

# Failure-Tolerance: A Design Principle for Iterated Hashfunctions

Krakow, June 2005

Stefan Lucks

University of Mannheim  
Germany

# Roadmap

## Introduction

- Current Iterated Hashfunctions (Merkle-Damgård)

- The Bright Side

- The Dark Side

- A new View at Hash Functions?

## Failure-Tolerant Iterated Hash Functions

- The Wide-Pipe Hash

- The Double-Pipe Hash

- Related Work

## Discussion

- Performance Issues

- Different Modes for Different Applications

- Final Remarks

## Introduction

Current Iterated Hashfunctions (Merkle-Damgård)

The Bright Side

The Dark Side

A new View at Hash Functions?

## Failure-Tolerant Iterated Hash Functions

The Wide-Pipe Hash

The Double-Pipe Hash

Related Work

## Discussion

Performance Issues

Different Modes for Different Applications

Final Remarks

# Current Iterated Hashfunctions (Merkle-Damgård)

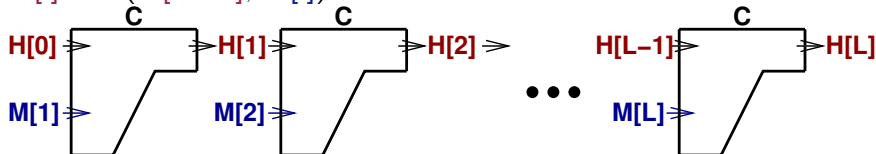
Iterate a given compression function

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

(input size  $n + m$ )

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

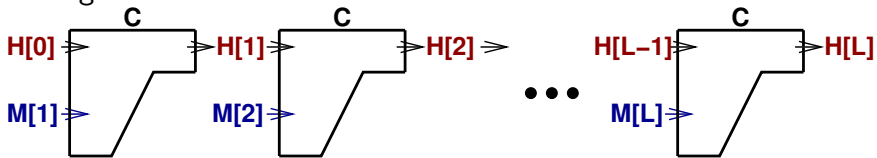
$$H[i] = C(H[i-1], M[i]):$$



# The Bright Side: A Formal Proof of Security

**If**  
finding collisions for  $C$  is infeasible,

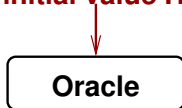
**then**  
finding collisions for  $H$  is infeasible as well.



# The Dark Side: Weakness of Compression Functions

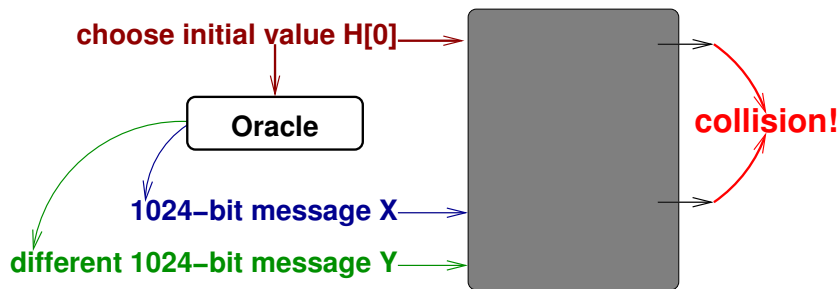
- ▶ Example: the Wang/Yu attack against MD5

**choose initial value  $H[0]$**



# The Dark Side: Weakness of Compression Functions

- ▶ Example: the Wang/Yu attack against MD5



## The Dark Side:

### Structural Weaknesses – Extending Collisions

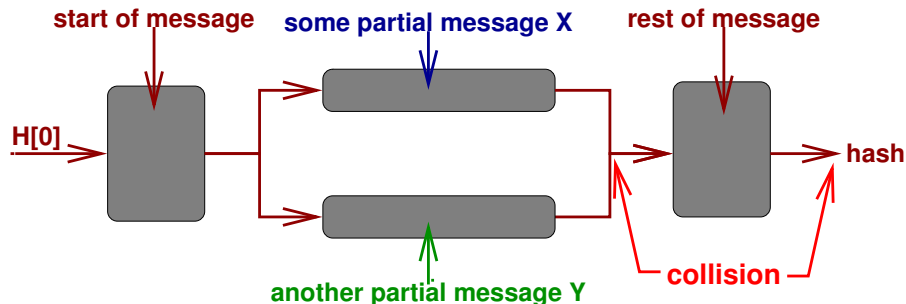
Common **start** and **rest**, random **X**  $\neq$  **Y**:

$$\text{Hash}(\text{start}, \mathbf{X}, \text{rest}) = \text{Hash}(\text{start}, \mathbf{Y}, \text{rest})$$

