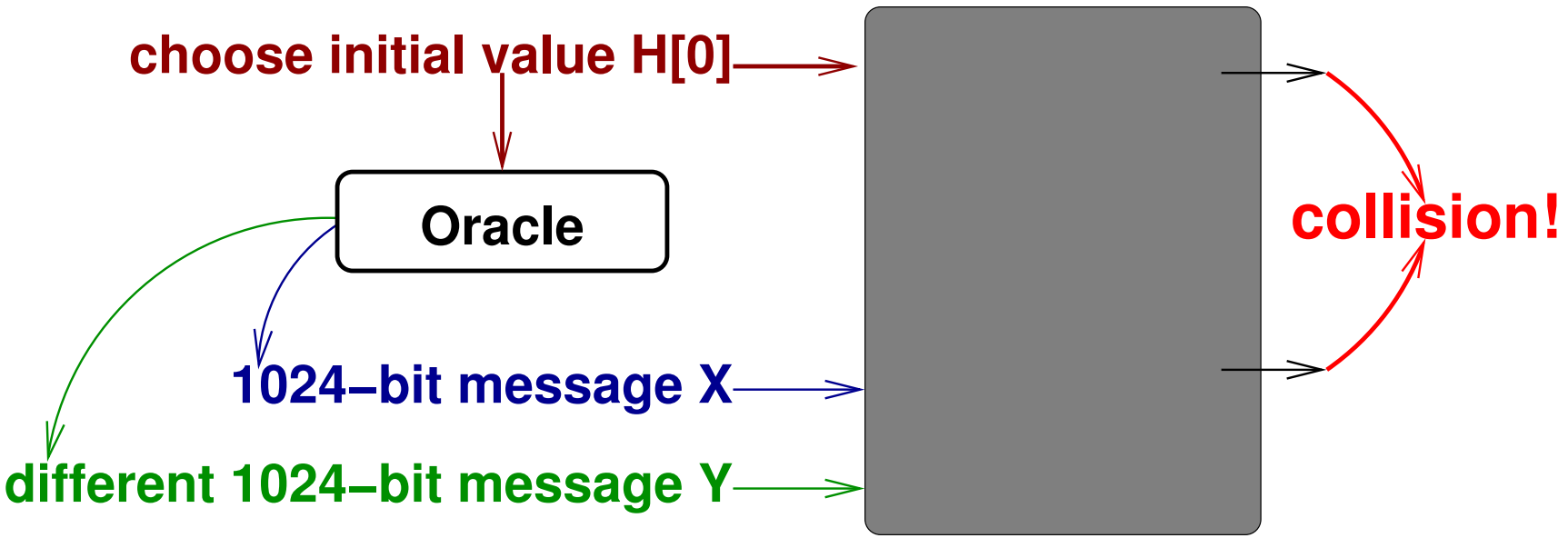
The Dark Side: Weakness of Compression Functions

- Example: the Wang/Yu attack against MD5



The Dark Side: Weakness of Compression Functions

- Example: the Wang/Yu attack against MD5



The Dark Side: Structural Weaknesses – Extending Collisions

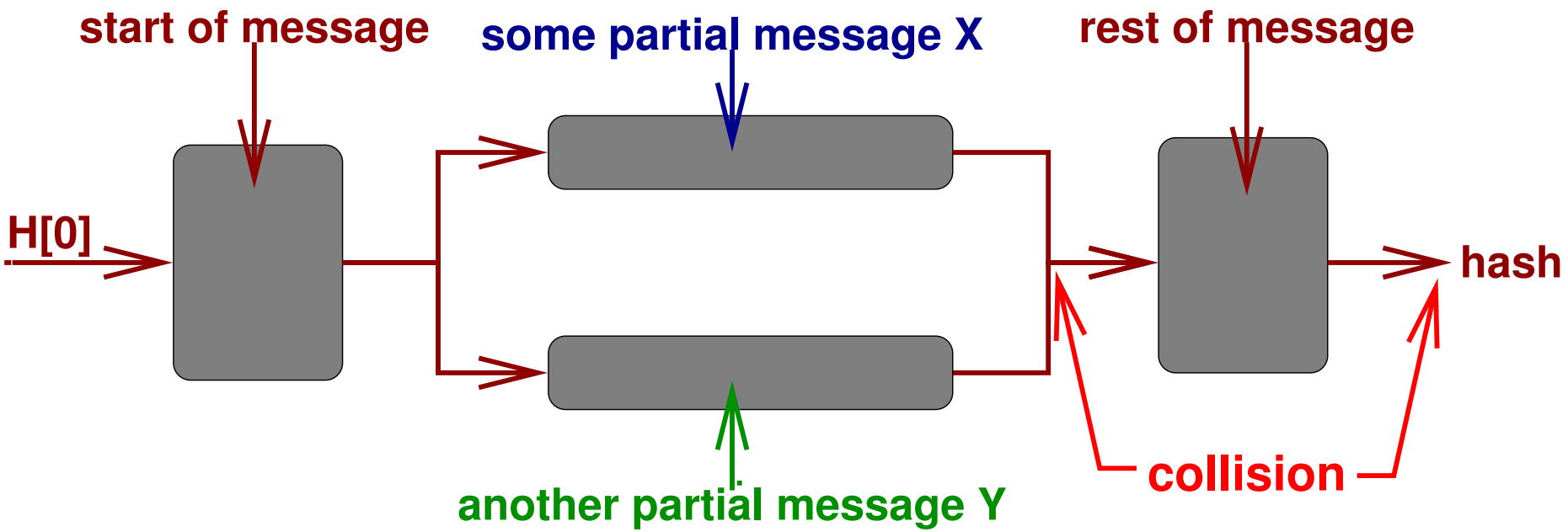
Common **start** and **rest**, random **X** $\neq$ **Y**:

Hash(start,X,rest) = Hash(start,Y,rest)

The Dark Side: Structural Weaknesses – Extending Collisions

Common **start** and **rest**, random **X** $\neq$ **Y**:

Hash(start,X,rest) = Hash(start,Y,rest)



The Dark Side: Exploiting for the Collision Extenstion Weakness

- **Joux'** multicollision attack (Crypto 04).

The Dark Side: Exploiting for the Collision Extenstion Weakness

- **Joux'** multicollision attack (Crypto 04).
- Turning "random" collisions at the compression function level into meaningful collisions at the hash level:
  - **D. Kaminski**: MD5 to be considered harmful someday.
  - **O. Mikle**: Practical Attacks on Digital Signatures using MD5 message digest
  - **A. Lenstra, X. Wang, B. de Weger**: Colliding X.509 Certificates.
  - **A. Lenstra, B. de Weger**: On the possibility of constructing meaningful hash collisions for public keys.
  - **M. Daum, S. Lucks**: The Story of Alice and her Boss.

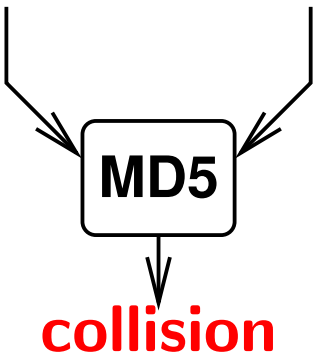
The Dark Side:
The Story of Alice and her Boss

Letter of Recommendation

. . . I highly recommend hiring her.

Order

. . . access to all confidential and secret information . . .



A new View at Hash Functions?

- Traditional view at iterated hash functions:
  - hash **long messages**,
    internally some compression function with a fixed input size.
- Traditional view at block ciphers:
  - encrypt **fixed-size message blocks**,
    use some mode of operation to encrypt long messages.

A new View at Hash Functions?

- Traditional view at iterated hash functions:
  - hash **long messages**,
    internally some compression function with a fixed input size.
- Traditional view at block ciphers:
  - encrypt **fixed-size message blocks**,
    use some mode of operation to encrypt long messages.
- Treat iterated hash functions like block ciphers:
  - distinguish the primitive from its mode of operation,
  - and **analyse primitive and mode independently!**

Failure-Tolerance for Hash Functions

Failure-Tolerant Iterated Hash Functions

Failure-tolerant systems:

"The ability to respond gracefully to an unexpected failure."

