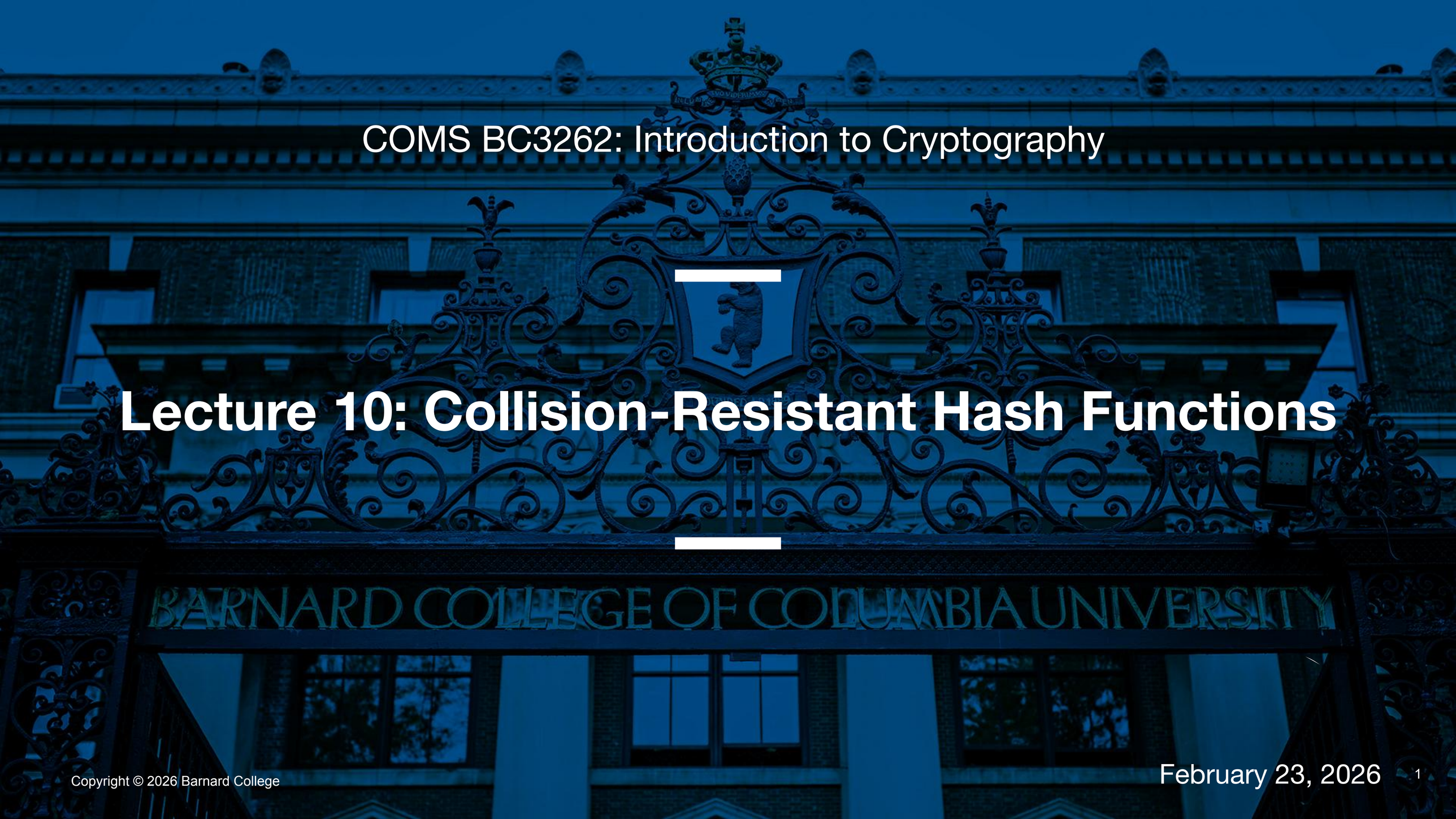



COMS BC3262: Introduction to Cryptography

# Lecture 10: Collision-Resistant Hash Functions



February 23, 2026

# Logistics

- Today is another snow day / remote teaching day
  - My office hours will be remote via appointment today
  - TA will not be having his regular office hours on Tuesday this week

