Cryptology 4

(Hash functions, MACs)

Georg Fuchsbauer

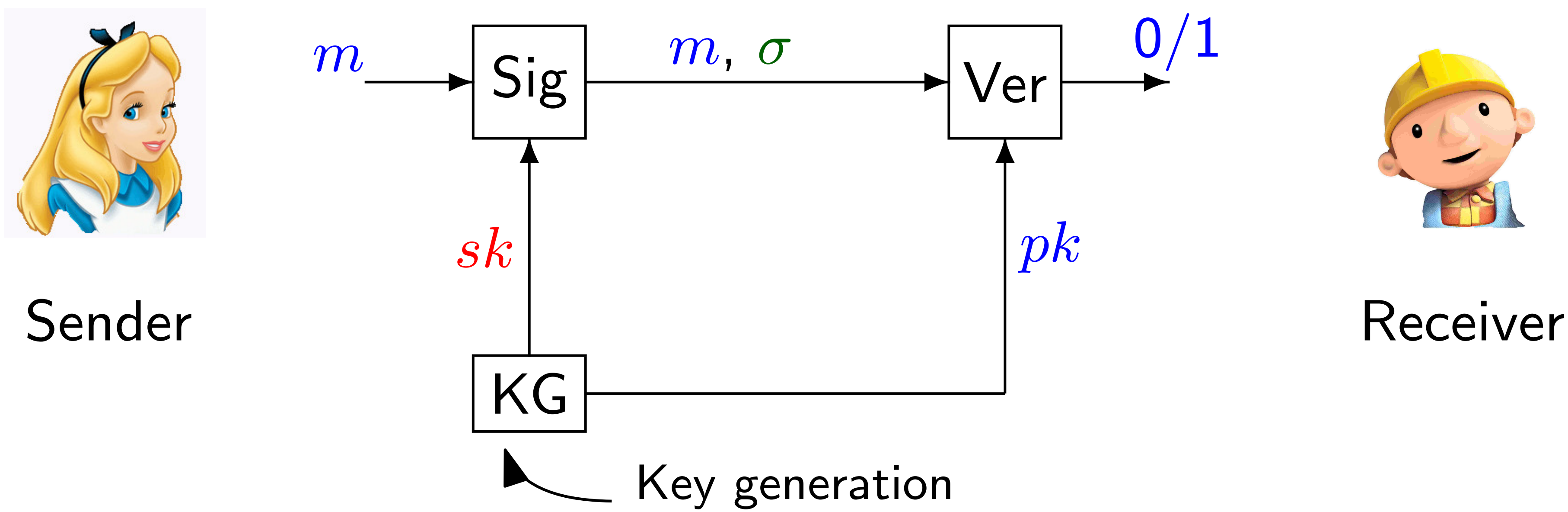
www.di.ens.fr/~fuchsbau/cryptoESILV4.pdf

ESILV Feb-Mar 2019

Hash functions

Motivation

Recall: **Digital signatures**



provide:

- sender authenticity
- message integrity
- **non-repudiation**

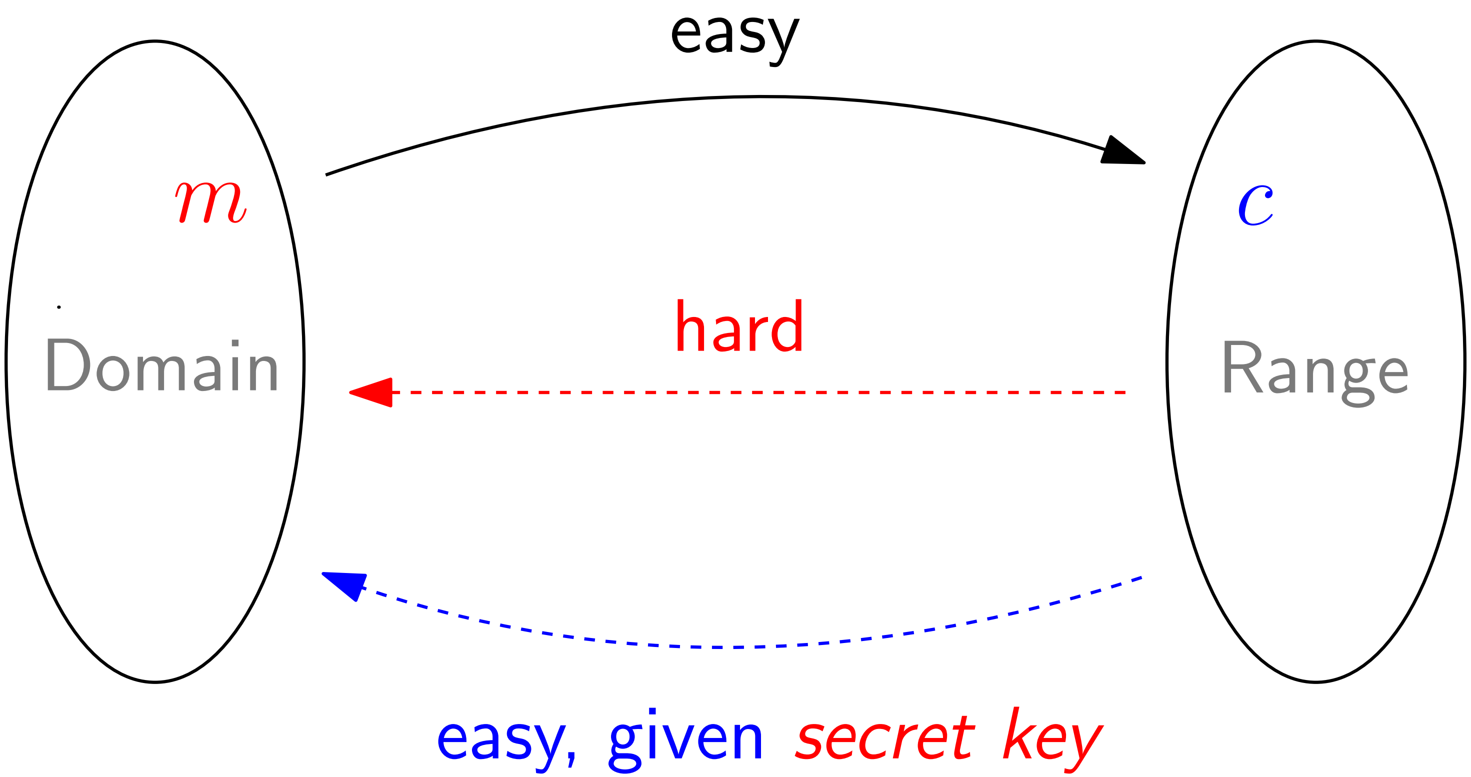
Motivation

Recall: (textbook) **RSA encryption**

public key: (N, e)

private key: d

$(= e^{-1} \pmod{\phi(N)})$



Motivation

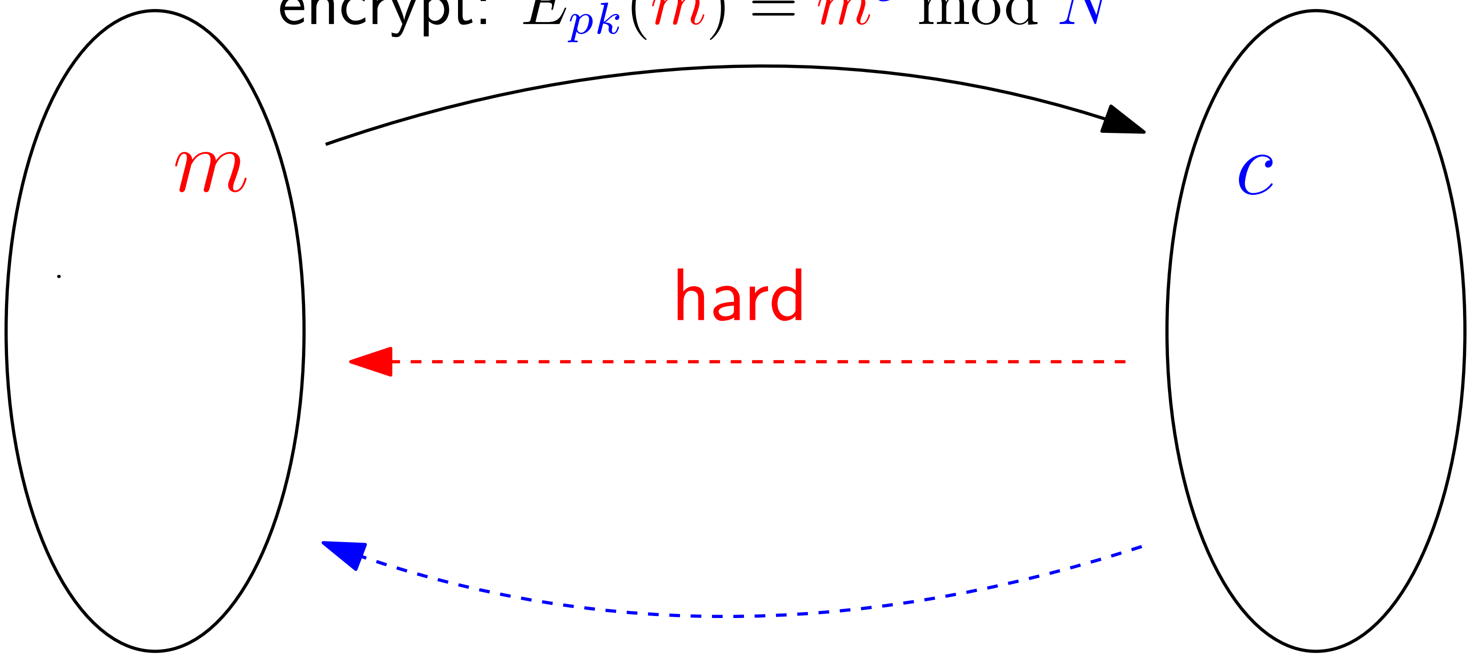
Recall: (textbook) **RSA encryption**

public key: (N, e)

private key: d

$(= e^{-1} \pmod{\phi(N)})$

encrypt: $E_{pk}(m) = m^e \bmod N$



decrypt: $D_d(c) = c^d \bmod N$

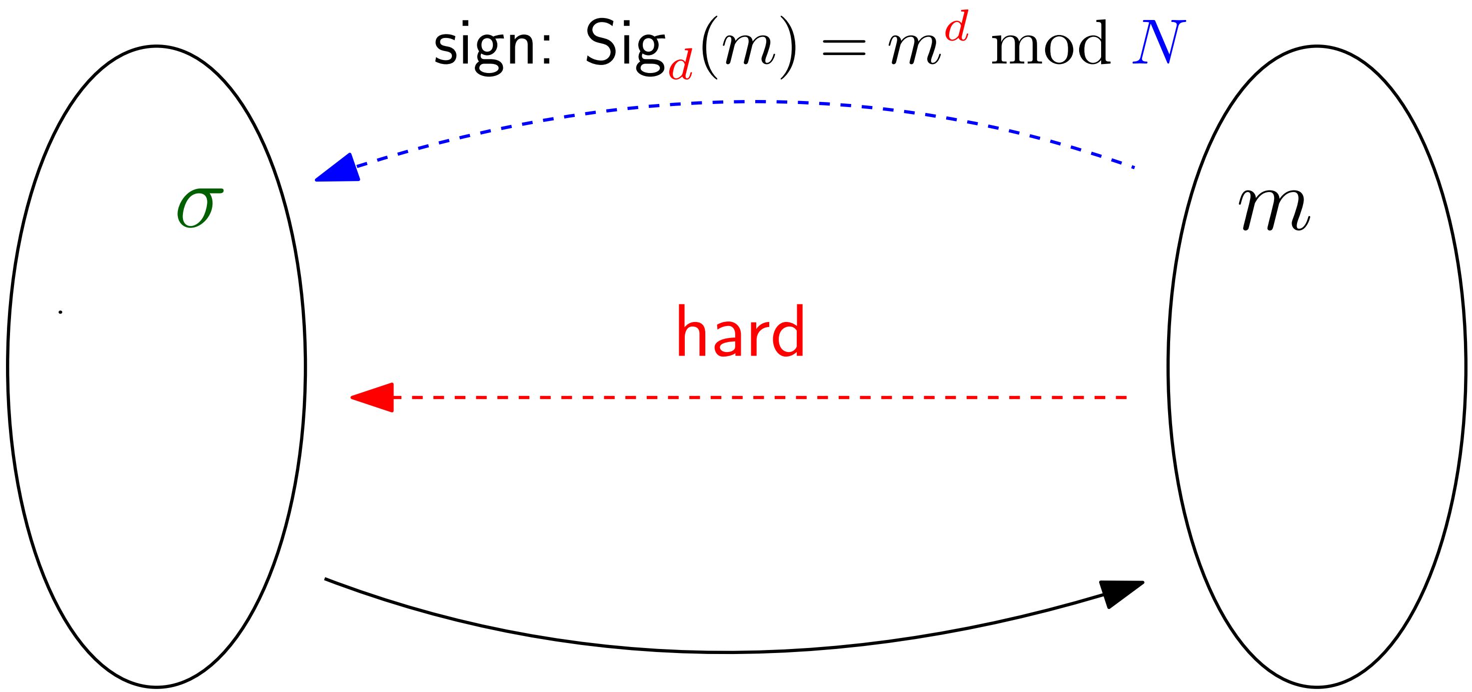
Motivation

Recall: (textbook) **RSA signature**

public key: (N, e)

private key: d

$(= e^{-1} \pmod{\phi(N)})$



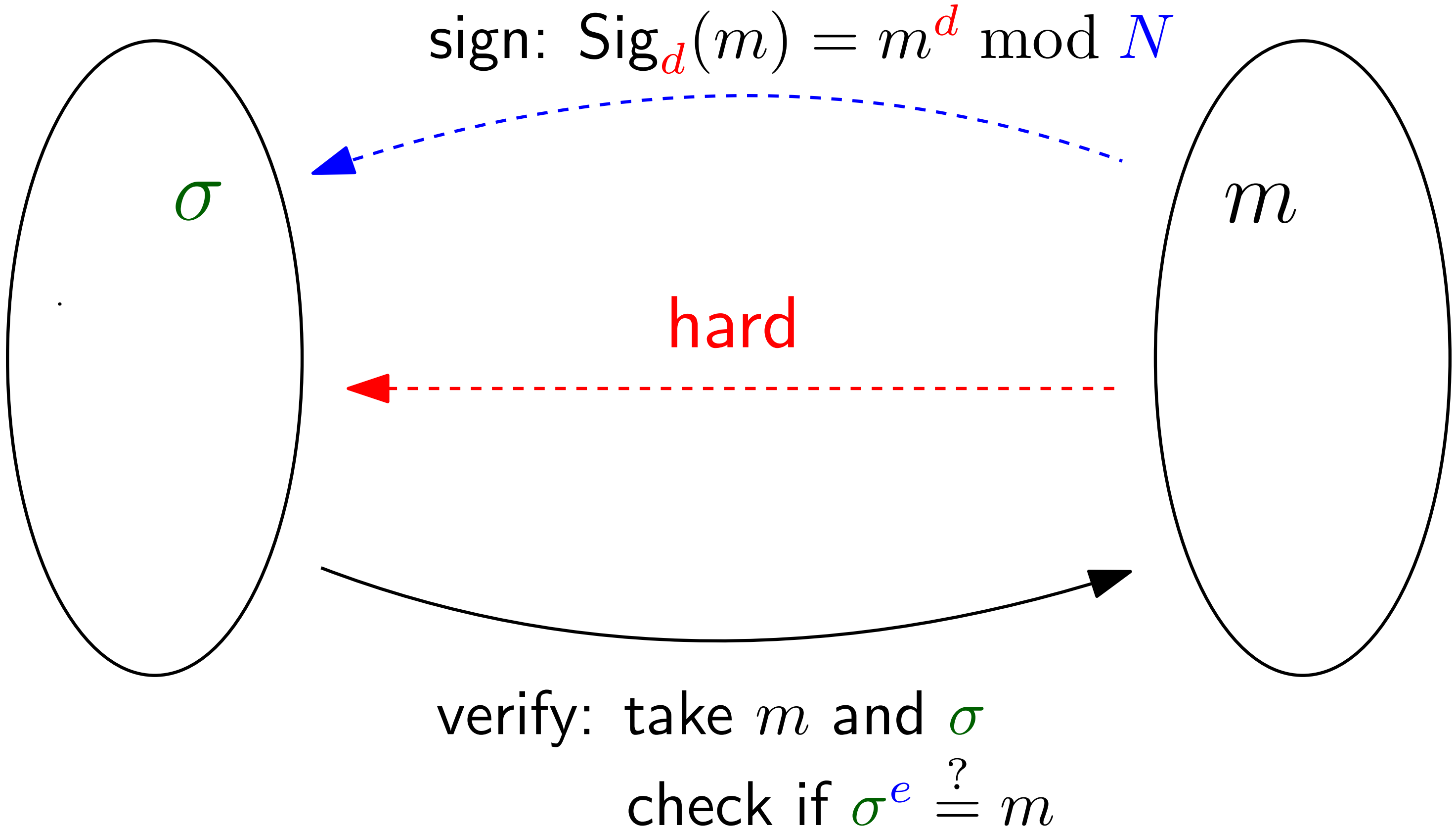
Motivation

Recall: (textbook) **RSA signature**

public key: (N, e)