Failure-Tolerant Iterated Hash Functions

Failure-tolerant systems:

"The ability of a system to respond gracefully to an unexpected failure."

Failure-tolerant iterated hash functions:

- If the compression function is *collision resistant*: OK!
- If the compression function is *fails badly*: No Chance!
- But what, if the compression function **slightly fails**?

Failure-Tolerant Iterated Hash Functions

Failure-tolerant systems:

"The ability of a system to respond gracefully to an unexpected failure."

Failure-tolerant iterated hash functions:

"The ability to provide some security, even if the compression function slightly fails its stated goal of being collision resistant."

The Wide-Pipe Hash

Idea:

larger internal state
(**w**-bit instead of $\mathbf{n} < \mathbf{w}$)
two compression functions

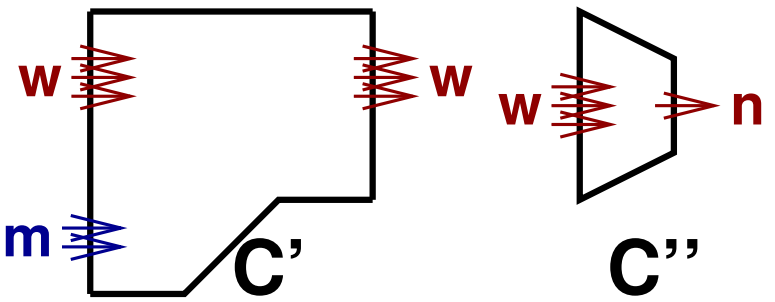
$$\begin{aligned} \mathbf{C}' &: \{0,1\}^{\mathbf{w}} \times \{0,1\}^{\mathbf{m}} \to \{0,1\}^{\mathbf{w}} \text{ and} \\ \mathbf{C}'' &: \{0,1\}^{\mathbf{w}} \to \{0,1\}^{\mathbf{n}}. \end{aligned}$$

The Wide-Pipe Hash

Idea:

larger internal state
(**w**-bit instead of $\mathbf{n} < \mathbf{w}$)

two compression functions



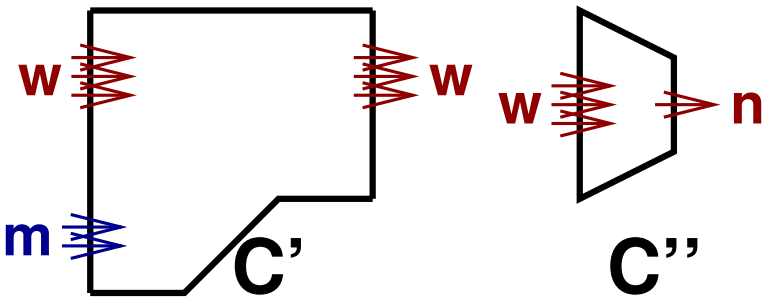
$$\mathbf{C}' : \{0,1\}^{\mathbf{w}} \times \{0,1\}^{\mathbf{m}} \to \{0,1\}^{\mathbf{w}} \text{ and}$$
$$\mathbf{C}'' : \{0,1\}^{\mathbf{w}} \to \{0,1\}^{\mathbf{n}}.$$

The Wide-Pipe Hash

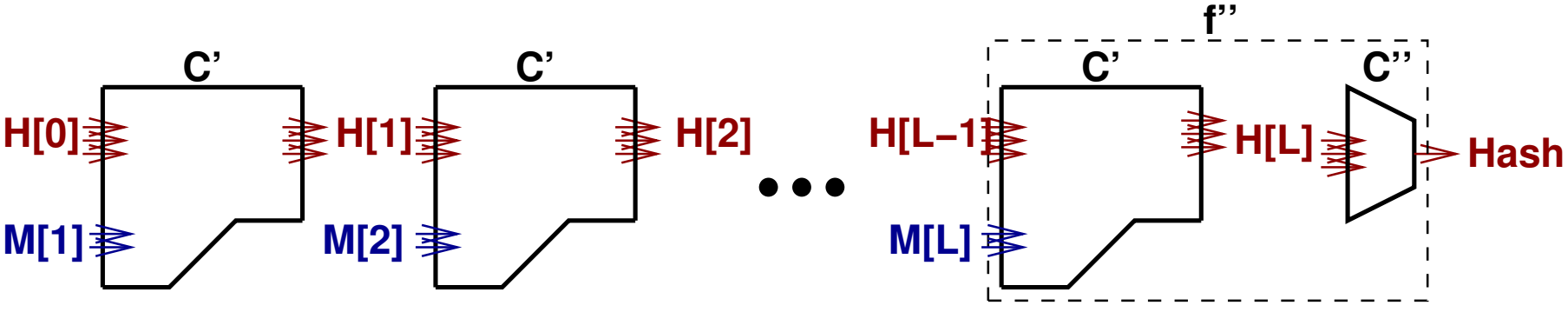
Idea:

larger internal state
(**w**-bit instead of $\mathbf{n} < \mathbf{w}$)

two compression functions



$$\mathbf{C}' : \{0,1\}^{\mathbf{w}} \times \{0,1\}^{\mathbf{m}} \to \{0,1\}^{\mathbf{w}} \text{ and}$$
$$\mathbf{C}'' : \{0,1\}^{\mathbf{w}} \to \{0,1\}^{\mathbf{n}}.$$

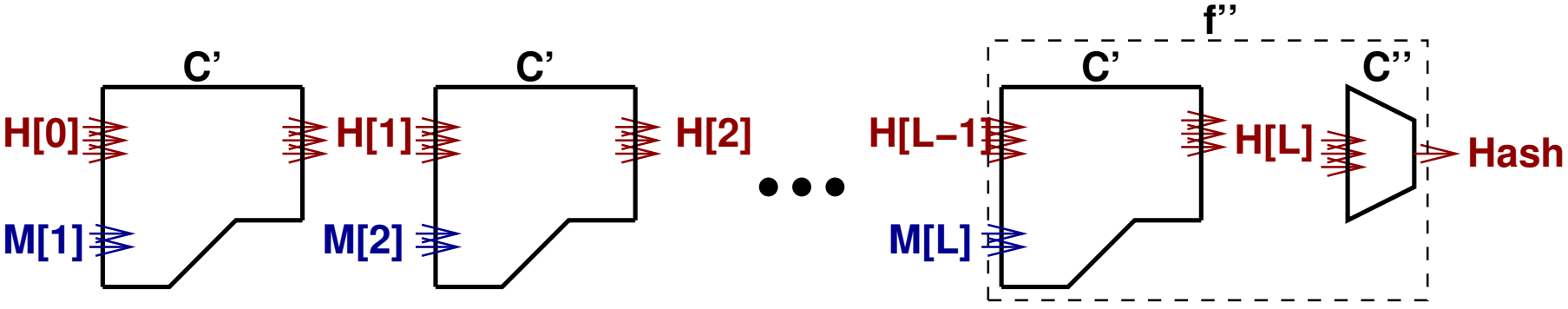


The Wide-Pipe Hash

Rationale:

Collision resistance for $\mathbf{C}'$ more plausible than collision resistance for $\mathbf{C}''$ (or $\mathbf{f}''$) ("dotted line", where security could break).

$$\mathbf{C}' : \{0,1\}^{\mathbf{w}} \times \{0,1\}^{\mathbf{m}} \to \{0,1\}^{\mathbf{w}} \text{ and}$$
$$\mathbf{C}'' : \{0,1\}^{\mathbf{w}} \to \{0,1\}^{\mathbf{n}}.$$