# The Dark Side: Structural Weaknesses – Extending Collisions

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

$$\text{Hash}(\text{start}, X, \text{rest}) = \text{Hash}(\text{start}, Y, \text{rest})$$



## The Dark Side:

# Exploiting for the Collision Extension Weakness

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

## The Dark Side:

# Exploiting for the Collision Extension 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.

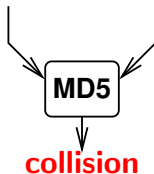
# The Dark Side: The Story of Alice and her Boss

## 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

### Introduction

Current Iterated Hashfunctions (Merkle-Damgård)

The Bright Side

The Dark Side

A new View at Hash Functions?

### Failure-Tolerant Iterated Hash Functions

The Wide-Pipe Hash

The Double-Pipe Hash

Related Work

### Discussion

Performance Issues

Different Modes for Different Applications

Final Remarks

# 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  $n < w$ )

two compression functions

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

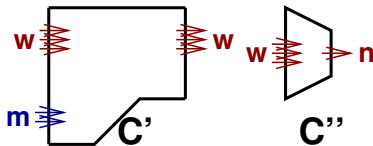
# The Wide-Pipe Hash

## Idea:

larger internal state

( $w$ -bit instead of  $n < w$ )

two compression functions



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

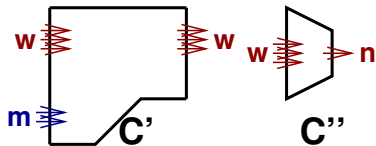
# The Wide-Pipe Hash

## Idea:

larger internal state

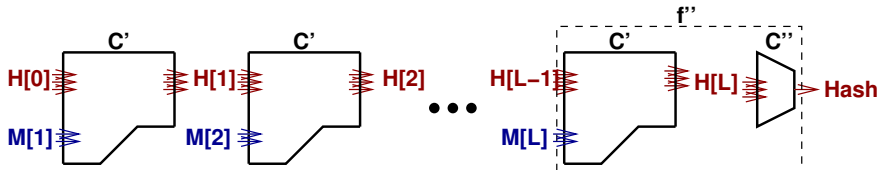
( $w$ -bit instead of  $n < w$ )

two compression functions



$$C' : \{0, 1\}^w \times \{0, 1\}^m \rightarrow \{0, 1\}^w \text{ and}$$

$$C'' : \{0, 1\}^w \rightarrow \{0, 1\}^n.$$



# The Wide-Pipe Hash

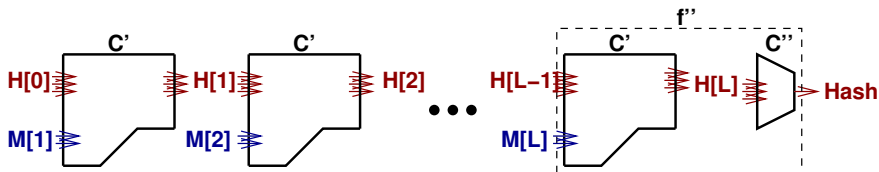
## Rationale:

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



$$C' : \{0, 1\}^w \times \{0, 1\}^m \rightarrow \{0, 1\}^w \text{ and}$$

$$C'' : \{0, 1\}^w \rightarrow \{0, 1\}^n.$$

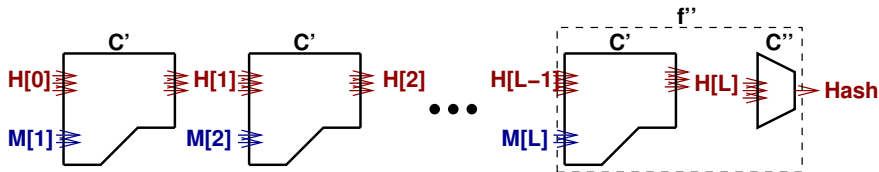


## The Wide-Pipe Hash: Security

**Results:** If  $C'$  is collision resistant, then the Wide-Pipe hash is

1. resistant against multi-collision attacks, if  $f''$  is so,

All results hold, *even if  $f''$  is not collision resistant!*



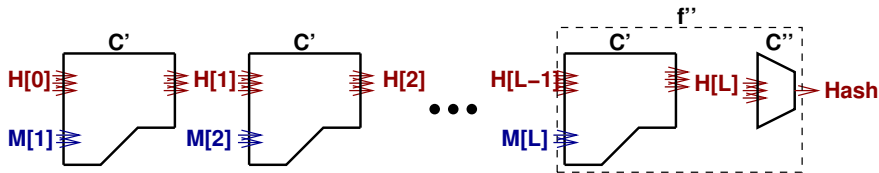
## The Wide-Pipe Hash: Security

**Results:** If  $C'$  is collision resistant, then the Wide-Pipe hash is

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

All results hold, *even if  $f''$  is not collision resistant!*

First two results: proof of security in standard model.



## The Wide-Pipe Hash: Security

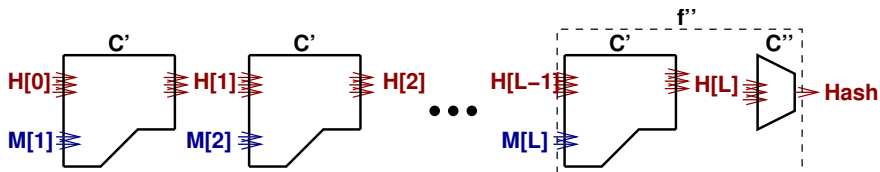
**Results:** If  $C'$  is collision resistant, then the Wide-Pipe hash is

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

All results hold, *even if  $f''$  is not collision resistant!*

First two results: proof of security in standard model.

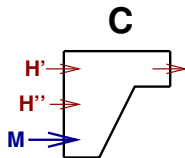
Third result: proof of security in random oracle model.



# The Double-Pipe Hash

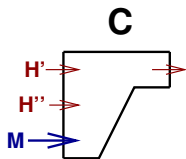
Idea: Realise  $\mathbf{C}'$  and  $\mathbf{f}''$  by one single compression function

$$\mathbf{C} : \dots \rightarrow \{0, 1\}^n.$$



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

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



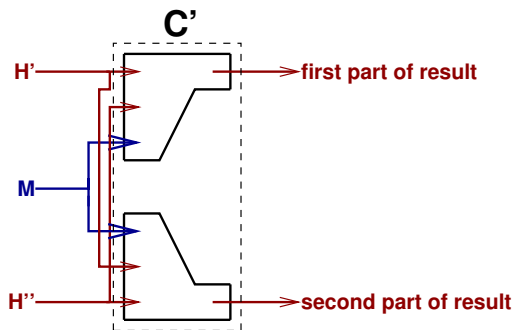
## Failure-Tolerance for Hash Functions

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

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

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

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



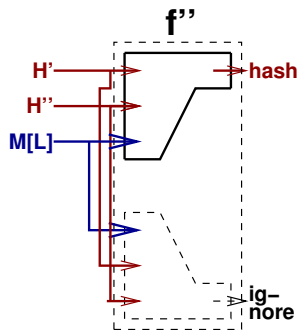
## Failure-Tolerance for Hash Functions

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

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

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

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



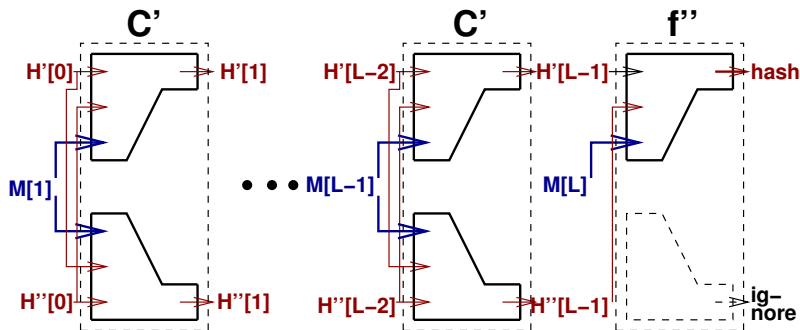
Realise  $\mathbf{C}'$  and  $\mathbf{f}''$  by one single compression function