private key: d

$(= e^{-1} \pmod{\phi(N)})$

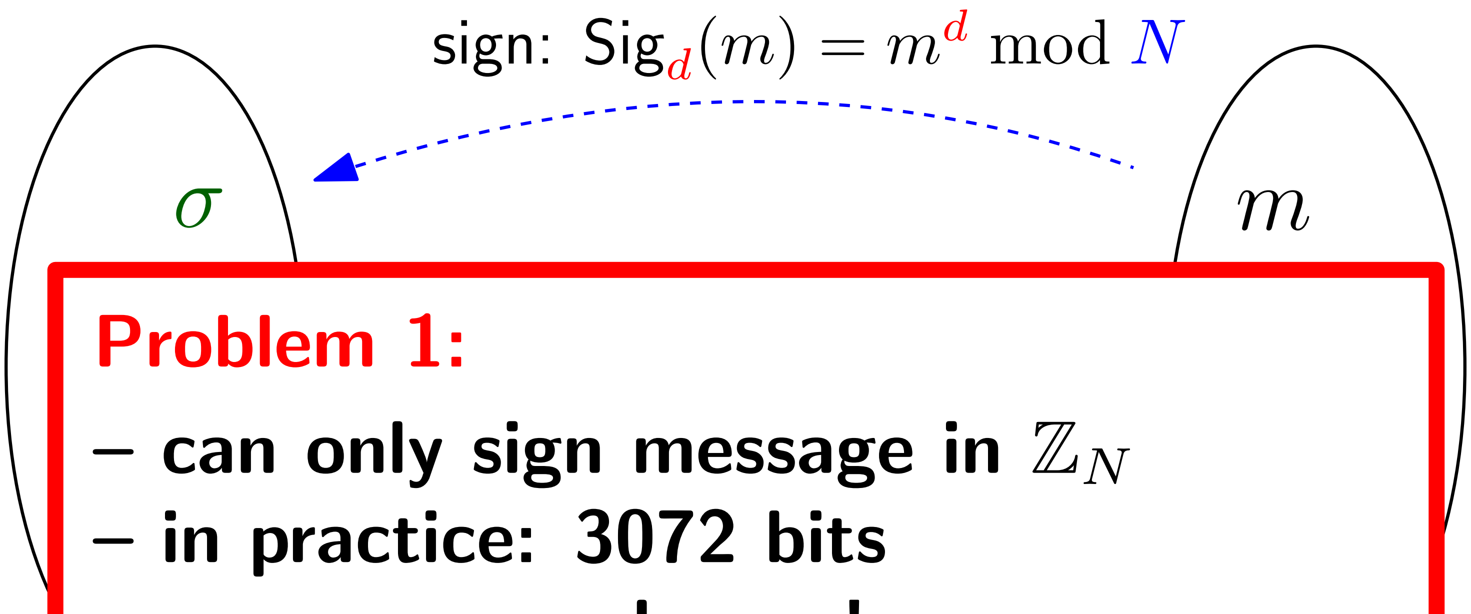


Motivation

Recall: (textbook) **RSA signature**

public key: (N, e)

private key: d
$(= e^{-1} \pmod{\phi(N)})$



Problem 1:

- **can only sign message in $\mathbb{Z}_N$**
- **in practice: 3072 bits**
- **messages are longer!**

Motivation

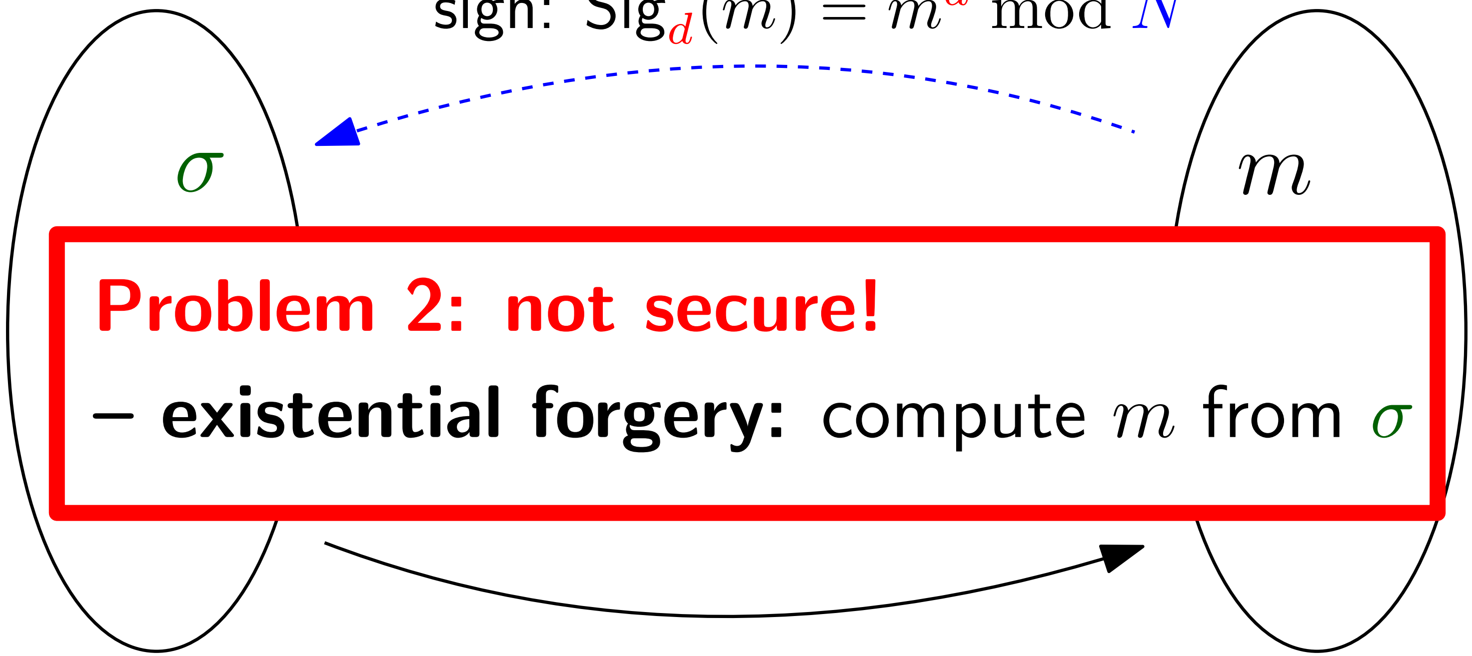
Recall: (textbook) **RSA signature**

public key: (N, e)

private key: d

$(= e^{-1} \pmod{\phi(N)})$

sign: $\mathsf{Sig}_d(m) = m^d \bmod N$



Problem 2: not secure!

– **existential forgery:** compute m from σ

verify: take m and σ

check if $\sigma^e \stackrel{?}{=} m$

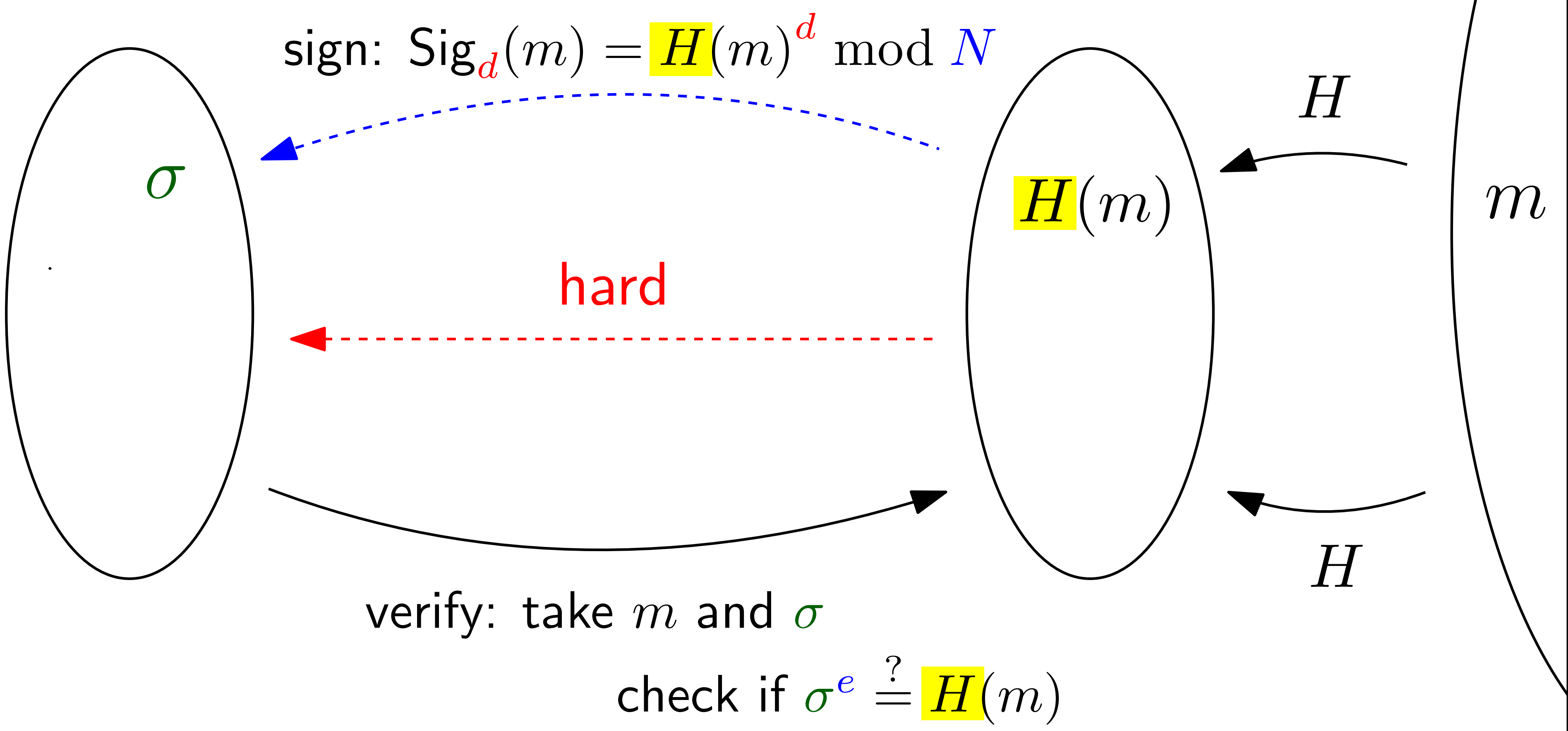
Motivation

Recall: (textbook) **RSA signature**

public key: (N, e)

private key: d

$(= e^{-1} \pmod{\phi(N)})$



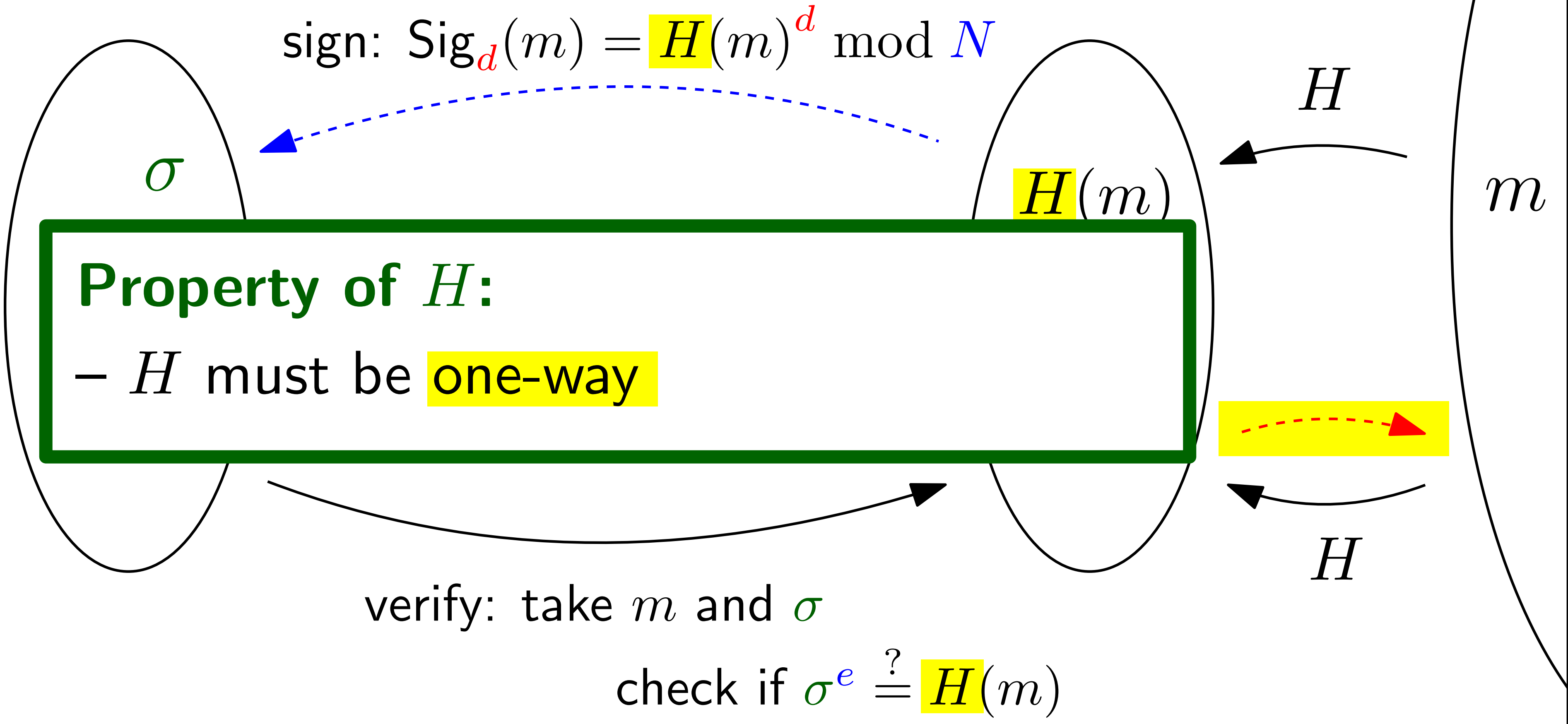
Motivation

Recall: (textbook) **RSA signature**

public key: (N, e)