The Wide-Pipe Hash: Security

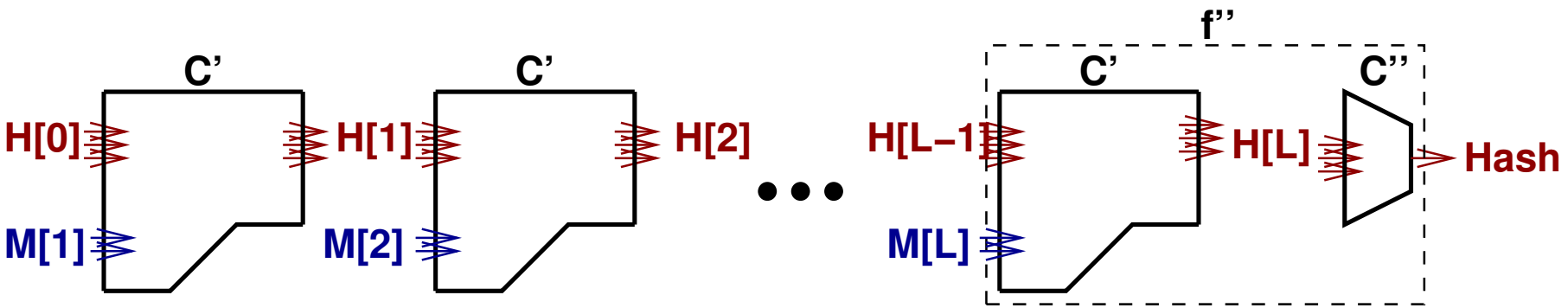
Results: If $\mathbf{C}'$ is collision resistant, than the Wide-Pipe hash is

1. resistant against multi-collision attacks , if $\mathbf{f}''$ is so,

All results hold, *even if* $\mathbf{f}''$ *is not collision resistant!*

.

.

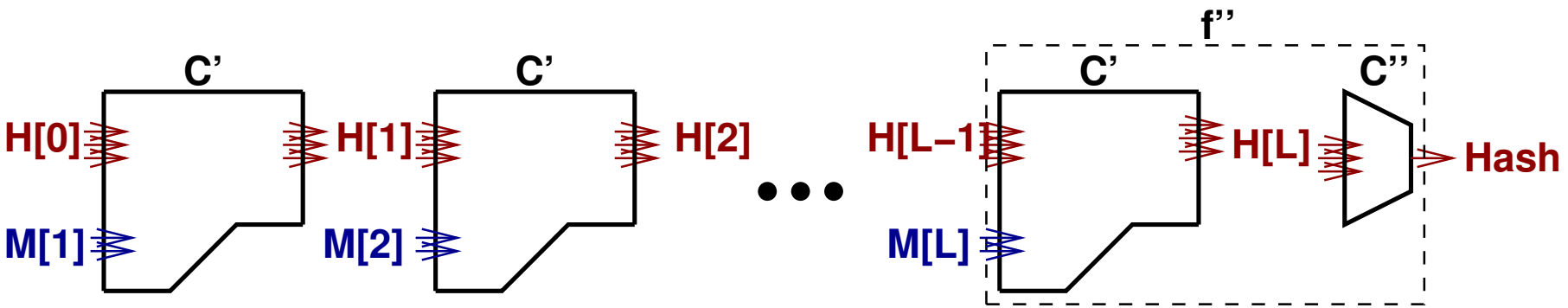


The Wide-Pipe Hash: Security

Results: If $\mathbf{C}'$ is collision resistant, than the Wide-Pipe hash is

1. resistant against multi-collision attacks , if $\mathbf{f}''$ is so,
2. resistant against multi-(2nd) preimage attacks, if $\mathbf{f}''$ is so, and

All results hold, *even if* $\mathbf{f}''$ *is not collision resistant!*
First two results: proof of security in standard model.
.

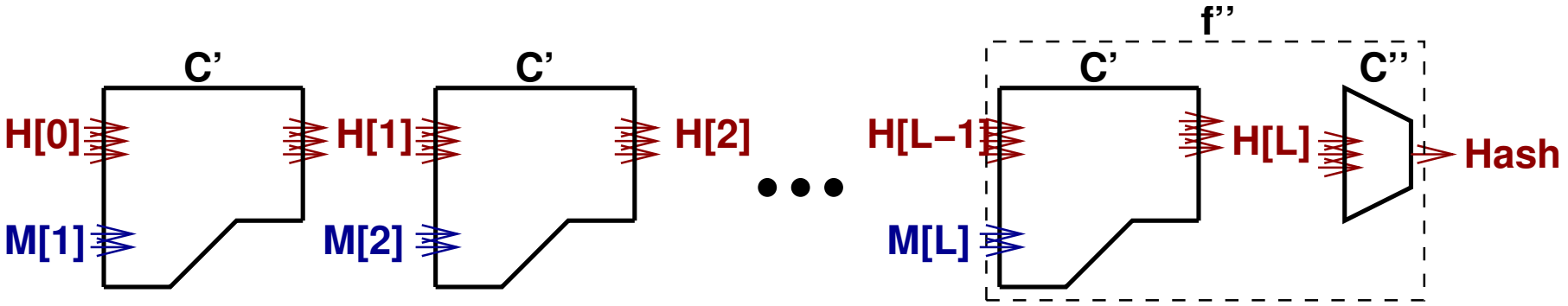


The Wide-Pipe Hash: Security

Results: If $\mathbf{C}'$ is collision resistant, than the Wide-Pipe hash is

1. resistant against multi-collision attacks , if $\mathbf{f}''$ is so,
2. resistant against multi-(2nd) preimage attacks, if $\mathbf{f}''$ is so, and
3. resistant against collision extension attacks.

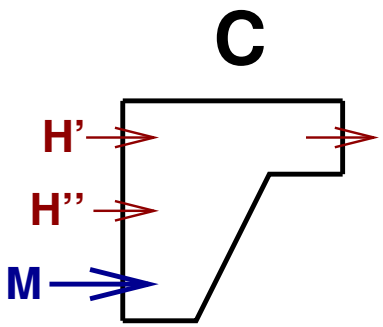
All results hold, *even if* $\mathbf{f}''$ *is not collision resistant!*
First two results: proof of security in standard model.
Third result: proof of security in random oracle model.



The Double-Pipe Hash

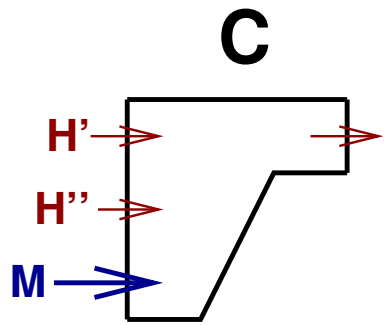
Idea: Realise $\mathbf{C}'$ and $\mathbf{f}''$ by one single compression function

$$\mathbf{C} : \ldots \longrightarrow \{0,1\}^{\mathbf{n}}.$$



Realise $\mathbf{C}'$ and $\mathbf{f}''$ by one single compression function

$$\mathbf{C} : \{0,1\}^{\mathbf{n}} \times \{0,1\}^{\mathbf{n}} \times \{0,1\}^{\mathbf{m}} \to \{0,1\}^{\mathbf{n}}$$