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

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

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

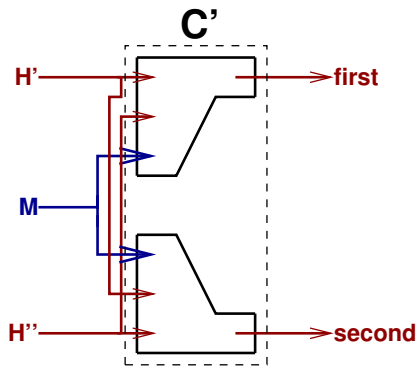
The full hash function:



## Double-Pipe Hash: Security

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

$$C'(H', H'', M) = (C(H', H'', M), C(H'', H', M))$$



## Double-Pipe Hash: Security

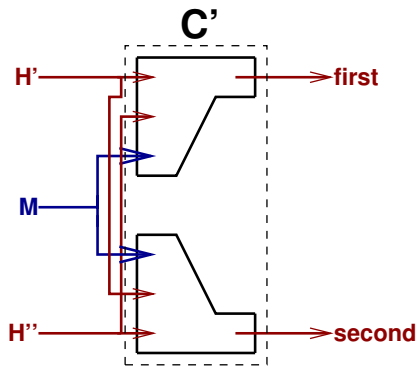
- ▶ Double-Pipe as a specific instantiation of Wide-Pipe?

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

- ▶ Set  $\mathbf{H}' = \mathbf{H}''$ .

Finding collisions for  $\mathbf{C}'$

as easy as finding them for  $\mathbf{C}$ .



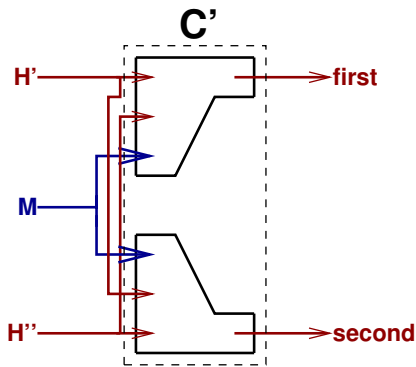
## Double-Pipe Hash: Security

- ▶ 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

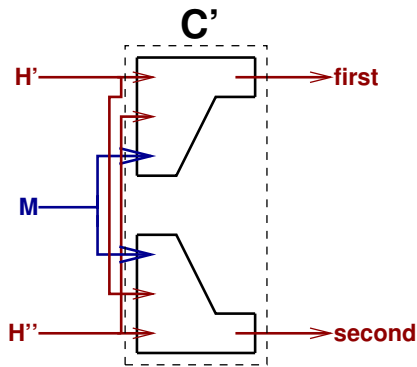
- ▶ Double-Pipe as a specific instantiation of Wide-Pipe?

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

- ▶ 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?

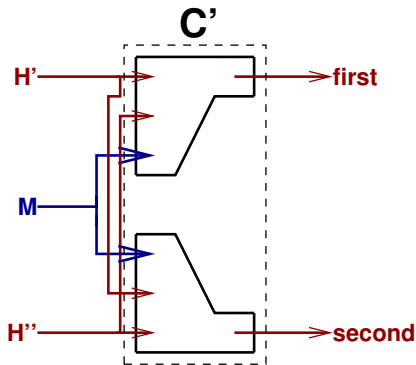
$$\mathbf{C}'(\mathbf{H}', \mathbf{H}'', \mathbf{M}) = (\mathbf{C}(\mathbf{H}', \mathbf{H}'', \mathbf{M}), \mathbf{C}(\mathbf{H}'', \mathbf{H}', \mathbf{M}))$$

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

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

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

(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

$2^n$  oracle queries.

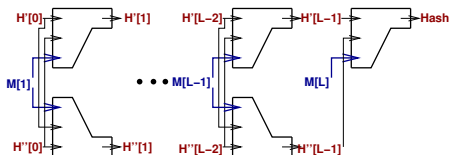
## Double-Pipe Hash: Security (3)

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

$2^n$  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

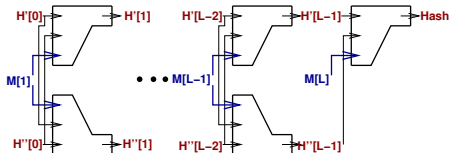
### Double-Size variants of RIPEMD-128 and RIPEMD-160:

- ▶ 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:

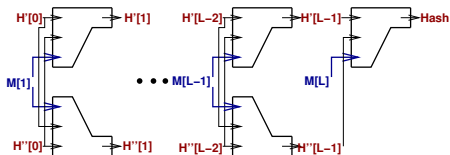
- ▶ very similar to our wide-pipe approach – *but*
  - ▶ **2n** size hash
  - ▶ no formal analysis
- ▶ on the other hand:
  - ▶ hash size can be reduced to **n** bit



## Related Work: RIPEMD

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

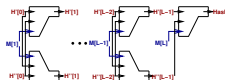
- ▶ very similar to our wide-pipe approach – *but*
  - ▶ **2n** size hash
  - ▶ no formal analysis
  - ▶ convenience feature – **no claims to improve security**
- ▶ on the other hand:
  - ▶ hash size can be reduced to **n** bit
  - ▶ **formal justification to claim improved security**



## 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.

## Failure-Tolerance for Hash Functions

### Introduction

Current Iterated Hashfunctions (Merkle-Damgård)

The Bright Side

The Dark Side

A new View at Hash Functions?

### Failure-Tolerant Iterated Hash Functions

The Wide-Pipe Hash

The Double-Pipe Hash

Related Work

### Discussion

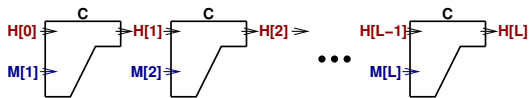
Performance Issues

Different Modes for Different Applications

Final Remarks

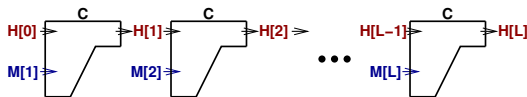
# Performance Issues

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

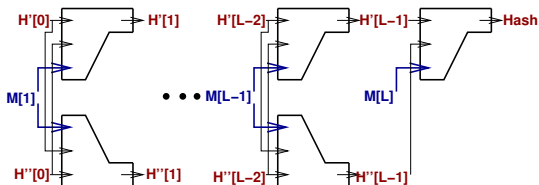


## Performance Issues

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



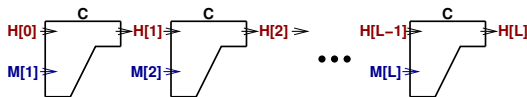
- ▶ the double-pipe hash, calling **C** *twice* / *message block*:



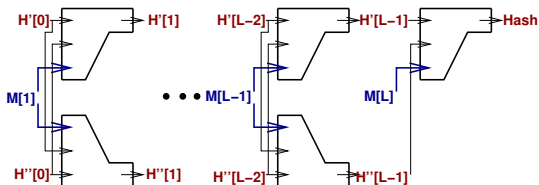
- ▶ *message blocks smaller* (by **n** bit)

## Performance Issues

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



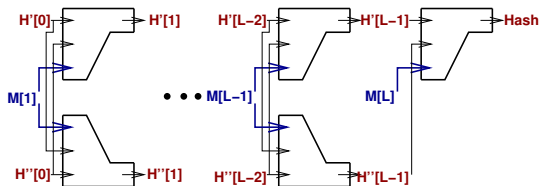
- ▶ the double-pipe hash, calling **C** *twice / message block*:



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

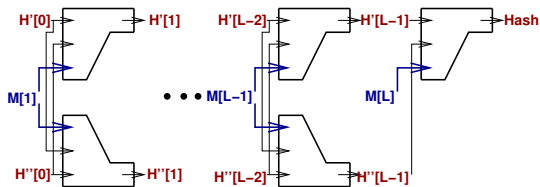
# Different Modes for Different Applications

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

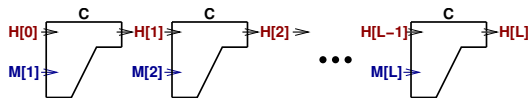


# Different Modes for Different Applications

- ▶ **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 principle”)

### proposals:

### challenges:

## Final Remarks

### suggestions:

- ▶ separate primitive (compression function) from its mode of operation (“design principle”)
- ▶ 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 principle”)
- ▶ 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 principle”)
- ▶ 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 principle”)
- ▶ 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  $\mathbf{C}_1, \mathbf{C}_2, \dots$ , no cascade
  - provably secure if at least one of them is secure.