private key: d

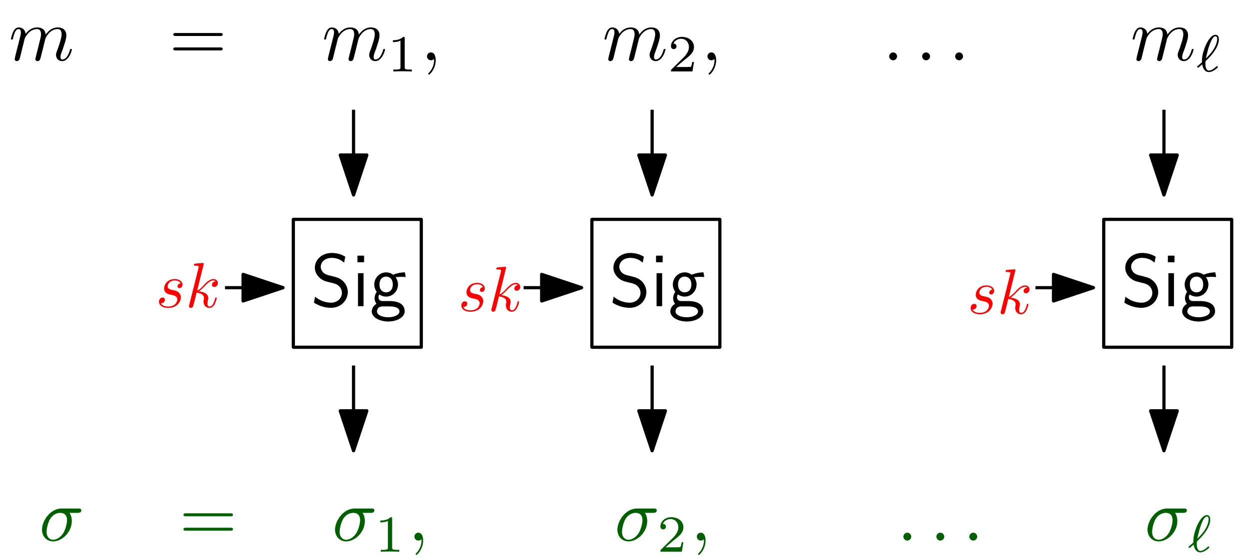
$(= e^{-1} \pmod{\phi(N)})$



Property of H:

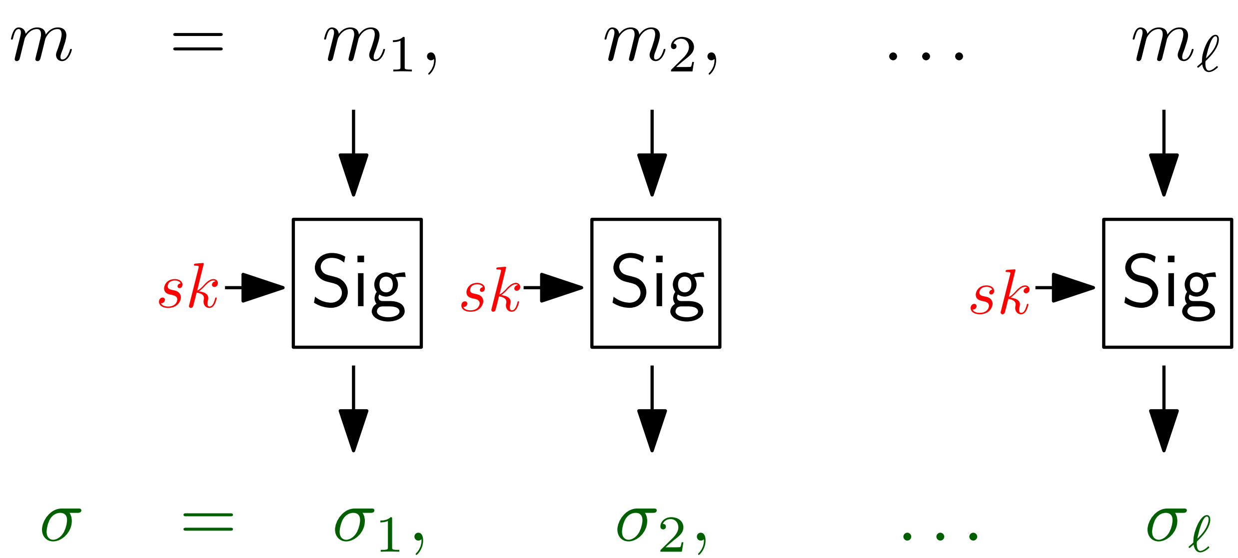
– H must be one-way

Bad idea



why is this a **bad** idea?

Bad idea

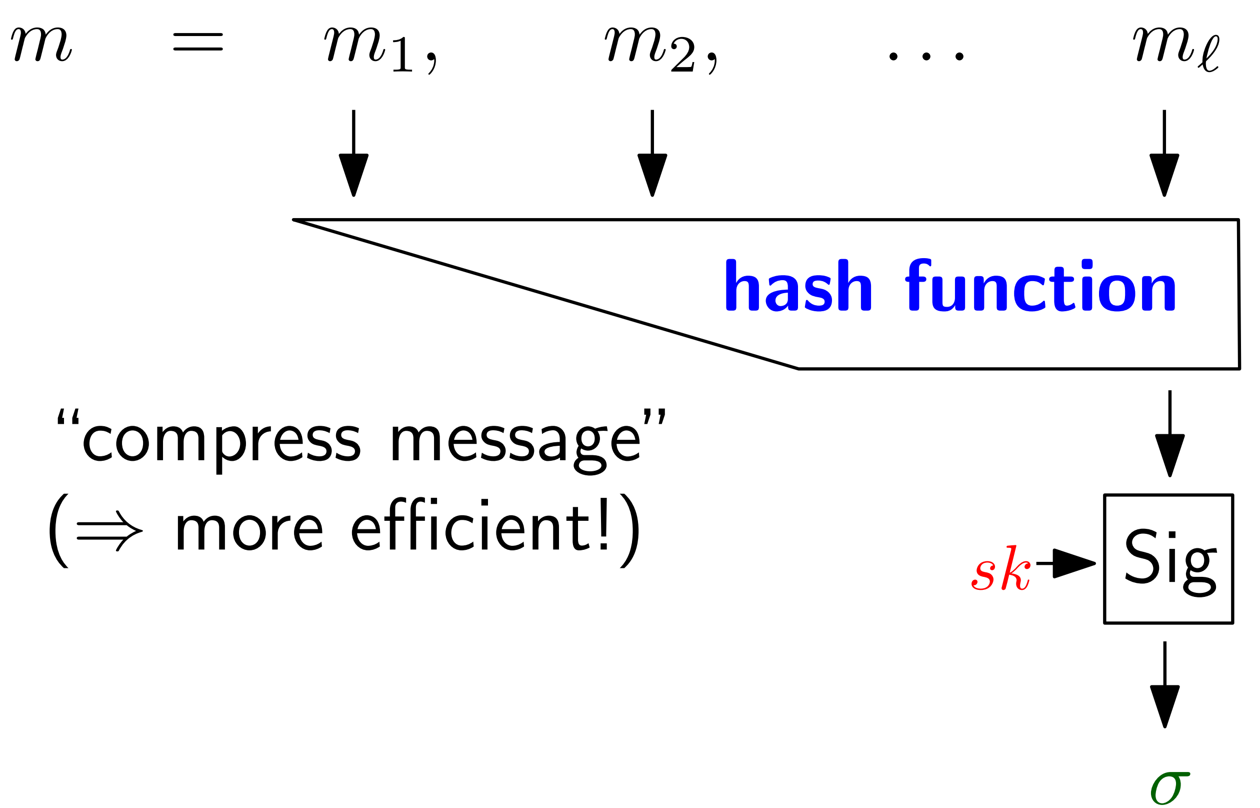


why is this a **bad** idea?

e.g. (σ_1, σ_3) is signature on (m_1, m_3)

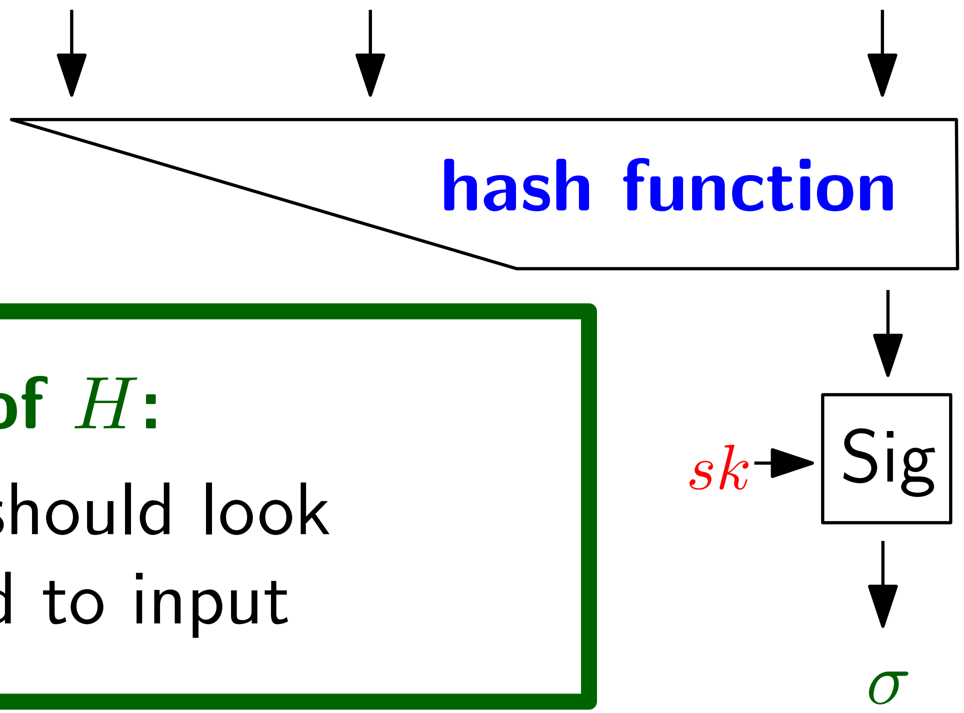
e.g. remove appendix from contract!

Better idea


$m = m_1, \quad m_2, \quad \ldots \quad m_\ell$
hash function
"compress message"
(⇒ more efficient!)
sk
Sig
σ

Better idea

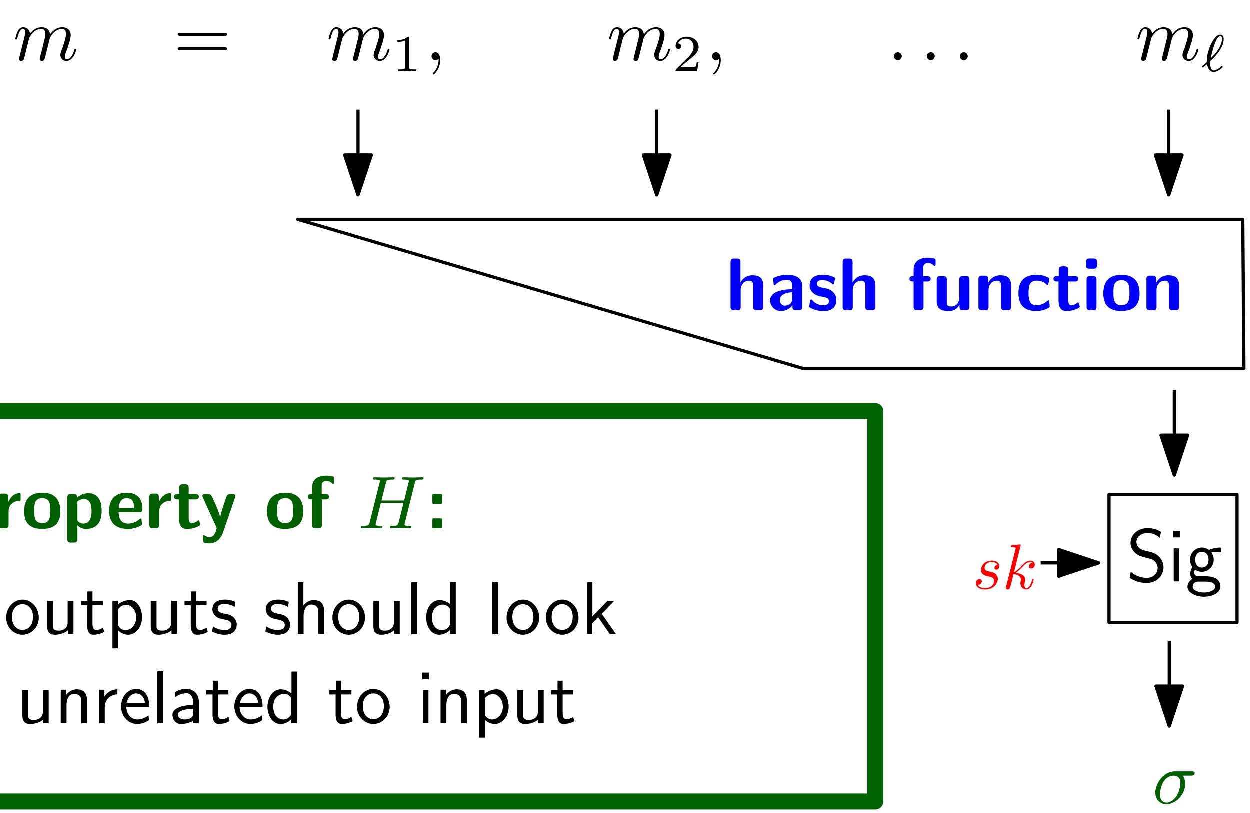
$$m \quad = \quad m_1, \quad m_2, \quad \ldots \quad m_\ell$$



Property of H:

– outputs should look unrelated to input

Better idea



Property of H:

– outputs should look unrelated to input

SHA1("The quick brown fox jumps over the lazy dog") =
2fd4e1c67a2d28fced849ee1bb76e7391b93eb12

SHA1("The quick brown fox jumps over the lazy cog") =
de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b3

An attack

transfer EUR 100 to Bob

m, σ ← signed by Alice

valid?

$$\sigma^e \stackrel{?}{\equiv} H(m) \bmod N$$

An attack

m, σ

valid?

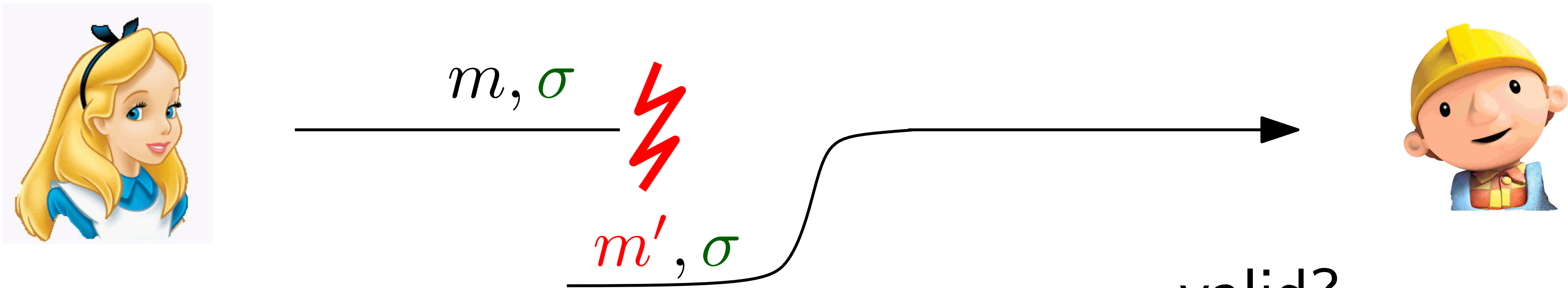
$\sigma^e \stackrel{?}{\equiv} H(m) \bmod N$

find m' s.t.