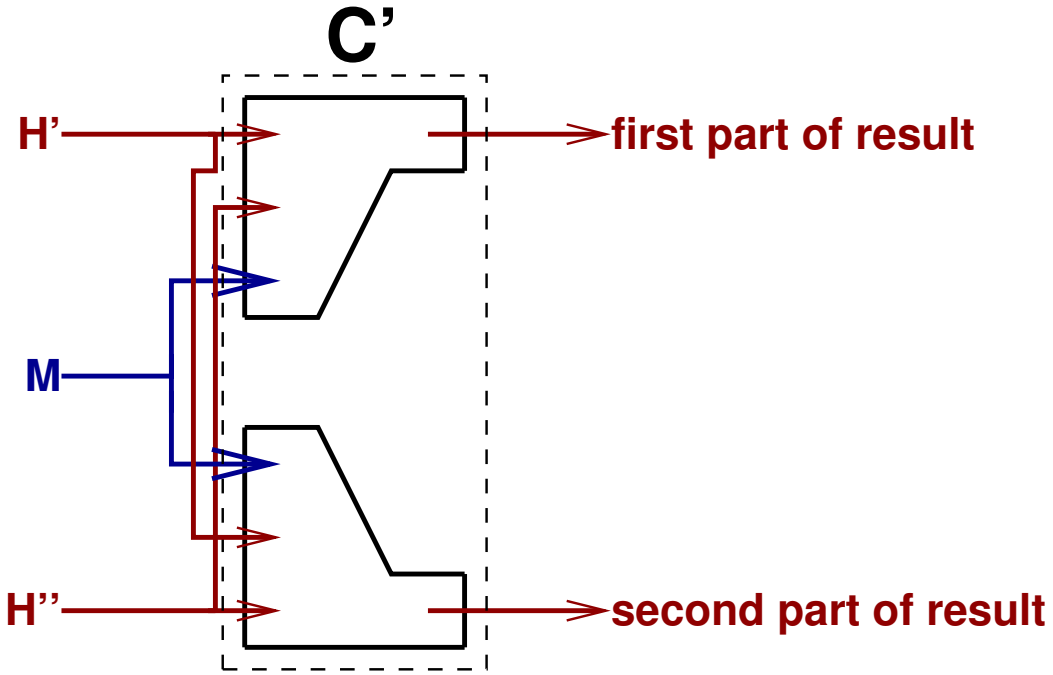


Realise $\mathbf{C}'$ and $\mathbf{f}''$ by one single compression function

$$\mathbf{C} : \{0,1\}^{\mathbf{n}} \times \{0,1\}^{\mathbf{n}} \times \{0,1\}^{\mathbf{m}} \to \{0,1\}^{\mathbf{n}}$$

$$\mathbf{C}'(\mathbf{H}', \mathbf{H}'', \mathbf{M}) \quad = \quad (\mathbf{C}(\mathbf{H}', \mathbf{H}'', \mathbf{M}), \mathbf{C}(\mathbf{H}'', \mathbf{H}', \mathbf{M}))$$

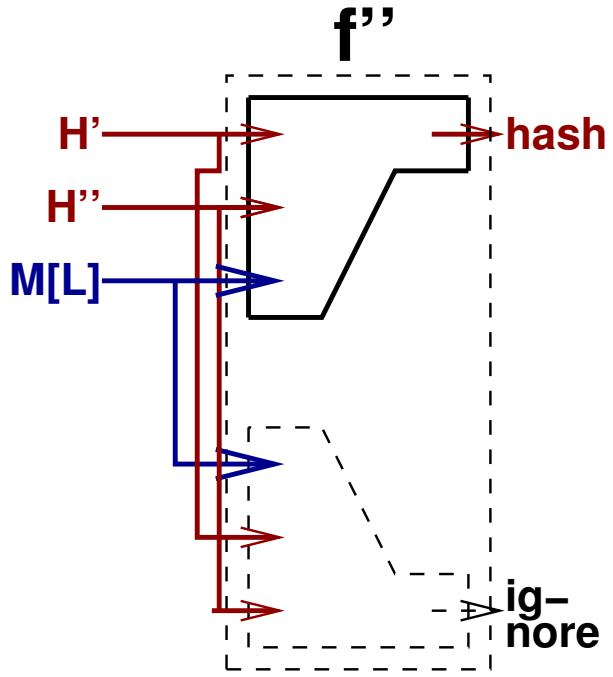
(change the order of $\mathbf{H}'$ and $\mathbf{H}''$)



Realise $\mathbf{C}'$ and $\mathbf{f}''$ by one single compression function

$$\mathbf{C} : \{0,1\}^{\mathbf{n}} \times \{0,1\}^{\mathbf{n}} \times \{0,1\}^{\mathbf{m}} \to \{0,1\}^{\mathbf{n}}$$

$$\begin{aligned} \mathbf{C}'(\mathbf{H}', \mathbf{H}'', \mathbf{M}) &= (\mathbf{C}(\mathbf{H}', \mathbf{H}'', \mathbf{M}), \mathbf{C}(\mathbf{H}'', \mathbf{H}', \mathbf{M})) \\ \mathbf{f}''(\mathbf{H}', \mathbf{H}'', \mathbf{M}) &= \mathbf{C}(\mathbf{H}', \mathbf{H}'', \mathbf{M}) \ \textit{(truncated } \mathbf{C}'\textit{)} \end{aligned}$$

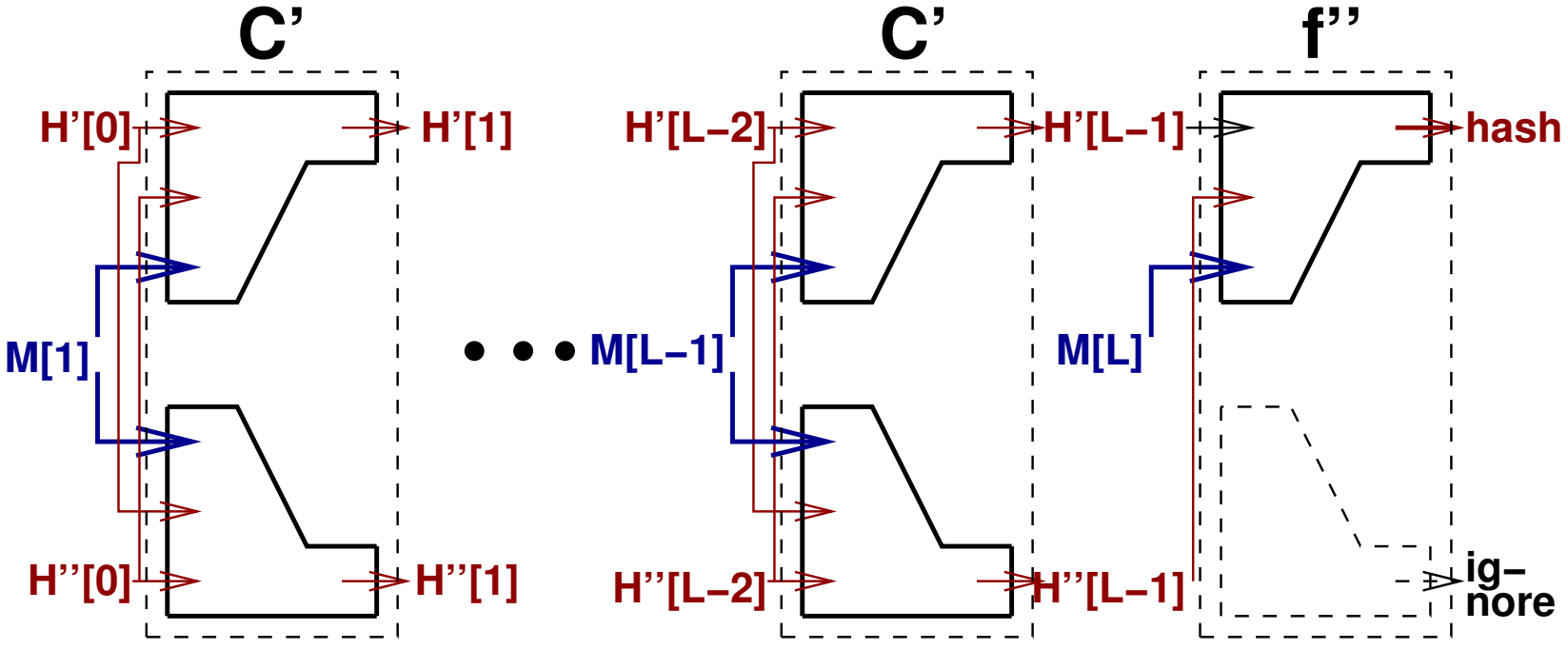


Realise $\mathbf{C}'$ and $\mathbf{f}''$ by one single compression function

$$\mathbf{C} : \{0,1\}^{\mathbf{n}} \times \{0,1\}^{\mathbf{n}} \times \{0,1\}^{\mathbf{m}} \to \{0,1\}^{\mathbf{n}}$$

$$\begin{aligned} \mathbf{C}'(\mathbf{H}', \mathbf{H}'', \mathbf{M}) &= (\mathbf{C}(\mathbf{H}', \mathbf{H}'', \mathbf{M}), \mathbf{C}(\mathbf{H}'', \mathbf{H}', \mathbf{M})) \\ \mathbf{f}''(\mathbf{H}', \mathbf{H}'', \mathbf{M}) &= \mathbf{C}(\mathbf{H}', \mathbf{H}'', \mathbf{M}) \ \textit{(truncated } \mathbf{C}'\textit{)} \end{aligned}$$