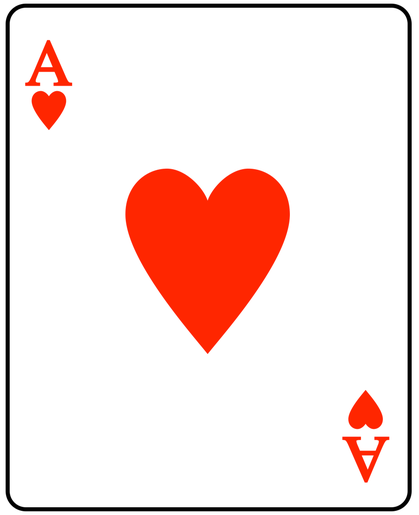
# Implies Logic

- Recall from discrete math: “A implies B” (“if A then B”)
  - **Contrapositive** is “not B implies not A”
- For example, consider event A as “drawing the Ace of Hearts” and event B as “drawing a red card”.
  - If I drew the Ace of Hearts, then I drew a red card
  - If I **didn’t** draw a red card, then I **didn’t** draw an ace of hearts
- This is the logic behind reductions in this class

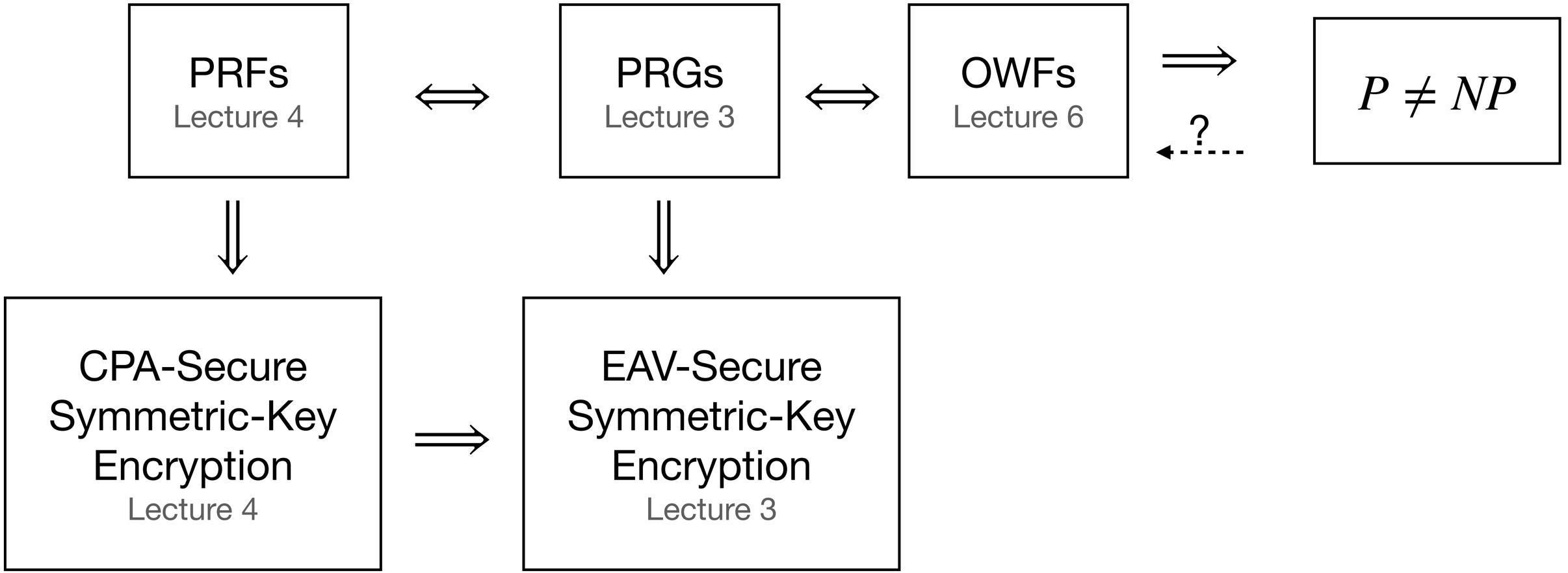
“cryptographic assumption holds” implies “construction is secure”