$H(m) = H(m')$

transfer EUR 100 to Beelzebot

An attack



find m' s.t.

$H(m) = H(m')$

valid?

$$\sigma^e \stackrel{?}{\equiv} H(m) \bmod N$$
$$\equiv H(m') \bmod N$$

$\Rightarrow$ σ valid for m'

An attack

m, σ

m', σ

find m' s.t.
$H(m) = H(m')$

valid?

$\sigma^e \stackrel{?}{\equiv} H(m) \bmod N$
$\equiv H(m') \bmod N$
$\Rightarrow$ σ valid for m'

Property of H:

– given m, it must be **hard** to find m':
$$H(m) = H(m')$$
"2nd-preimage resistance"

Definition

Definition: A **hash function** is a function

- taking input arbitrary-length bit strings (from $\{0, 1\}^*$)
- produce a fixed-length string as output (from $\{0, 1\}^n$)

Definition

Definition: A **hash function** is a function

- taking input arbitrary-length bit strings (from $\{0,1\}^*$)
- produce a fixed-length string as output (from $\{0,1\}^n$)

Security

- one-wayness ("preimage-resistance"):

  given y, find m such that $H(m) = y$

- 2nd-preimage resistance:

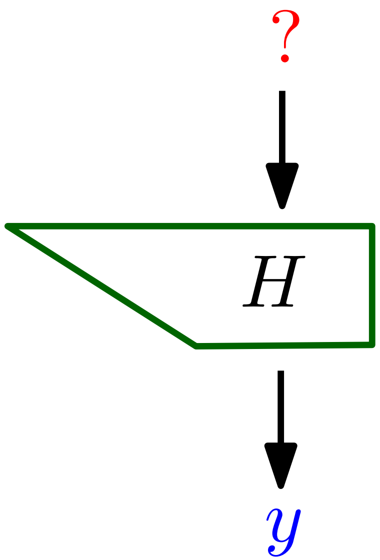
  given m_0, find m_1 such that $H(m_1) = H(m_0)$

- **collision-resistance**:

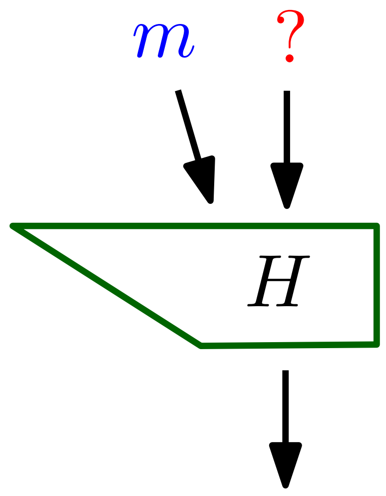
  find m_0 and m_1 such that $H(m_0) = H(m_1)$

Definition

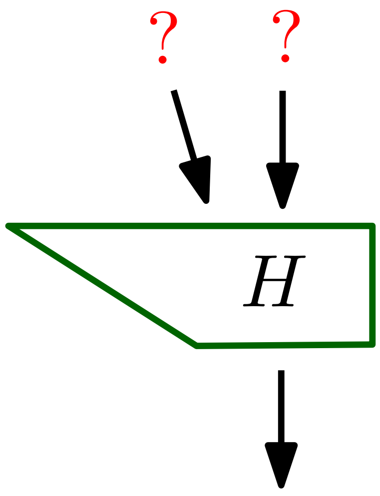
- one-wayness



- 2nd-preimage resistance

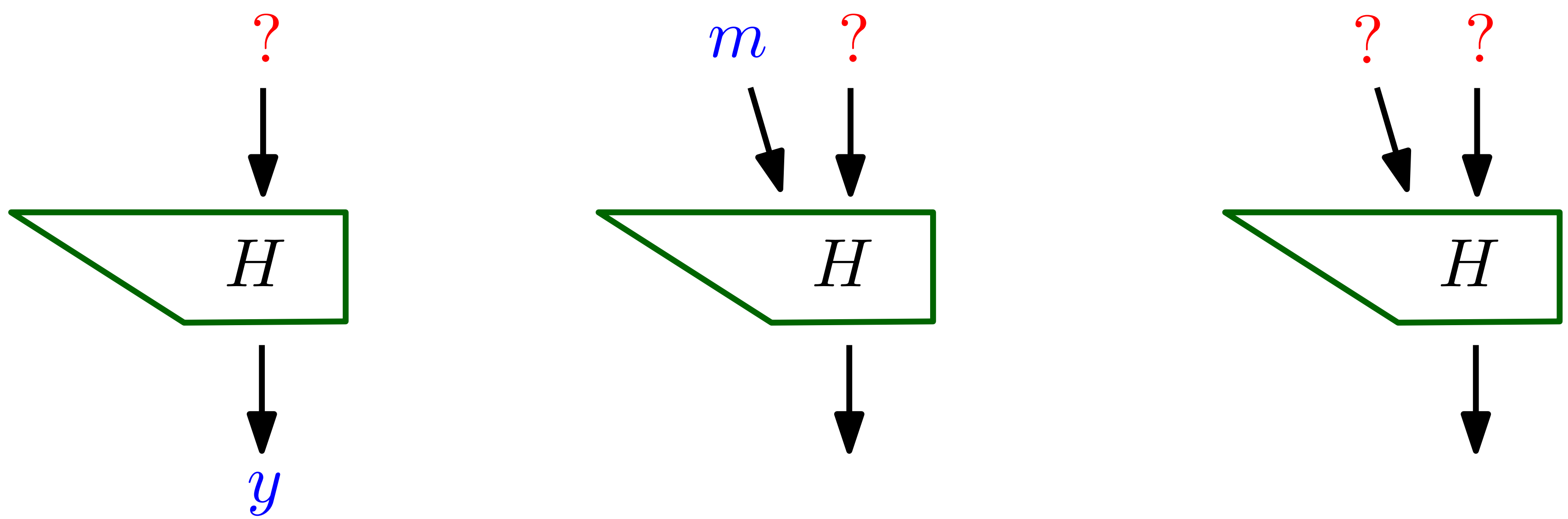


- collision-resistance



Definition

- one-wayness
- 2nd-preimage resistance
- collision-resistance



one-wayness $\Leftarrow$ 2nd-preimage resistance $\Leftarrow$ **collision-resistance**

Collision-resistance

- Collisions exist. (H maps any string to a string in $\{0,1\}^n$)
- How hard is it to find them?

Birthday "paradox"

- The probability that among 23 people two have the same birthday is $> 1/2$

Collision-resistance

- Collisions exist. (H maps any string to a string in $\{0,1\}^n$)
- How hard is it to find them?

Birthday "paradox"

- The probability that among 23 people two have the same birthday is $> 1/2$
- Why?
  - probability that 2 people have same birthday? $1/365$
  - how many pairs among q people? $\binom{q}{2} = q(q-1)/2$

Collision-resistance

- Collisions exist. (H maps any string to a string in $\{0,1\}^n$)
- How hard is it to find them?

Birthday "paradox"

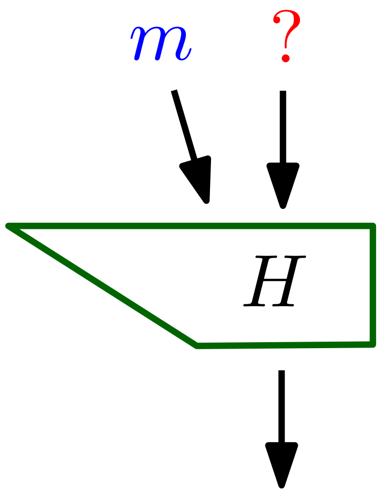
- The probability that among 23 people two have the same birthday is $> 1/2$
- Why?
  - probability that 2 people have same birthday? $1/365$
  - how many pairs among q people? $\binom{q}{2} = q(q-1)/2$
  - probablity that there is one pair $\approx \underbrace{1/365 + 1/365 + \ldots + 1/365}_{\binom{q}{2} \text{ times}}$

$$\approx q(q-1)/2 \cdot 1/[\text{nmb of days}]$$

Collision-resistance

- Collisions exist. (H maps any string to a string in $\{0,1\}^n$)
- How hard is it to find them?

- **2nd-preimage attack:**
  - given m, find m': $H(m) = H(m')$

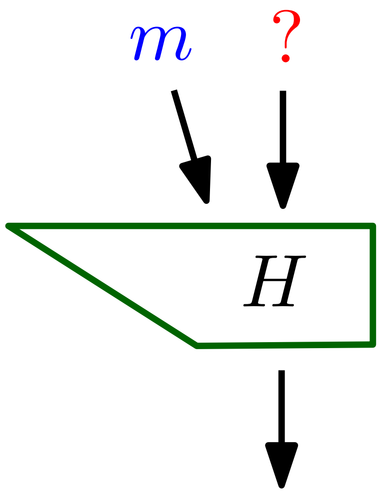


Collision-resistance

- Collisions exist. (H maps any string to a string in $\{0,1\}^n$)
- How hard is it to find them?

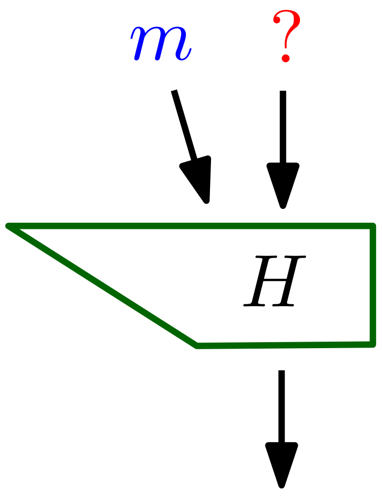
- **2nd-preimage attack:**
  - given m, find m': $H(m) = H(m')$
  - brute-force: try arbitrary m's:
    - evaluate $H(m_1)$, check $H(m_1) \stackrel{?}{=} H(m)$
    - evaluate $H(m_2)$, check $H(m_2) \stackrel{?}{=} H(m)$

      $\vdots$



Collision-resistance

- Collisions exist. (H maps any string to a string in $\{0,1\}^n$)
- How hard is it to find them?



- **2nd-preimage attack:**
  - given m, find m': $H(m) = H(m')$
  - brute-force: try arbitrary m's:
    - evaluate $H(m_1)$, check $H(m_1) \stackrel{?}{=} H(m)$
    - evaluate $H(m_2)$, check $H(m_2) \stackrel{?}{=} H(m)$

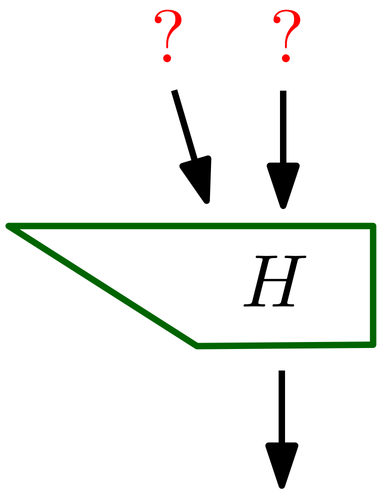
    $\vdots$

    expected complexity: 2^n

    fine for $n = 80$

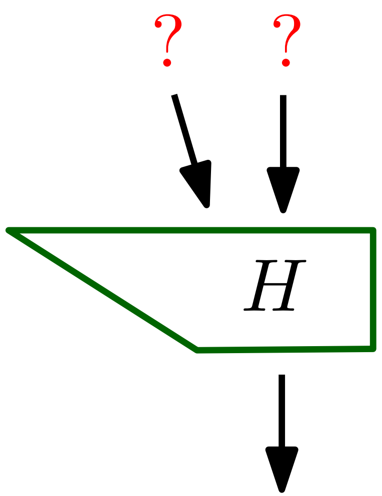
Collision-resistance

- **Collision attack:**
  - find m, m': $H(m) = H(m')$



Collision-resistance

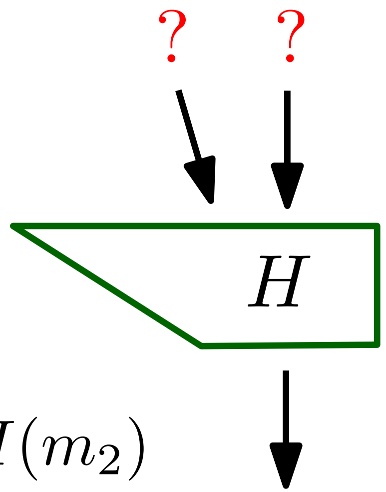
- **Collision attack:**
  - find m, m': $H(m) = H(m')$
  - brute-force: try arbitrary m's:
    - $*$ evaluate $H(m_1)$
    - $*$ evaluate $H(m_2)$, check $H(m_2) \stackrel{?}{=} H(m_1)$



Collision-resistance

- **Collision attack:**
  - find m, m': $H(m) = H(m')$
  - brute-force: try arbitrary m's:
    * evaluate $H(m_1)$
    * evaluate $H(m_2)$, check $H(m_2) \stackrel{?}{=} H(m_1)$
    * evaluate $H(m_3)$, check $H(m_3) \stackrel{?}{=} H(m_1)$ or $\stackrel{?}{=} H(m_2)$

      $\vdots$
    * evaluate $H(m_i)$, check $H(m_i) \stackrel{?}{\in} \{H(m_1), \dots, H(m_{i-1})\}$

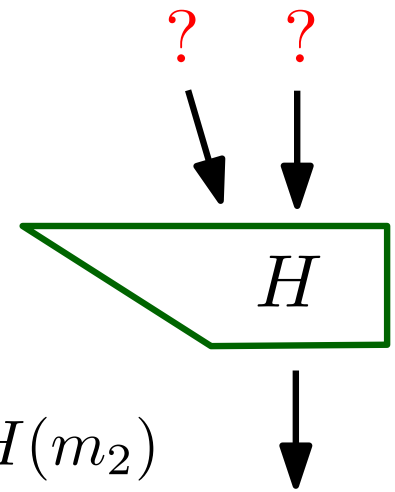


Collision-resistance

- **Collision attack:**
  - find m, m': $H(m) = H(m')$
  - brute-force: try arbitrary m's:
    - $*$ evaluate $H(m_1)$
    - $*$ evaluate $H(m_2)$, check $H(m_2) \stackrel{?}{=} H(m_1)$
    - $*$ evaluate $H(m_3)$, check $H(m_3) \stackrel{?}{=} H(m_1)$ or $\stackrel{?}{=} H(m_2)$

      $\vdots$
    - $*$ evaluate $H(m_i)$, check $H(m_i) \stackrel{?}{\in} \{H(m_1), \ldots, H(m_{i-1})\}$
  - prob. of collision after q values: $\approx q(q-1)/2 \cdot 1/[\text{nmb of hashes}]$

    (birthday paradox!)



Collision-resistance

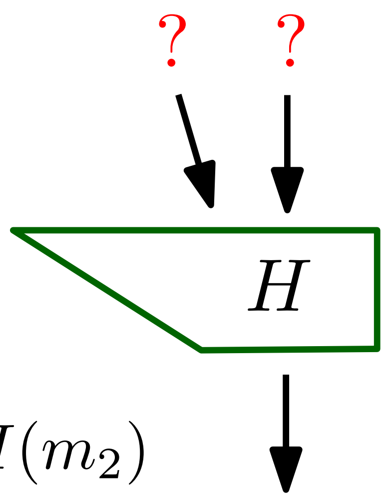
- **Collision attack:**
  - find m, m': $H(m) = H(m')$
  - brute-force: try arbitrary m's:
    - evaluate $H(m_1)$
    - evaluate $H(m_2)$, check $H(m_2) \stackrel{?}{=} H(m_1)$
    - evaluate $H(m_3)$, check $H(m_3) \stackrel{?}{=} H(m_1)$ or $\stackrel{?}{=} H(m_2)$

      $\vdots$
    - evaluate $H(m_i)$, check $H(m_i) \stackrel{?}{\in} \{H(m_1), \ldots, H(m_{i-1})\}$
  - prob. of collision after q values: $\approx q(q-1)/2 \cdot 1/[\text{nmb of hashes}]$

    (birthday paradox!)

    expected complexity: $\sqrt{2^n} = 2^{n/2}$

    not fine for $n = 80$!