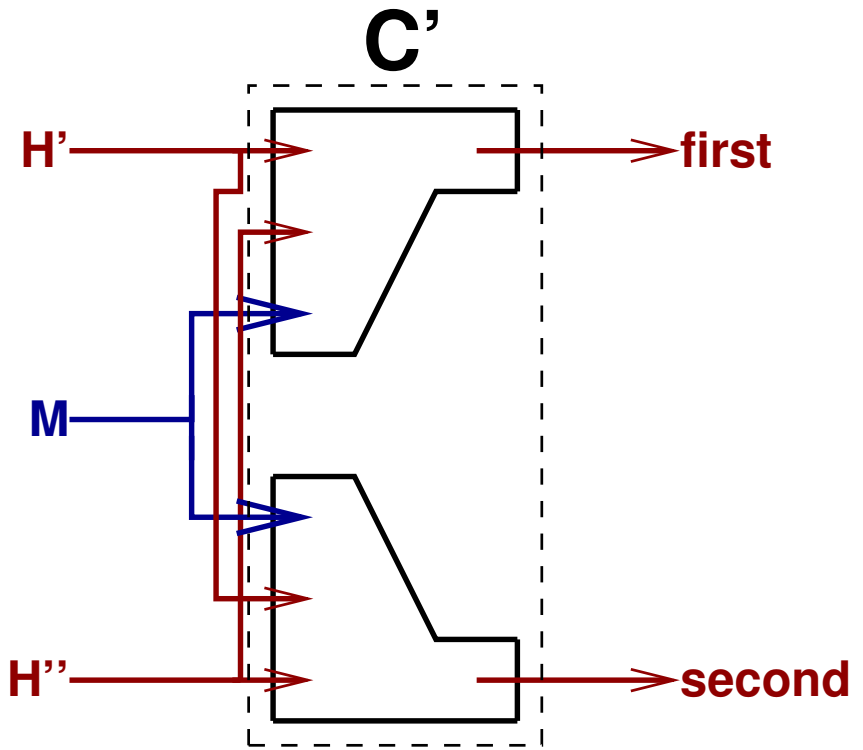
The full hash function:



Double-Pipe Hash: Security

- Double-Pipe as a specific instantiation of Wide-Pipe?

$$\mathbf{C}'(\mathbf{H}', \mathbf{H}'', \mathbf{M}) = (\mathbf{C}(\mathbf{H}', \mathbf{H}'', \mathbf{M}), \mathbf{C}(\mathbf{H}'', \mathbf{H}', \mathbf{M}))$$

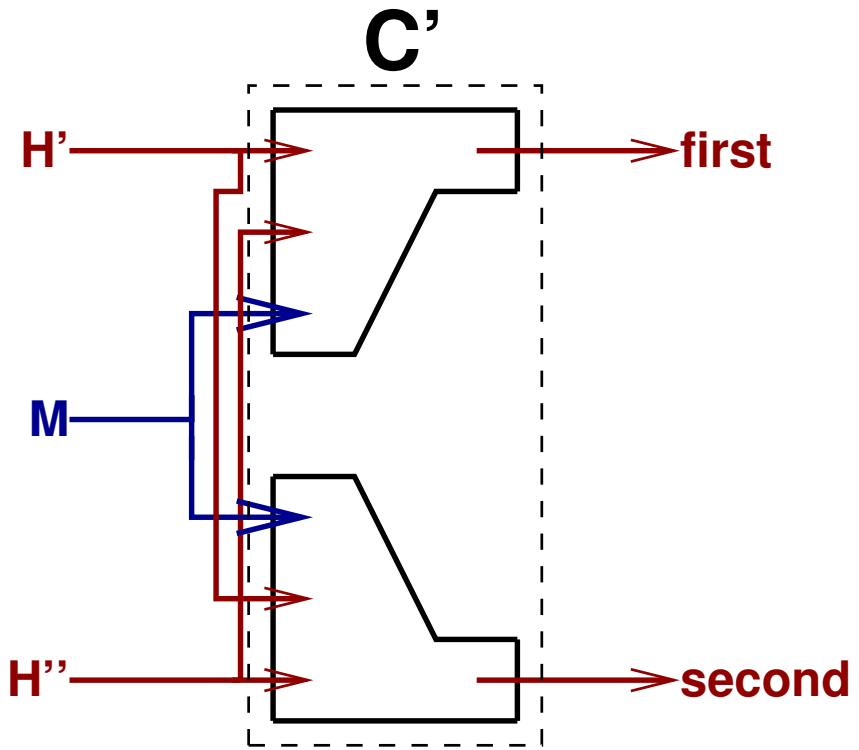


Double-Pipe Hash: Security

- Double-Pipe as a specific instantiation of Wide-Pipe?

$$\mathbf{C}'(\mathbf{H}', \mathbf{H}'', \mathbf{M}) = (\mathbf{C}(\mathbf{H}', \mathbf{H}'', \mathbf{M}), \mathbf{C}(\mathbf{H}'', \mathbf{H}', \mathbf{M}))$$

- Set $\mathbf{H}' = \mathbf{H}''$.
  Finding collisions for $\mathbf{C}'$
  as *easy* as finding them for $\mathbf{C}$.



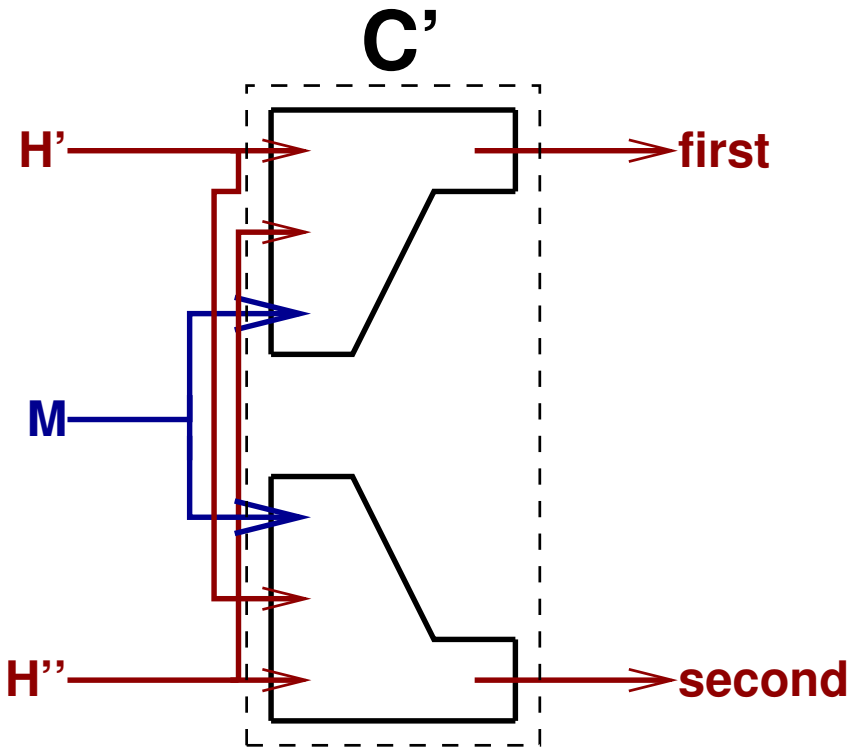
Double-Pipe Hash: Security

- Double-Pipe as a specific instantiation of Wide-Pipe?

$$\mathbf{C}'(\mathbf{H}', \mathbf{H}'', \mathbf{M}) = (\mathbf{C}(\mathbf{H}', \mathbf{H}'', \mathbf{M}), \mathbf{C}(\mathbf{H}'', \mathbf{H}', \mathbf{M}))$$

- Set $\mathbf{H}' = \mathbf{H}''$.
  Finding collisions for $\mathbf{C}'$
  as *easy* as finding them for $\mathbf{C}$.
- Exclude this special case, make new **assumption** on $\mathbf{C}$:

(weaker assumption than collision resistance for $\mathbf{C}$).