is equivalent to

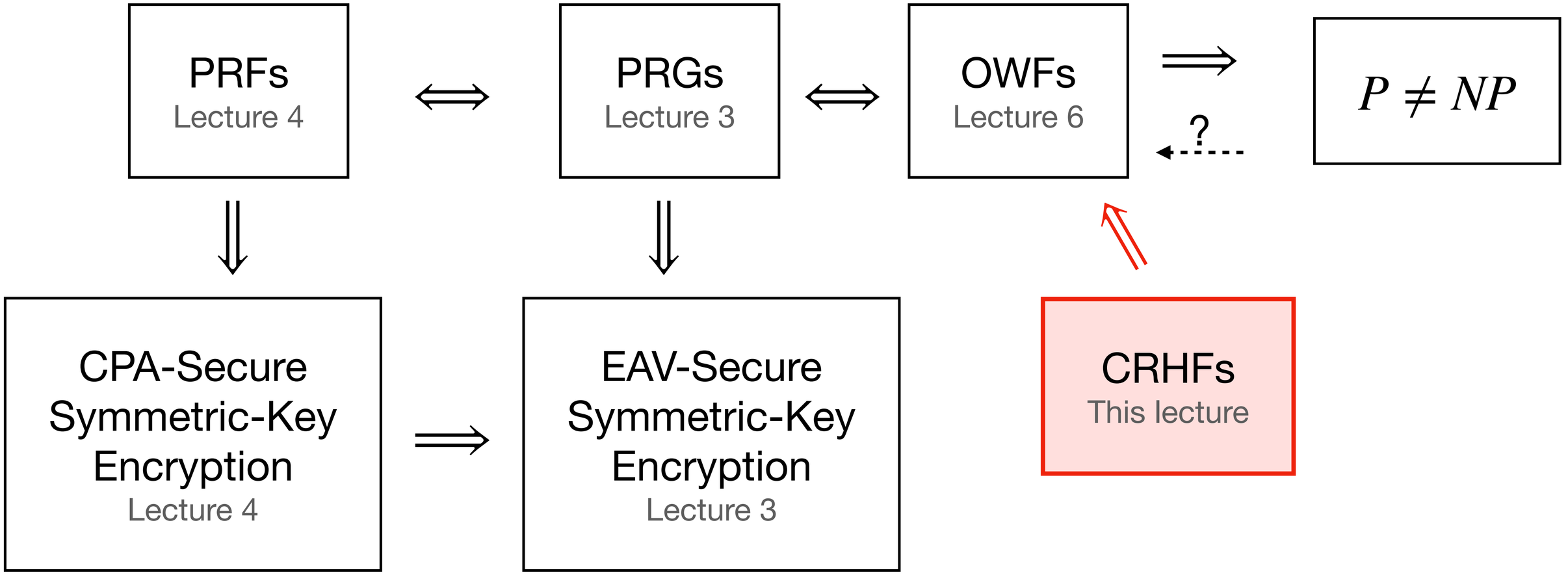
“construction is **not** secure” implies “cryptographic assumption does **not** hold”



# The World of Crypto Primitives We've Seen So Far



# The World of Crypto Primitives We've Seen So Far



# Today’s Lecture

- Collision-Resistant Hash Functions
- Generic Attacks
- Domain Extension
- Hash and Authenticate