Collision-resistance

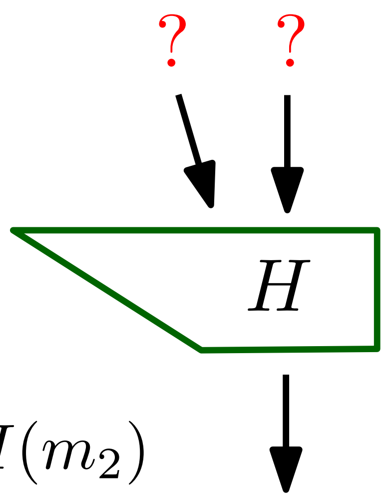
- **Collision attack:**
  - find m, m': $H(m) = H(m')$
  - brute-force: try arbitrary m's:
    * evaluate $H(m_1)$
    * evaluate $H(m_2)$, check $H(m_2) \stackrel{?}{=} H(m_1)$
    * evaluate $H(m_3)$, check $H(m_3) \stackrel{?}{=} H(m_1)$ or $\stackrel{?}{=} H(m_2)$

      $\vdots$
    * evaluate $H(m_i)$, check $H(m_i) \stackrel{?}{\in} \{H(m_1), \ldots, H(m_{i-1})\}$
  - prob. of collision after q values: $\approx q(q-1)/2 \cdot 1/[\text{nmb of hashes}]$

    (birthday paradox!)

    expected complexity: $\sqrt{2^n} = 2^{n/2}$

    not fine for $n = 80$!



Outputs of hash functions must be ≥ 160 bits

Constructing hash functions

- Assume we have a "**compression function**"

  $$f\colon \{0,1\}^{2n} \to \{0,1\}^n$$

- Can we build a hash function

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

Constructing hash functions

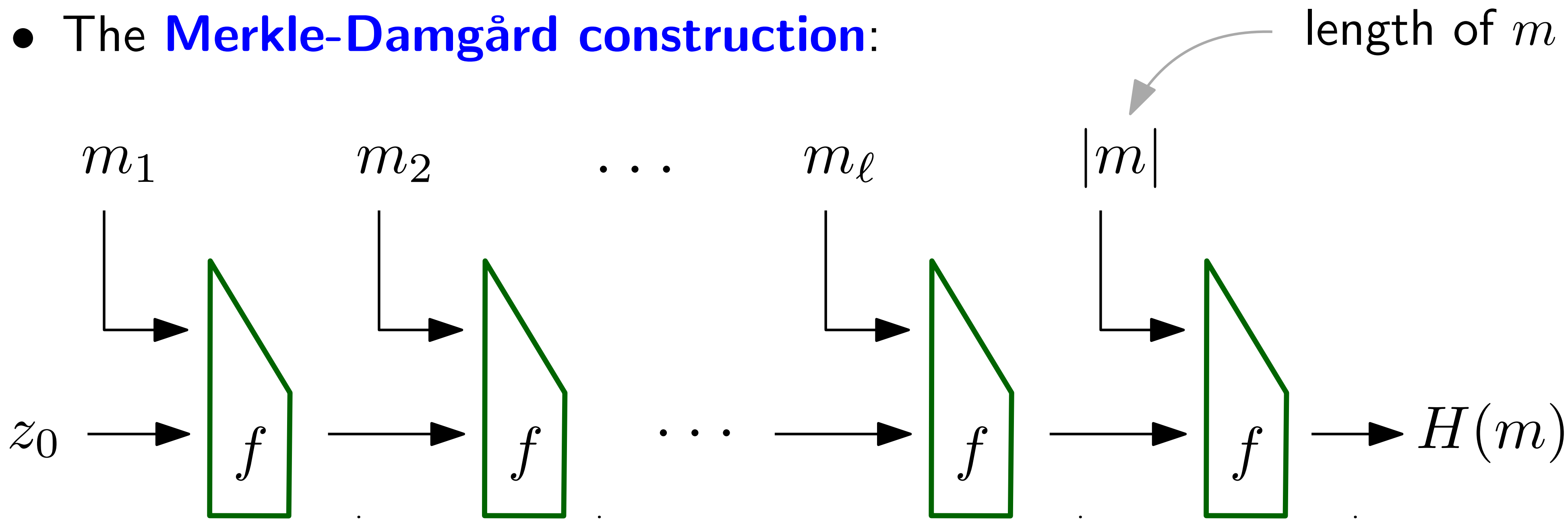
- Assume we have a "**compression function**"

  $$f\colon \{0,1\}^{2n} \to \{0,1\}^n$$

- Can we build a hash function

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

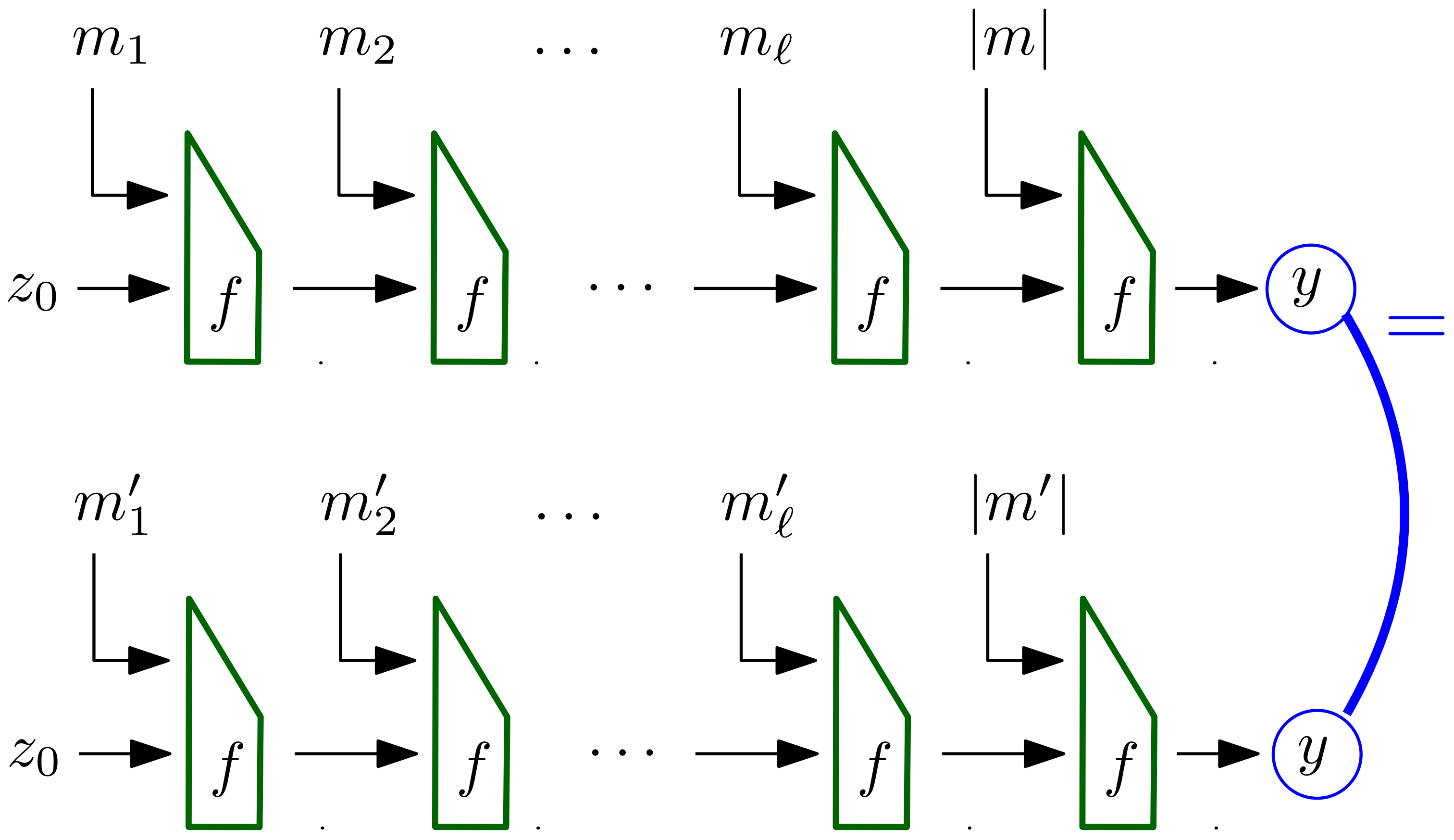
- The **Merkle-Damgård construction**:



Collision-resistance of Merkle–Damgård

Theorem: If f is collision-resistant, then so is H.

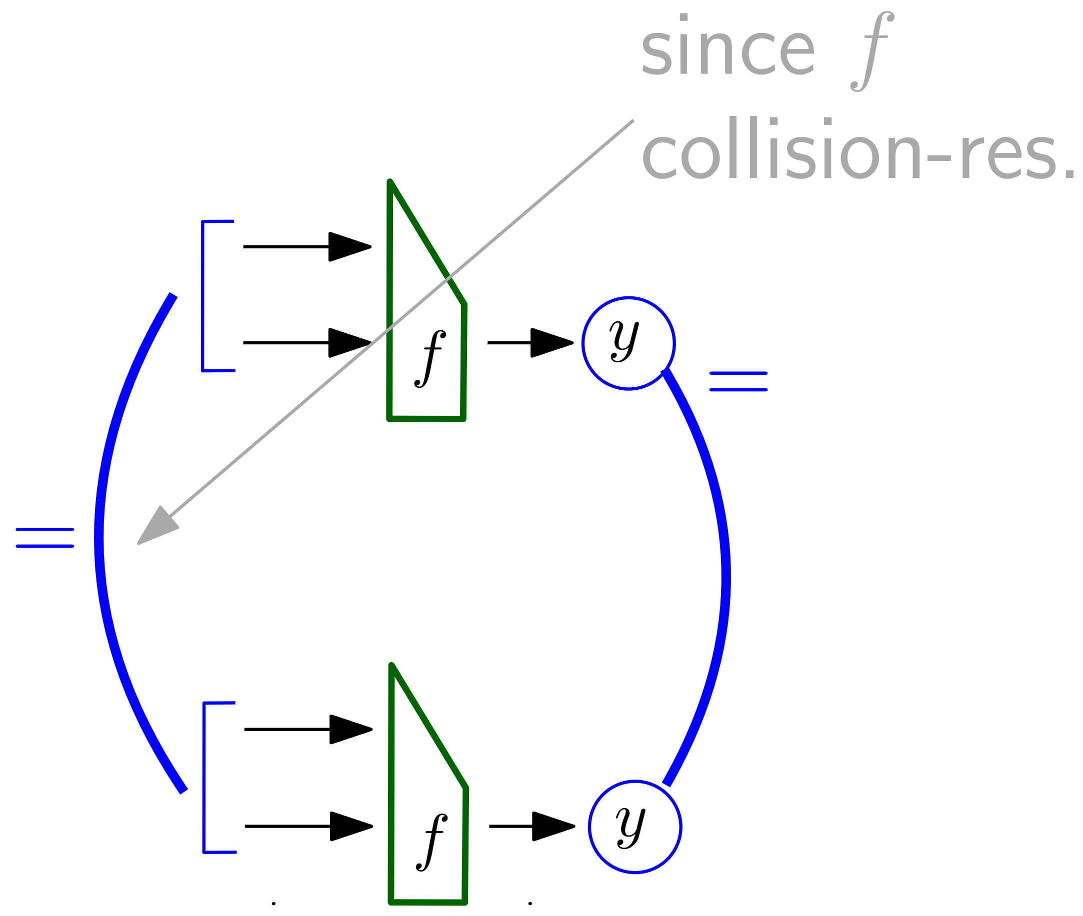
Proof:



Collision-resistance of Merkle–Damgård

Theorem: If f is collision-resistant, then so is H.

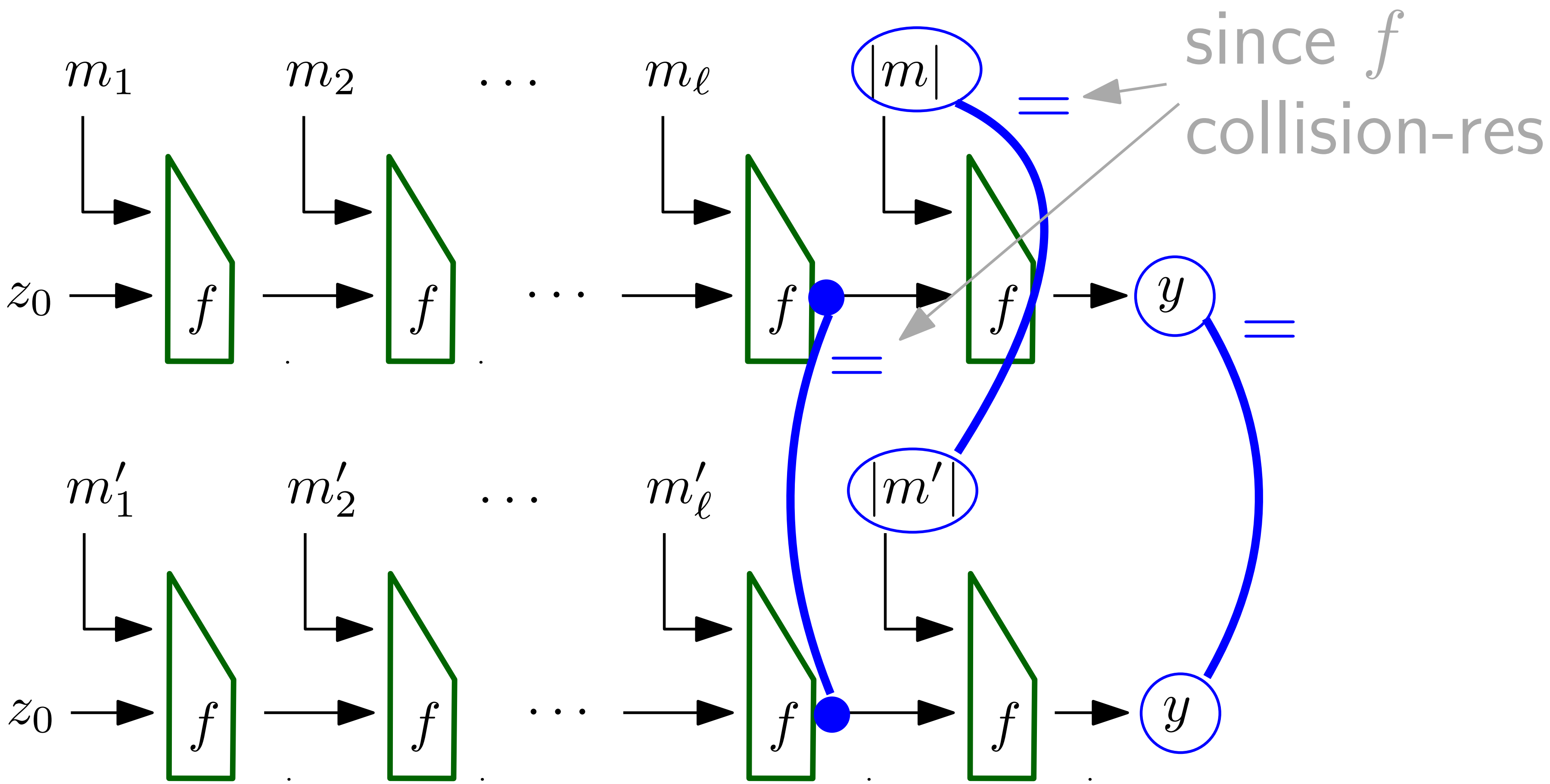
Proof:



Collision-resistance of Merkle–Damgård

Theorem: If f is collision-resistant, then so is H.