Double-Pipe Hash: Security

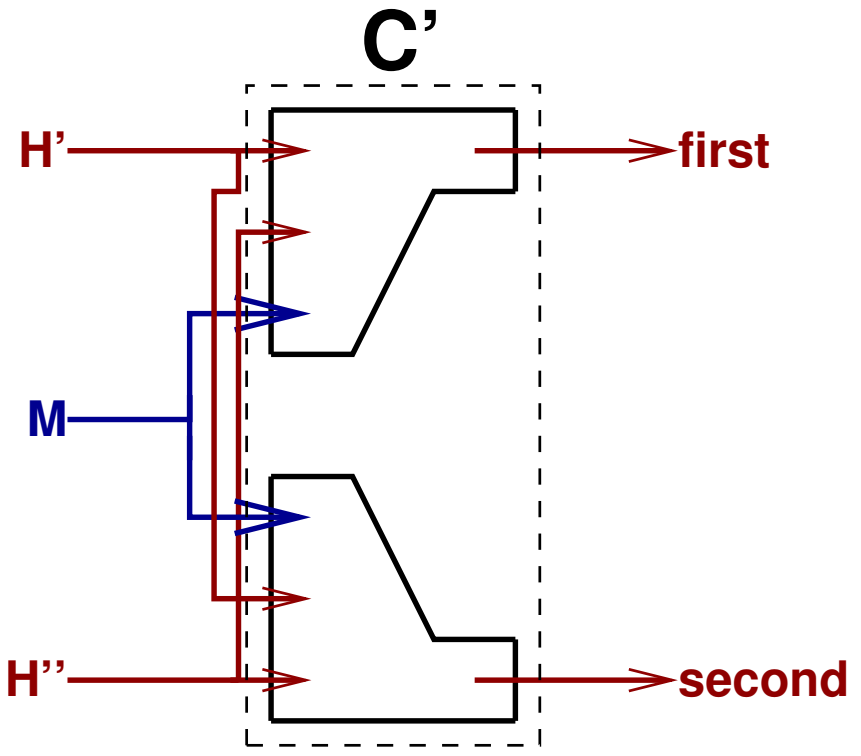
- Double-Pipe as a specific instantiation of Wide-Pipe?

$$\mathbf{C}'(\mathbf{H}', \mathbf{H}'', \mathbf{M}) = (\mathbf{C}(\mathbf{H}', \mathbf{H}'', \mathbf{M}), \mathbf{C}(\mathbf{H}'', \mathbf{H}', \mathbf{M}))$$

- Set $\mathbf{H}' = \mathbf{H}''$.
  Finding collisions for $\mathbf{C}'$
  as *easy* as finding them for $\mathbf{C}$.
- Exclude this special case, make new **assumption** on $\mathbf{C}$:

  It is **hard** to find
  $\mathbf{H}' \neq \mathbf{H}''$ and $\mathbf{M}$
  with **first** = **second**

  (weaker assumption than collision resistance for $\mathbf{C}$).



Double-Pipe Hash: Security (2)

- Double-Pipe as a specific instantiation of Wide-Pipe?

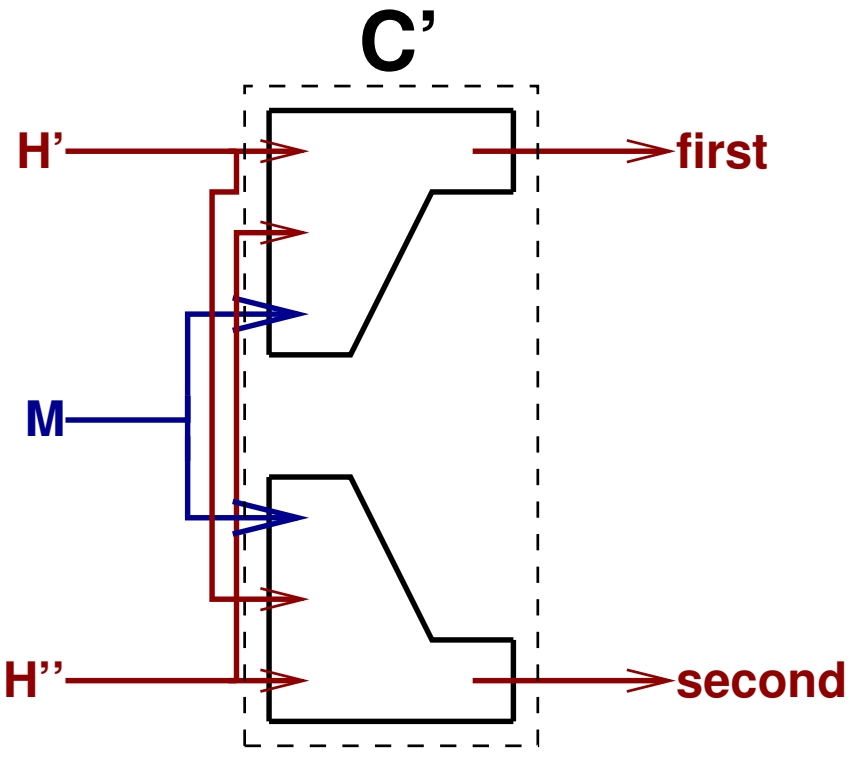
$$\mathbf{C}'(\mathbf{H}', \mathbf{H}'', \mathbf{M}) = (\mathbf{C}(\mathbf{H}', \mathbf{H}'', \mathbf{M}), \mathbf{C}(\mathbf{H}'', \mathbf{H}', \mathbf{M}))$$

- **Another assumption** on $\mathbf{C}$:

Special collision resistance of $\mathbf{C}'$:
It is **hard** to find

$$\mathbf{C}'(\overbrace{\mathbf{G}', \mathbf{G}''}^{\mathbf{G}' \neq \mathbf{G}''}, \mathbf{M}) = \mathbf{C}'(\overbrace{\mathbf{H}', \mathbf{H}''}^{\mathbf{H}' \neq \mathbf{H}''}, \mathbf{N})$$

(also weaker than collision resistance for $\mathbf{C}$).



Double-Pipe Hash: Security (3)

The **two new assumptions** are new but natural for a compression function **C**.

If we model **C** as a random oracle, breaking either assumption takes

$$2^{\mathbf{n}} \text{ oracle queries.}$$

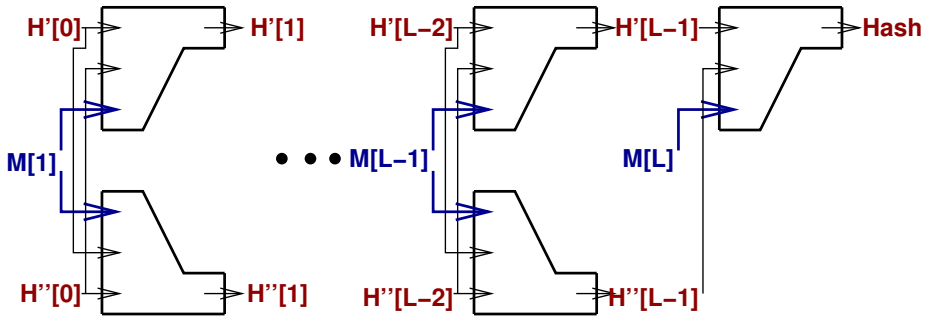
Double-Pipe Hash: Security (3)

The **two new assumptions** are new but natural for a compression function **C**.

If we model **C** as a random oracle, breaking either assumption takes

$$2^{n} \text{ oracle queries.}$$

Under these **two new assumptions** for **C**, we get similar **security results** for the Double-Pipe hash than for its Wide-Pipe counterpart.