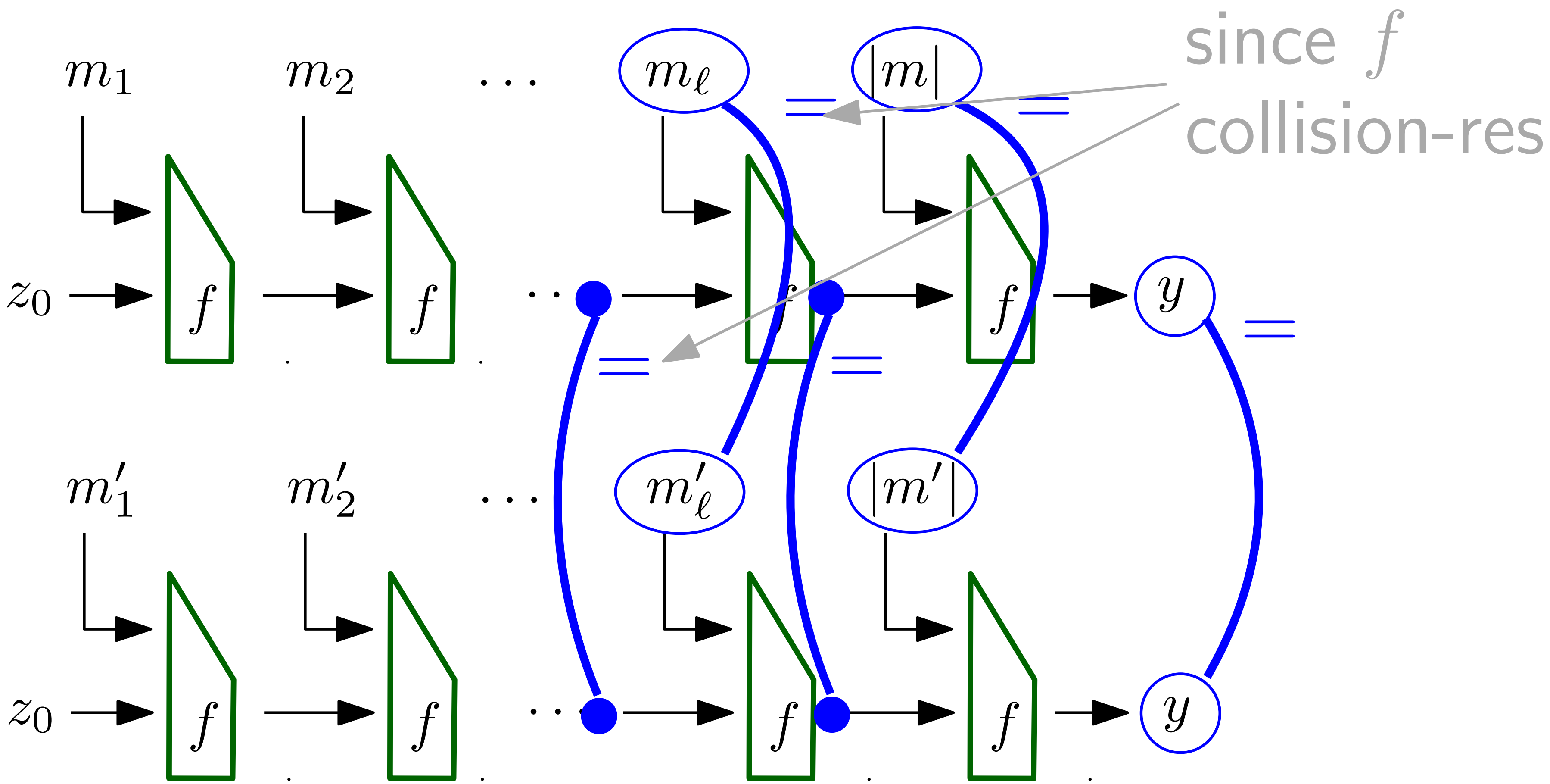
Proof:



Collision-resistance of Merkle–Damgård

Theorem: If f is collision-resistant, then so is H.

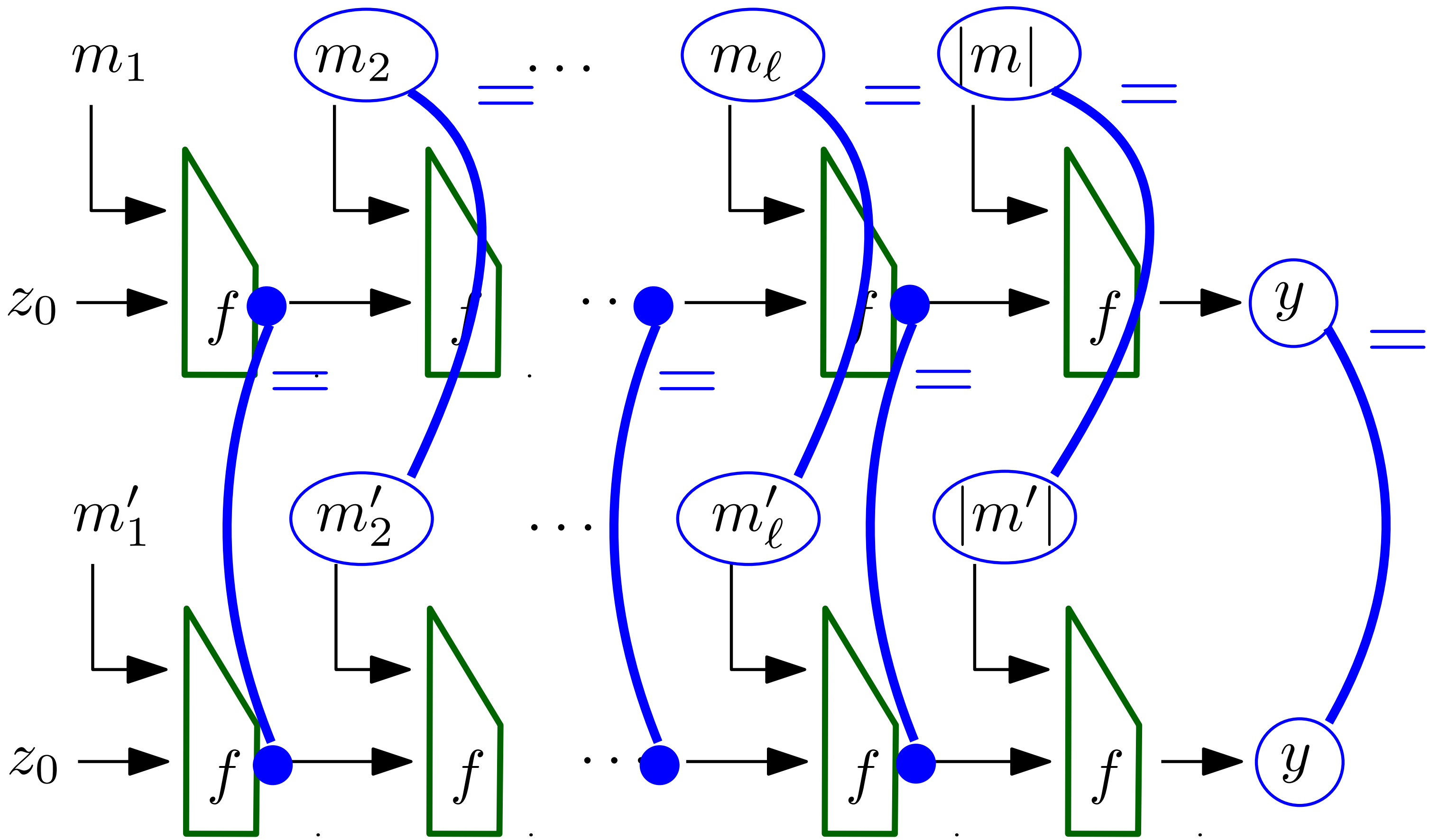
Proof:



Collision-resistance of Merkle–Damgård

Theorem: If f is collision-resistant, then so is H.

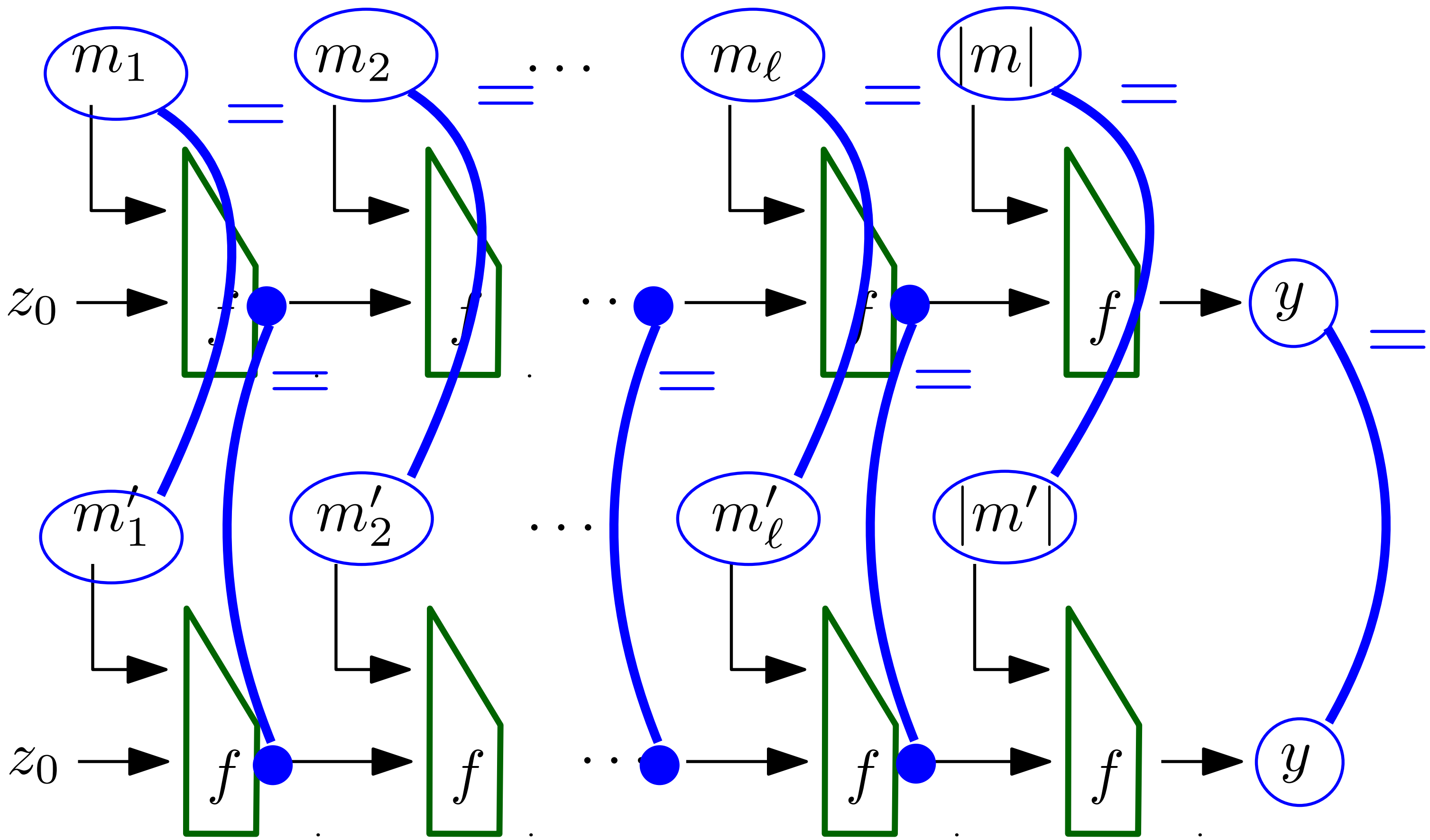
Proof:



Collision-resistance of Merkle–Damgård

Theorem: If f is collision-resistant, then so is H.

Proof:

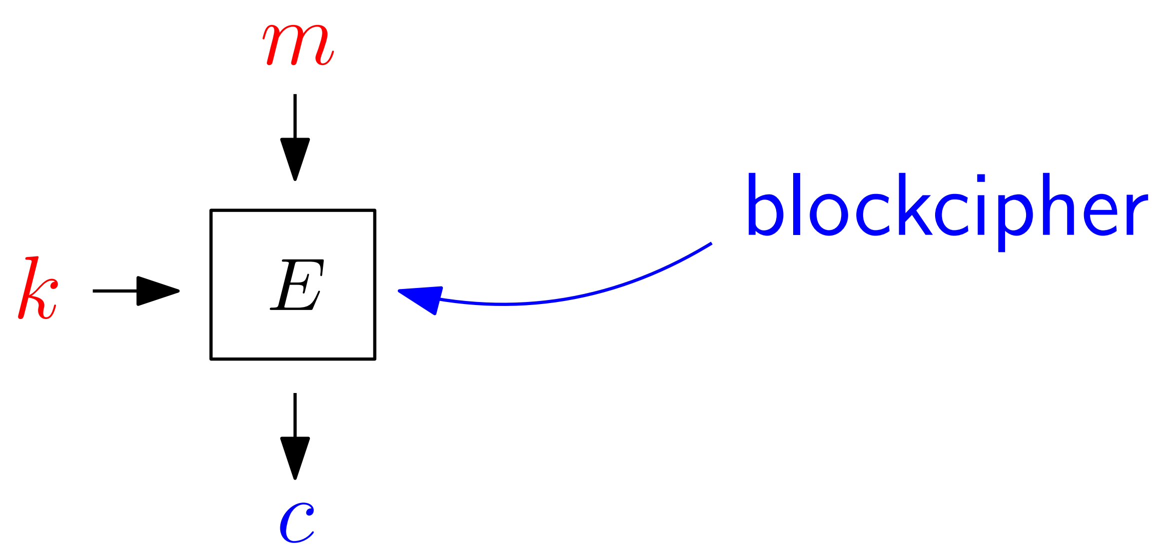


$\Rightarrow$ $m = m'$ and thus not a collision!