Related Work: RIPEMD

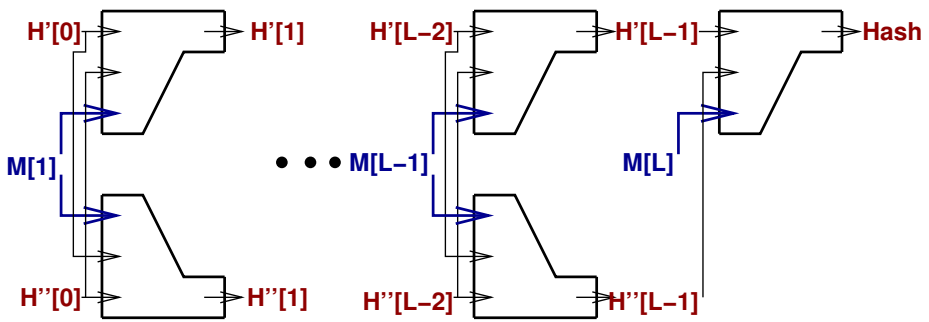
Double-Size variants of RIPEMD-128 and RIPEMD-160:

- very similar to our wide-pipe approach – *but*
  - **2n** size hash
  - no formal analysis

Related Work: RIPEMD

Double-Size variants of RIPEMD-128 and RIPEMD-160:

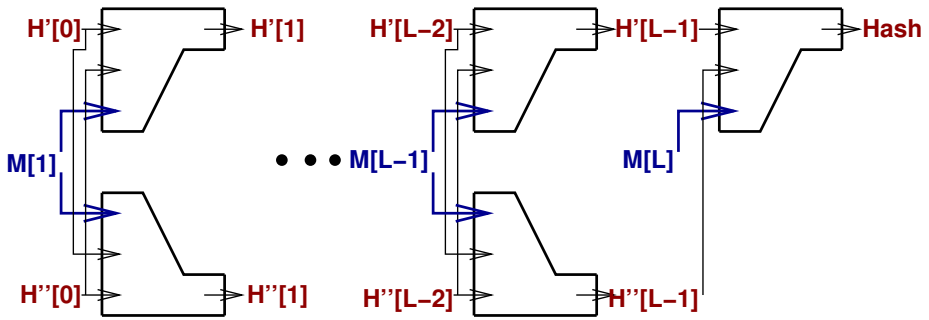
- very similar to our wide-pipe approach – *but*
  - **2n** size hash
  - no formal analysis
- on the other hand:
  - hash size can be reduced to **n** bit



Related Work: RIPEMD

Double-Size variants of RIPEMD-128 and RIPEMD-160:

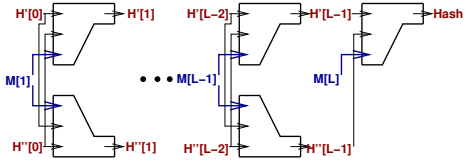
- very similar to our wide-pipe approach – *but*
  - **2n** size hash
  - no formal analysis
  - convenience feature – **no claims to improve security**
- on the other hand:
  - hash size can be reduced to **n** bit
  - **formal justification to claim improved security**



Related Work: RIPEMD

Double-Size variants of RIPEMD-128 and RIPEMD-160:

- very similar to our wide-pipe approach – *but*
  - **2n** size hash
  - no formal analysis
  - convenience feature – **no claims to improve security**
  - two *different* compression functions
- on the other hand:
  - hash size can be reduced to **n** bit
  - **formal justification to claim improved security**
  - one single compression function suffices



Related Work: Recent Modes of Operation

- M. Nandi, W. Lee, K. Sakurai, S. Lee. [...] 2/3-rate double length compression function in the black box model. FSE 2005.

Related Work: Recent Modes of Operation

- M. Nandi, W. Lee, K. Sakurai, S. Lee. [...] 2/3-rate double length compression function in the black box model. FSE 2005.

→ A possible instantiation for $\mathbf{C}'$, to be used for the wide-pipe hash.

Related Work: Recent Modes of Operation

- M. Nandi, W. Lee, K. Sakurai, S. Lee. [...] 2/3-rate double length compression function in the black box model. FSE 2005.

→ A possible instantiation for $\mathbf{C}'$, to be used for the wide-pipe hash.

- J. Coron, Y. Dodis, C.Malinaud, P. Punyia. Merkle-Damgård revisited: how to construct a hash function. Crypto 2005.

Related Work: Recent Modes of Operation

- M. Nandi, W. Lee, K. Sakurai, S. Lee. [...] 2/3-rate double length compression function in the black box model. FSE 2005.

→ A possible instantiation for $\mathbf{C}'$, to be used for the wide-pipe hash.

- J. Coron, Y. Dodis, C.Malinaud, P. Punyia. Merkle-Damgård revisited: how to construct a hash function. Crypto 2005.

→ Analysis includes the wide-pipe hash.

Related Work: Recent Modes of Operation

- M. Nandi, W. Lee, K. Sakurai, S. Lee. [...] 2/3-rate double length compression function in the black box model. FSE 2005.

→ A possible instantiation for $\mathbf{C}'$, to be used for the wide-pipe hash.

- J. Coron, Y. Dodis, C.Malinaud, P. Punyia. Merkle-Damgård revisited: how to construct a hash function. Crypto 2005.

→ Analysis includes the wide-pipe hash.

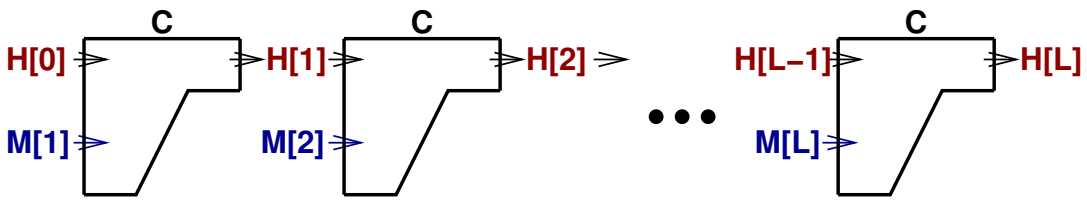
→ **But:** entirely different security goal.:

our goal: failure-tolerance
"secure even if compression function slightly fails".

their goal: improved security if compression function is ideal.

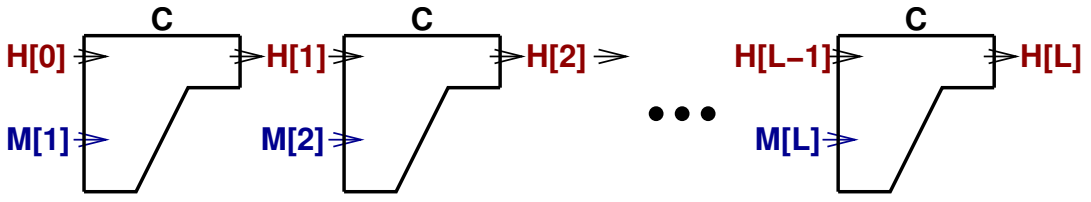
Performance Issues

- Merkle-Damgård, calling **C** *once / message block*:

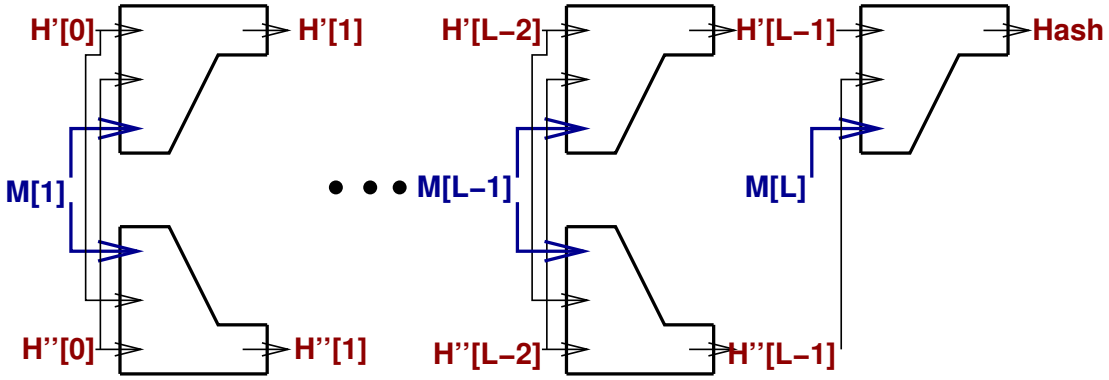


Performance Issues

- Merkle-Damgård, calling **C** *once / message block*:



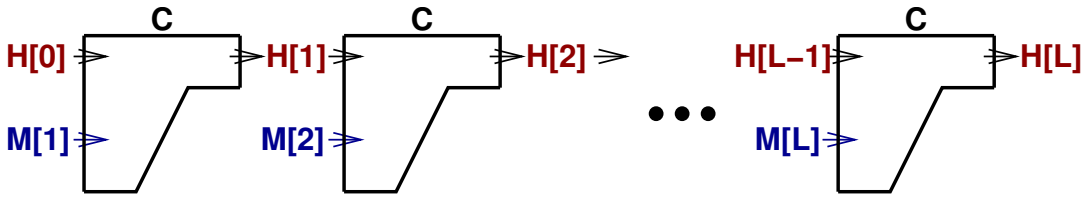
- the double-pipe hash, calling **C** *twice / message block*:



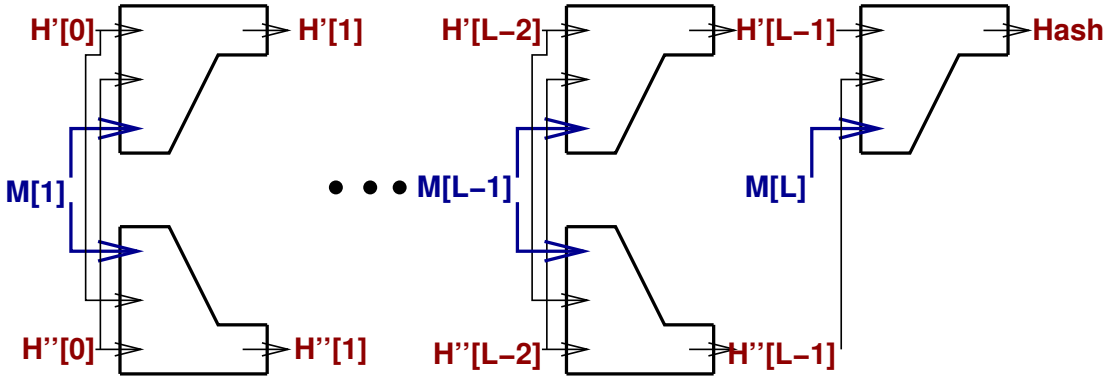
  - *message blocks smaller* (by **n** bit)

Performance Issues

- Merkle-Damgård, calling **C** *once / message block*:



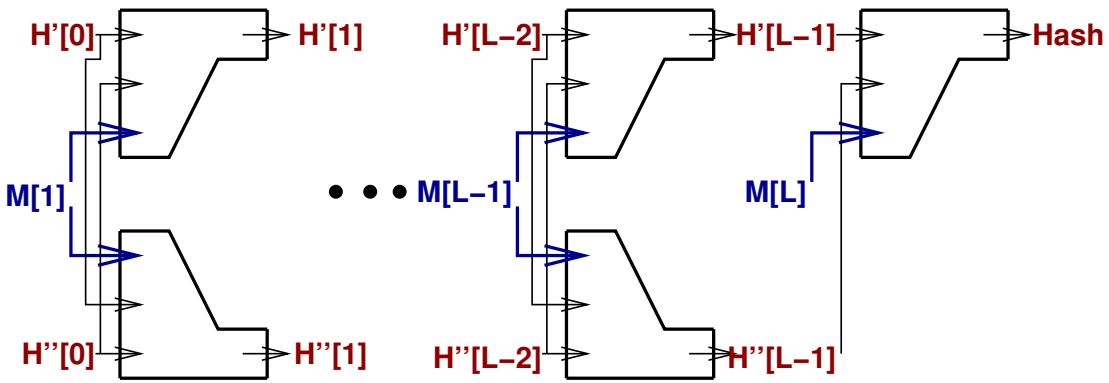
- the double-pipe hash, calling **C** *twice / message block*:



  - *message blocks smaller* (by **n** bit)
  - example SHA-256: *four times slower* than Merkle-Damgård

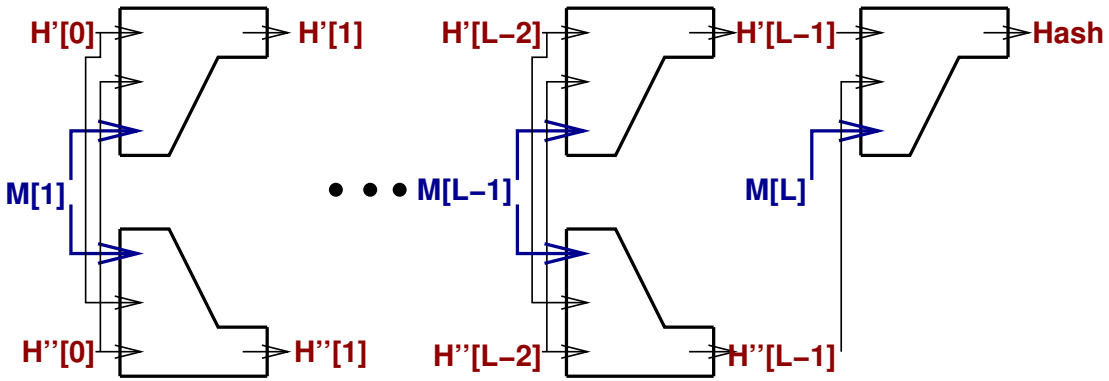
Different Modes for Different Applications

- **one** (hopefully secure) compression function **C**.
- the double-pipe hash as a failure-tolerant mode for normal app's:

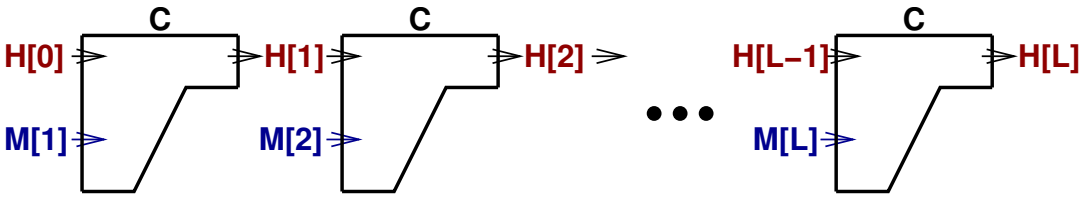


Different Modes for Different Applications

- **one** (hopefully secure) compression function **C**.
- the double-pipe hash as a failure-tolerant mode for normal app's:



- Merkle-Damg. as a high-performant but failure-intolerant mode:



Final Remarks

suggestions:

- separate primitive (compression function)
  from its mode of operation ("design principe")

proposals:

challenges:

Final Remarks

suggestions:

- separate primitive (compression function)
  from its mode of operation ("design principe")
- construct failure-tolerant hash functions
  (if the primitive "slightly fails")

proposals:

challenges:

Final Remarks

suggestions:

- separate primitive (compression function)
  from its mode of operation ("design principe")
- construct failure-tolerant hash functions
  (if the primitive "slightly fails")

proposals:

- two modes (wide-pipe and double-pipe), provably failure-tolerant

challenges:

Final Remarks

suggestions:

- separate primitive (compression function)
  from its mode of operation ("design principe")
- construct failure-tolerant hash functions
  (if the primitive "slightly fails")

proposals:

- two modes (wide-pipe and double-pipe), provably failure-tolerant

challenges:

- other modes of operation

Final Remarks

suggestions:

- separate primitive (compression function)
  from its mode of operation ("design principe")
- construct failure-tolerant hash functions
  (if the primitive "slightly fails")

proposals:

- two modes (wide-pipe and double-pipe), provably failure-tolerant

challenges:

- other modes of operation
- another kind of failure-tolerance:
  - two or more compression functions $\mathbf{C_1}$, $\mathbf{C_2}$, ..., no cascade
  - provably secure if at least one of them is secure.