Construction of compression function

Davies-Meyer Construction

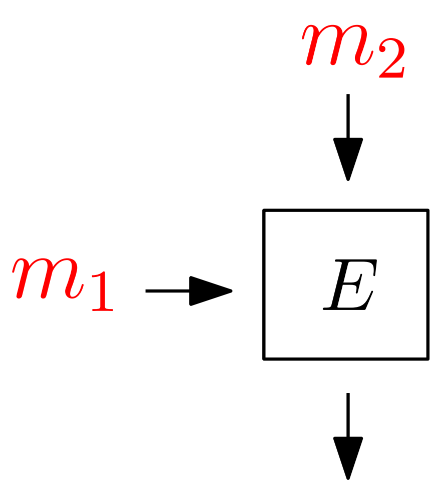
- compression function $f\colon \{0,1\}^{2n} \to \{0,1\}^n$ from a blockcipher



Construction of compression function

Davies-Meyer Construction

- compression function $f\colon \{0,1\}^{2n} \to \{0,1\}^n$ from a blockcipher
- first try:



"key"

$$f(m_1, m_2) = E(m_1, m_2)$$

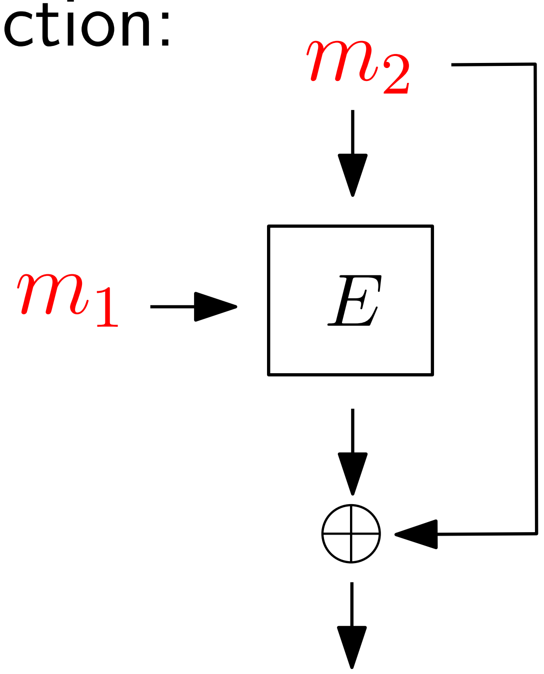
Not secure

$(m_1, D(m_1, 0))$ and $(m_1', D(m_1', 0))$ are a collision!
(since $f(m_1, D(m_1, 0)) = 0$)

Construction of compression function

Davies-Meyer Construction

- compression function $f\colon \{0,1\}^{2n} \to \{0,1\}^n$ from a blockcipher
- secure construction:



$$f(m_1, m_2) = E(m_1, m_2) \oplus m_2$$

Hash functions in practice

History

- "MD4 family"
  - MD4

Hash functions in practice

History

- "MD4 family"
  - MD4 broken in the 1980's
  - MD5 (very popular!)

Hash functions in practice

History

- "MD4 family"
  - MD4 broken in the 1980's
  - MD5 (very popular!) broken in 1991
  - SHA-1 (standard 1995, widely used)

Hash functions in practice

History

- "MD4 family"
  - MD4 broken in the 1980's
  - MD5 (very popular!) broken in 1991
  - SHA-1 (standard 1995, widely used)
    - theoretical attack 2004
    - collision in 2017 (taking 6 500 CPU years)

Hash functions in practice

History

- "MD4 family"
  - MD4 broken in the 1980's
  - MD5 (very popular!) broken in 1991
  - SHA-1 (standard 1995, widely used)
    - theoretical attack 2004
    - collision in 2017
  - SHA-2 (standard 2001, used in Bitcoin)

Hash functions in practice

History

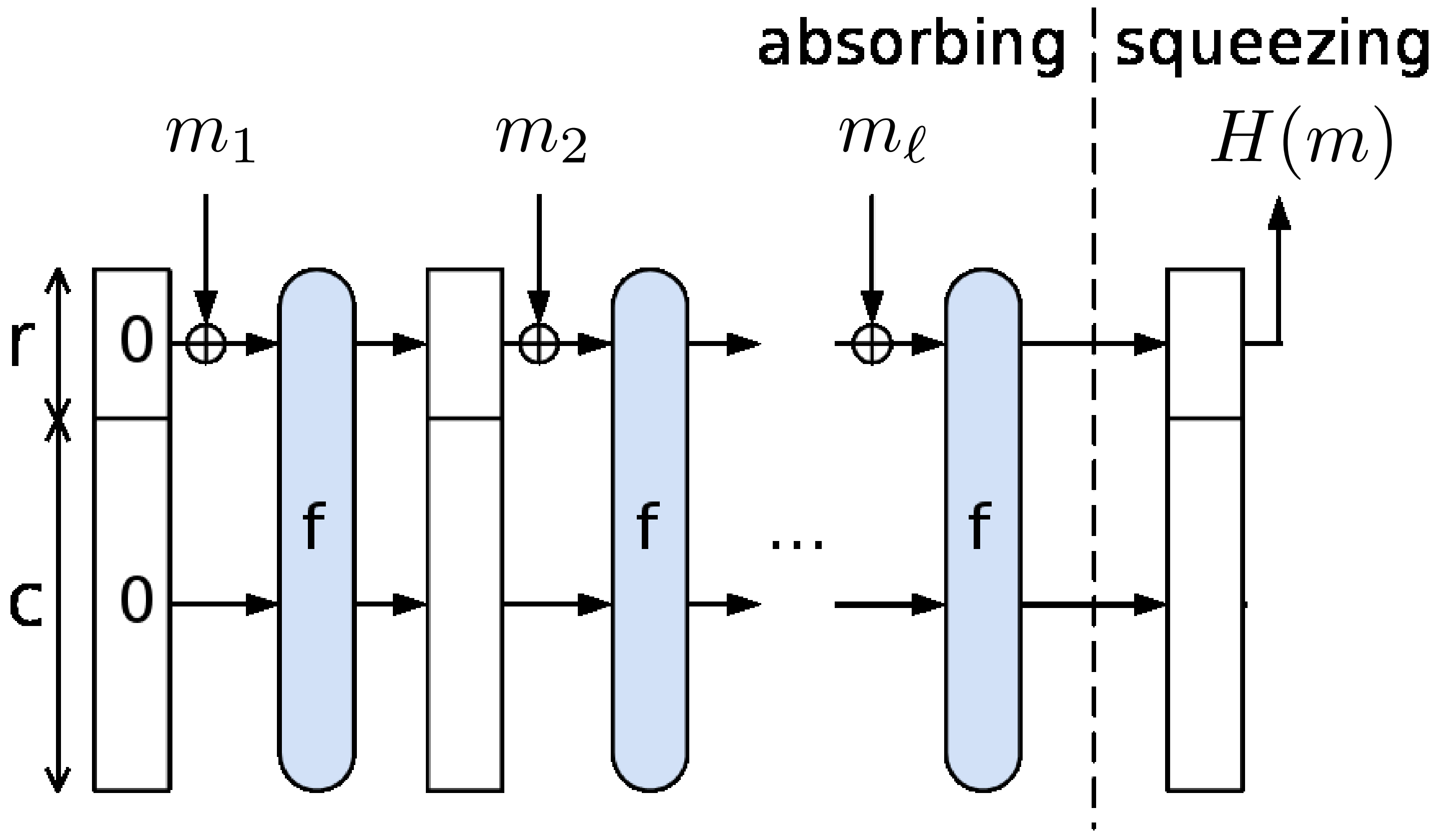
- "MD4 family"
  - MD4 broken in the 1980's
  - MD5 (very popular!) broken in 1991
  - SHA-1 (standard 1995, widely used)
    - theoretical attack 2004
    - collision in 2017
  - SHA-2 (standard 2001, used in Bitcoin)
- SHA-3 competition 2007

  Requirements: output lengths: 224/256/384/512

  2012 winner announced: Keccak

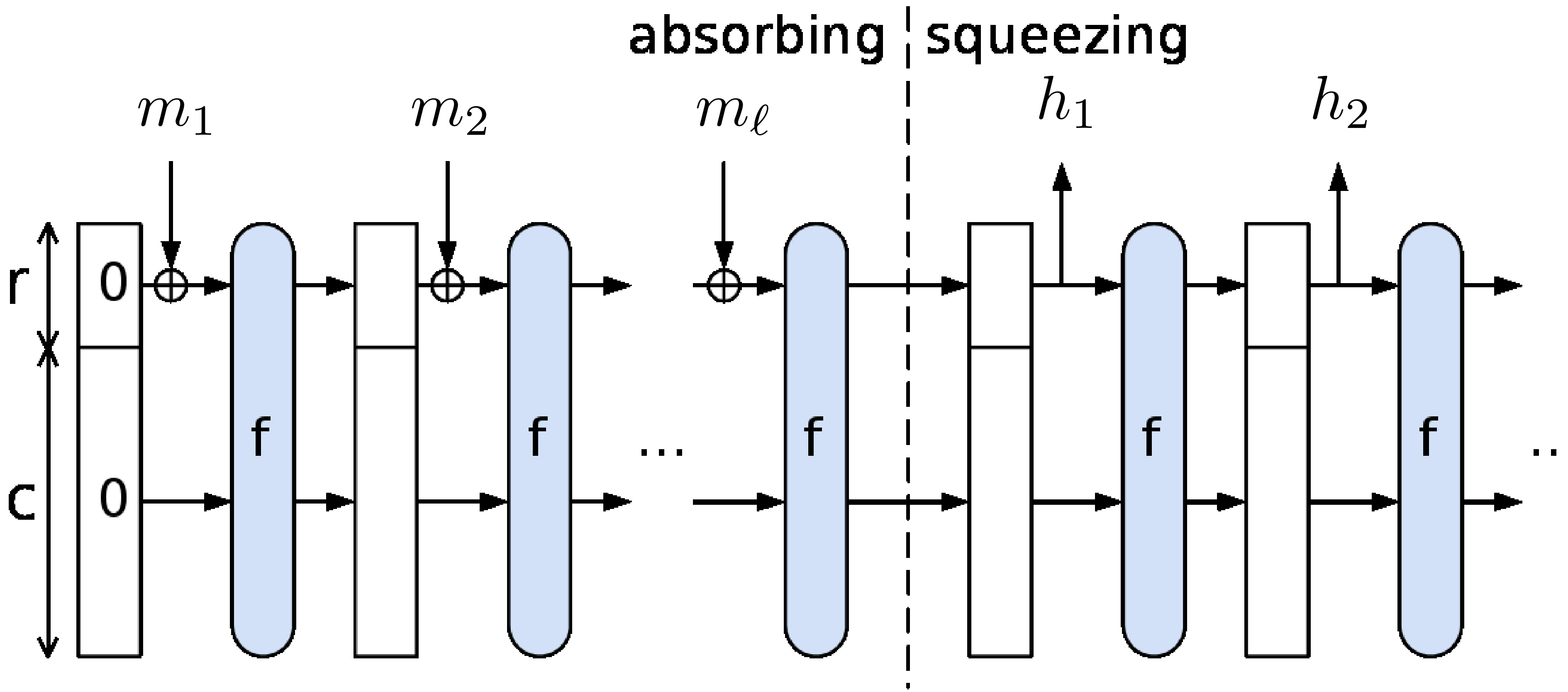
Keccak

"sponge construction"



Keccak

"sponge construction"

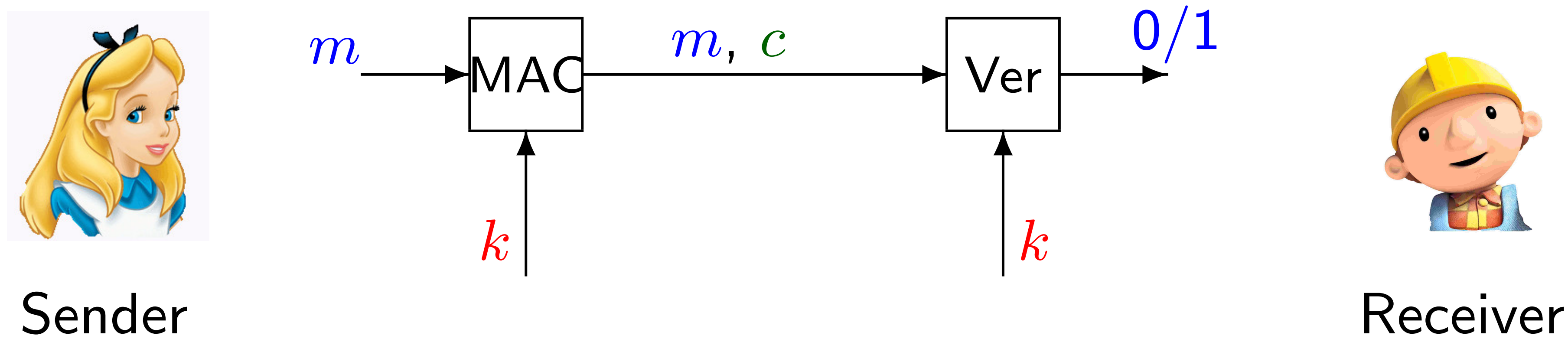


Message authentication codes

MACs

Message authentication codes

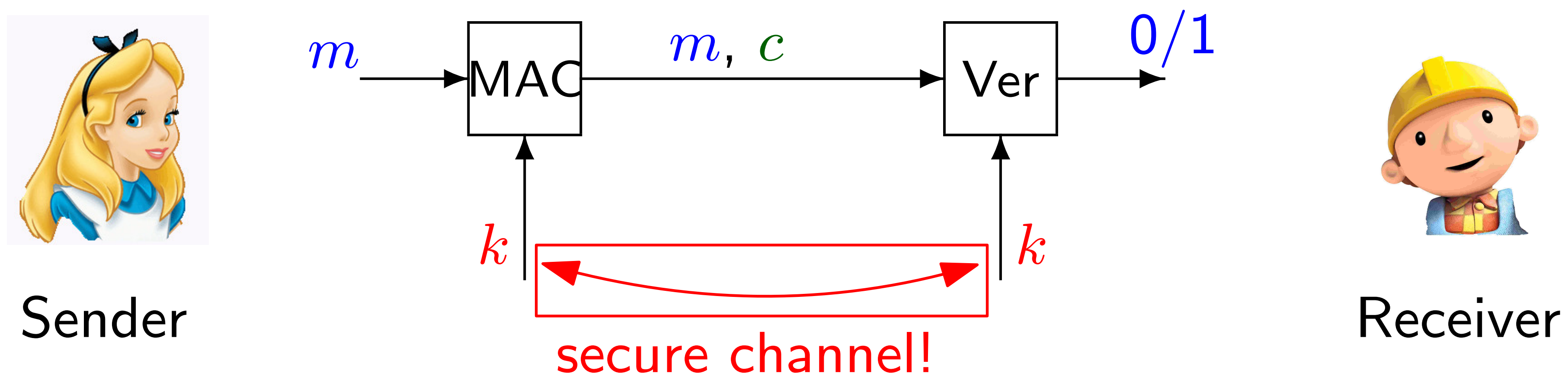
- "symmetric" version of signatures



MACs

Message authentication codes

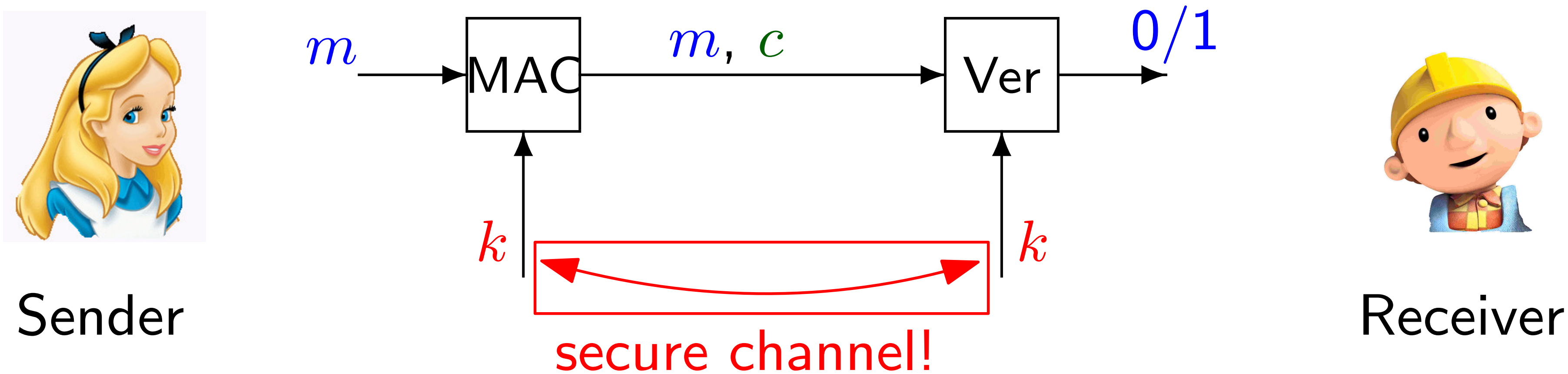
- "symmetric" version of signatures



MACs

Message authentication codes

- "symmetric" version of signatures



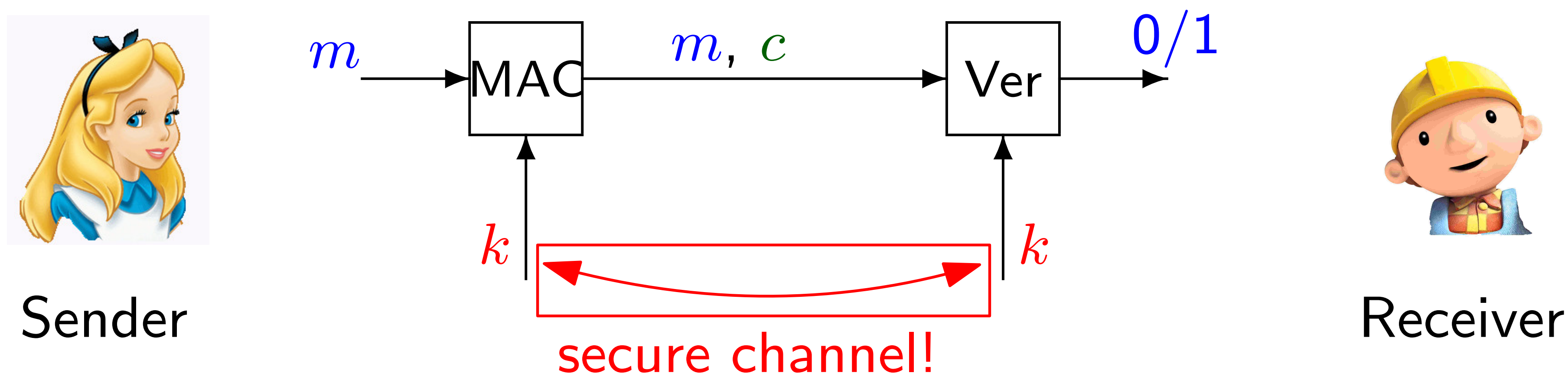
Why? (if we have signatures?)

- much more efficient!
- main application: **authenticated encryption**

MACs

Message authentication codes

- "symmetric" version of signatures



Verification of (m, c):

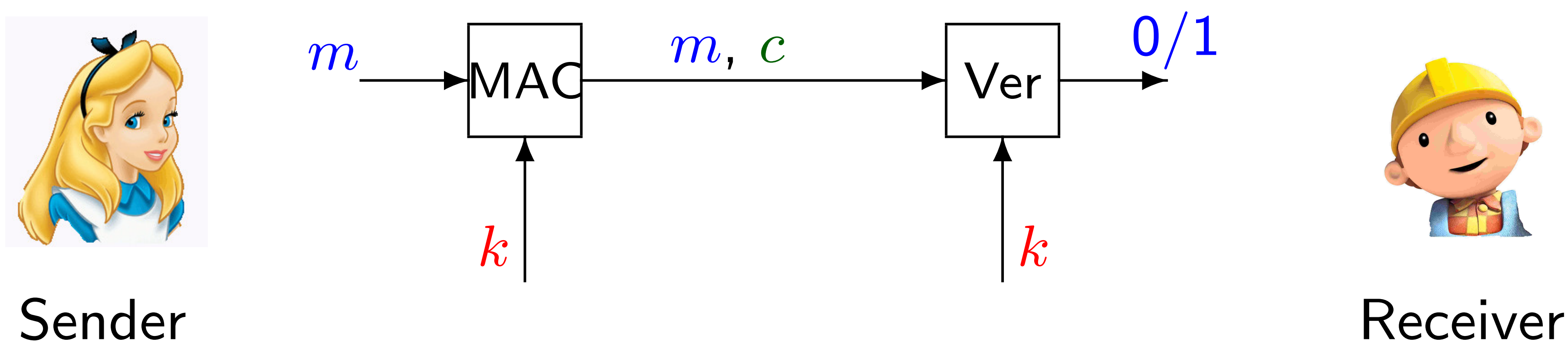
- compute: $c' = \mathsf{Enc}_k(m)$
- check $c' \stackrel{?}{=} c$

MACs

Properties of MACs

- arbitrary message length
- fixed length of MAC
- provide authentication/integrity

X non-repudiation **not** provided



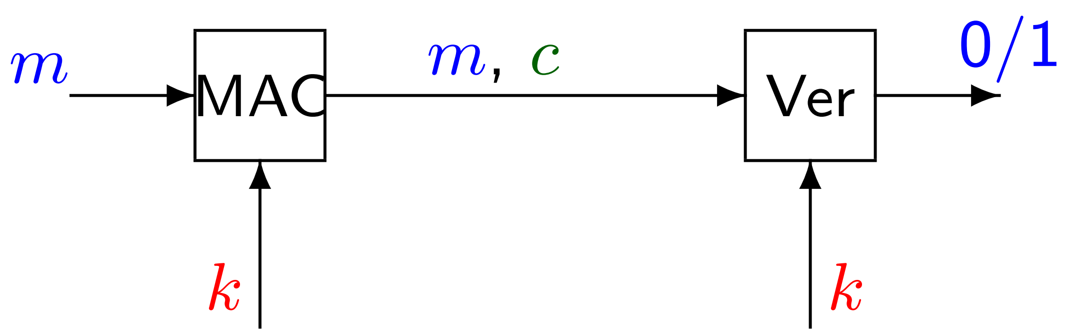
MACs

MACs from hash functions

- 1st idea: $\mathsf{MAC}_k(m) := H(k, m)$

  Problem: **Message-extension attack**

  - constructions of H à la Merkle–Damgård
  - from $H(k, m)$ anyone can compute $H(k, m\|m') = H(H(k, m), m')$



MACs

MACs from hash functions

- 1st idea: $\text{MAC}_k(m) := H(k, m)$

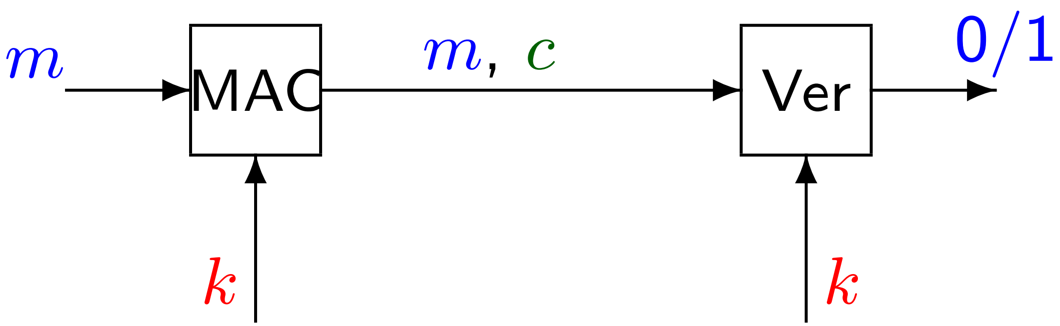
  Problem: **Message-extension attack**

  - constructions of H à la Merkle–Damgård
  - from $H(k, m)$ anyone can compute $H(k, m \| m') = H(H(k, m), m')$

- 2nd idea: $\text{MAC}_k(m) := H(m, k)$

  Problem: **Collision attack**

  - If $H(m) = H(m')$ then $\text{MAC}_k(m) = \text{MAC}_k(m')$
  - collisions "easier" to find (Birthday bound!)



MACs

idea that works: **HMAC**:

- proposed in 1996
- used in SSL/TLS
- idea: $\text{MAC}_{(k_1,k_2)}(m)$

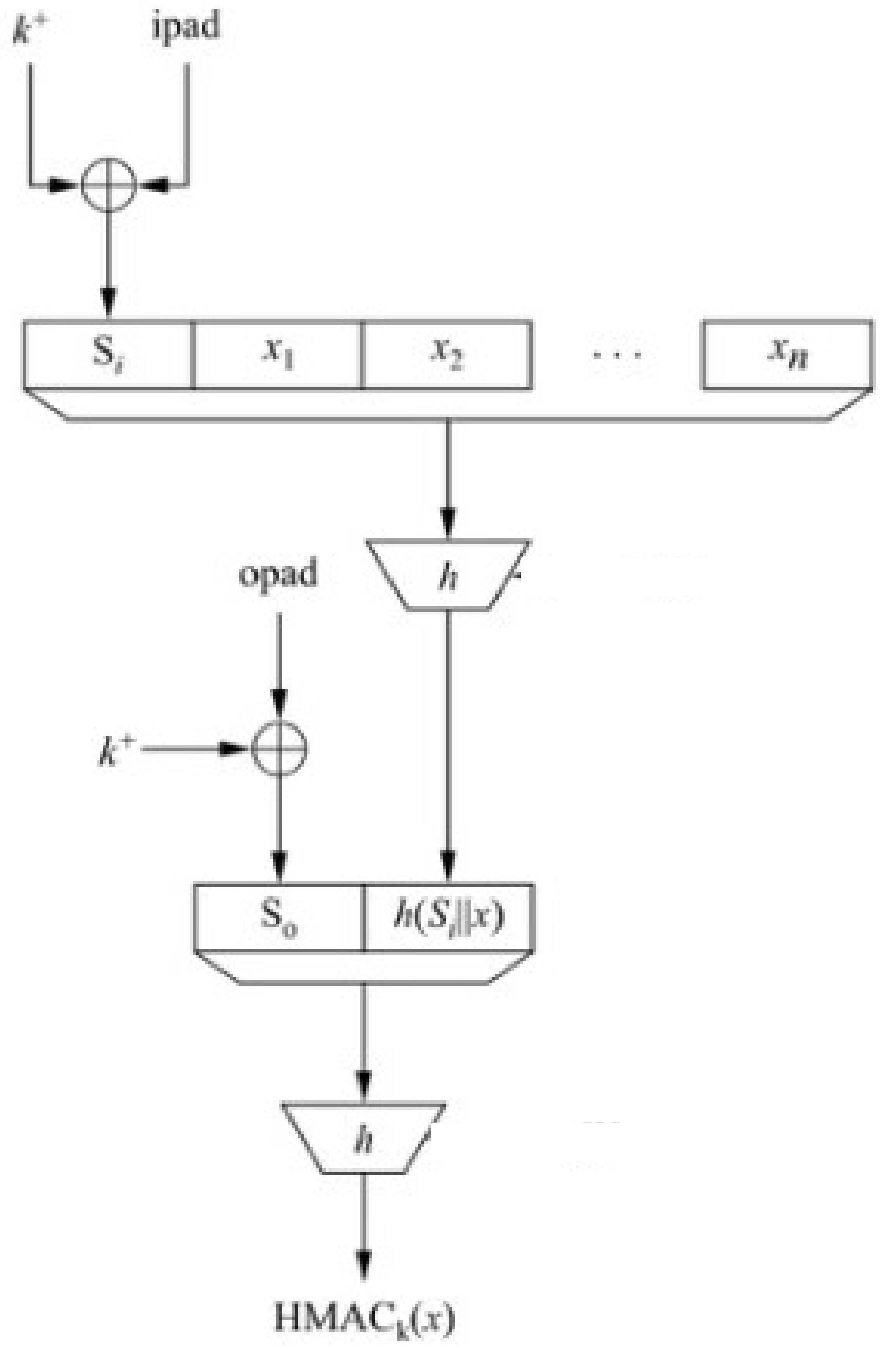
$$:= H(k_2, H(k_1, m))$$

MACs

idea that works: **HMAC**:

- proposed in 1996
- used in SSL/TLS
- idea: $\text{MAC}_{(k_1,k_2)}(m)$

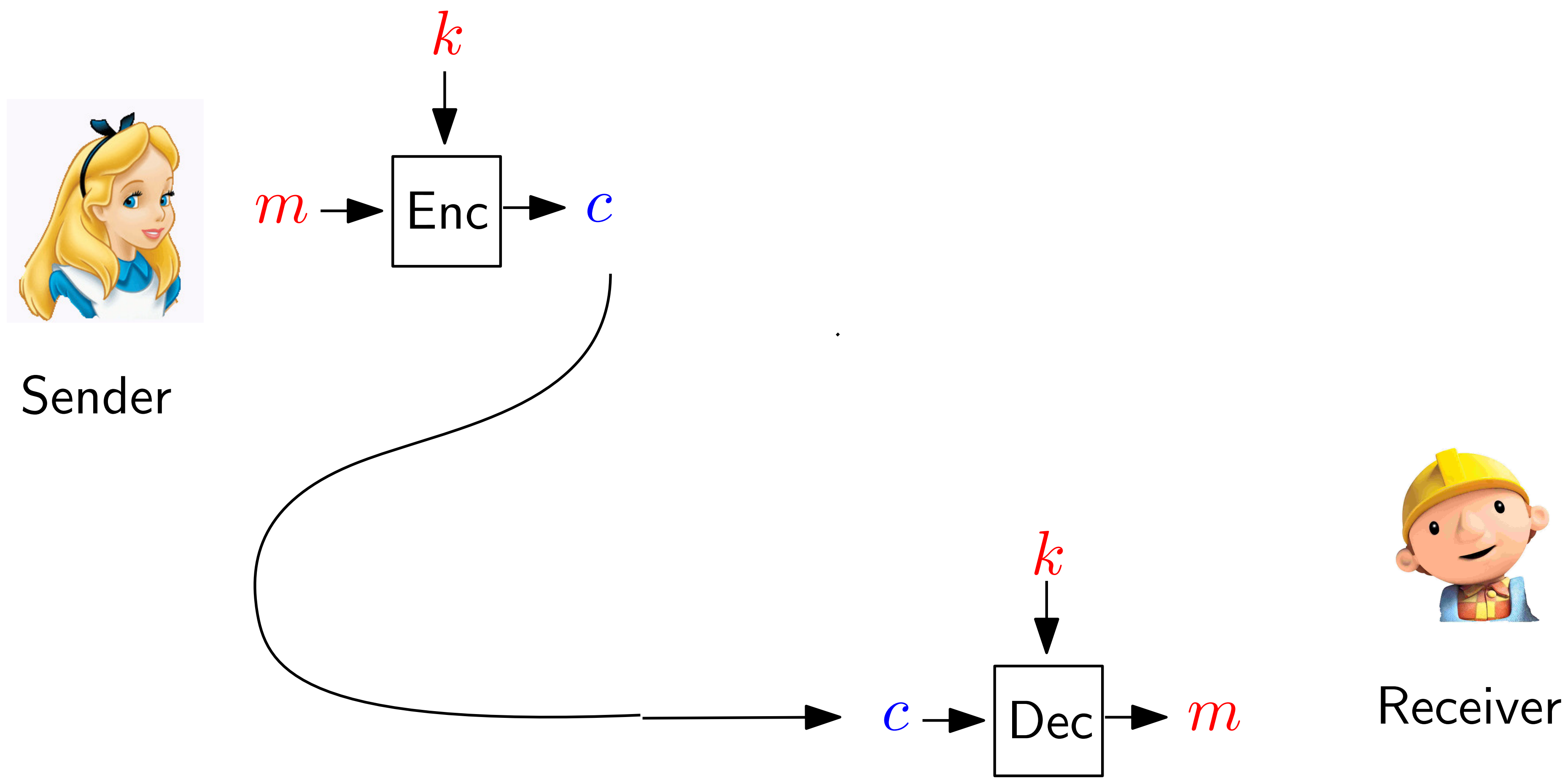
$$:= H(k_2, H(k_1, m))$$



MACs

Symmetric encryption

- confidentiality



MACs

Authenticated encryption

- confidentiality *and* authenticity