# Collision-Resistant Hash Functions

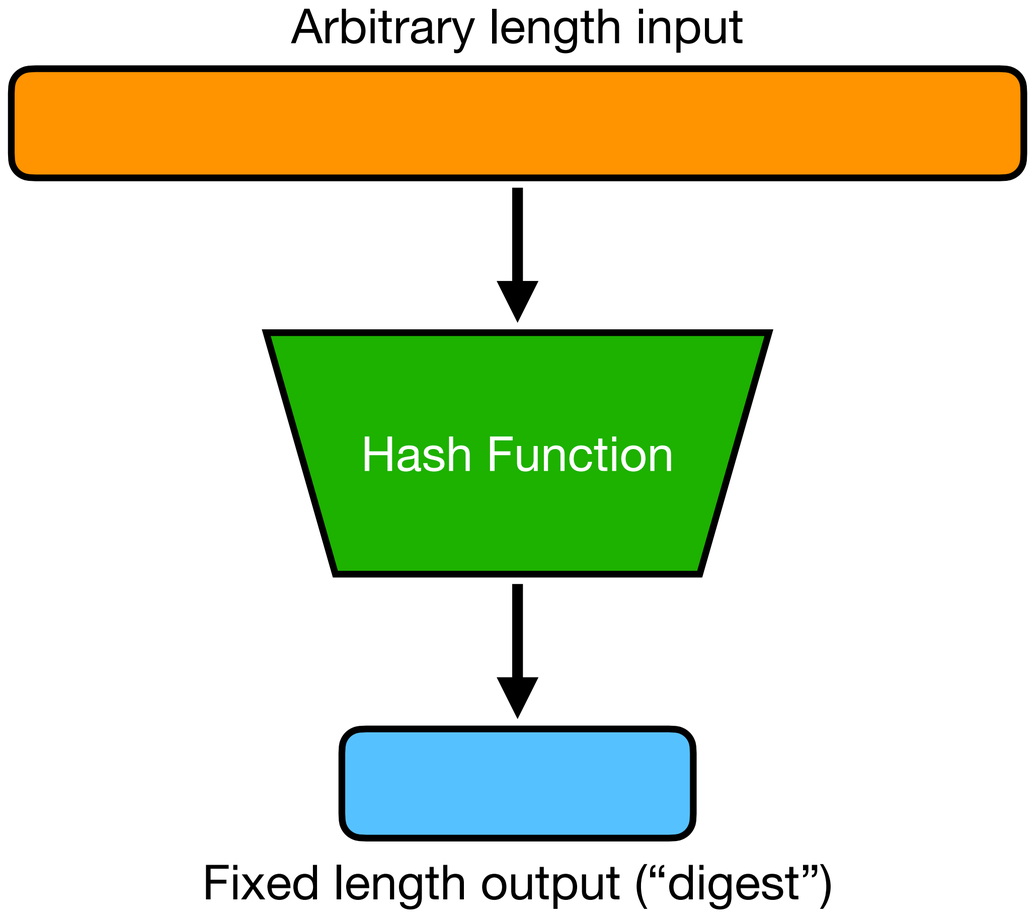
# Hash Functions

A hash function maps strings of arbitrary length to a fixed-length output

$H : \{0,1\}^* \rightarrow \{0,1\}^n$ for some $n \in \mathbb{N}$

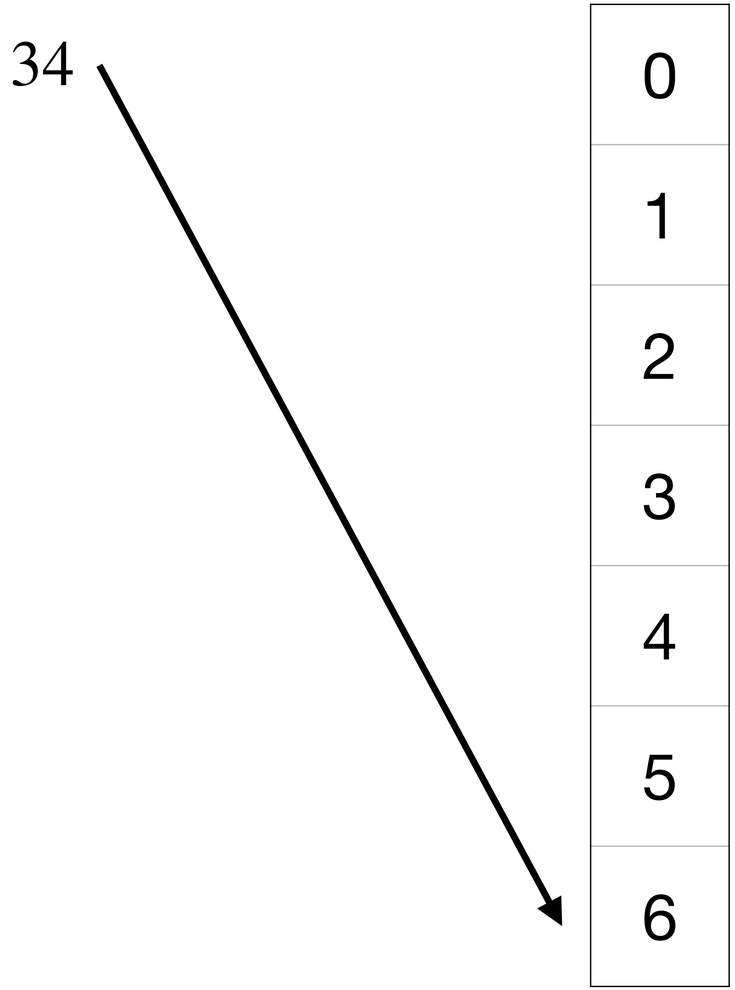
Desired properties:

- Compressing
- Given $x \in \{0,1\}^*$ and $y \in \{0,1\}^n$ can verify whether $y = H(x)$
- Output is distributed “randomly” (minimizes collisions)



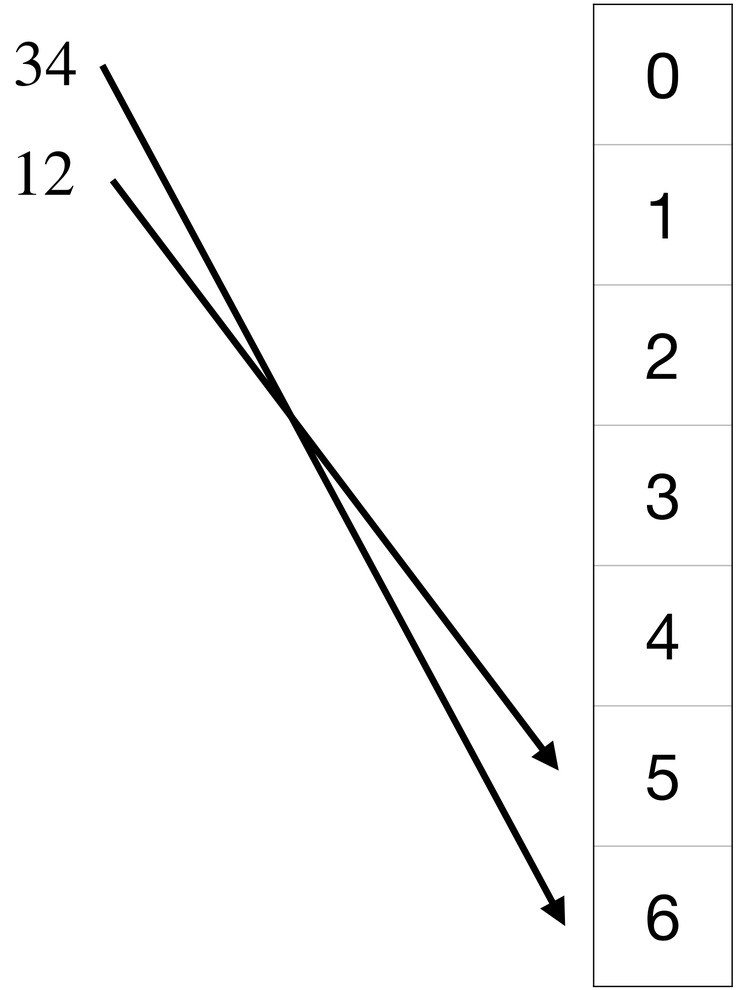
# Non-Cryptographic Hashes

- Heavily used in data structures
- Main emphasis is on reducing collisions and enabling efficient look-up
- Example: $x \mapsto x \mod 7$



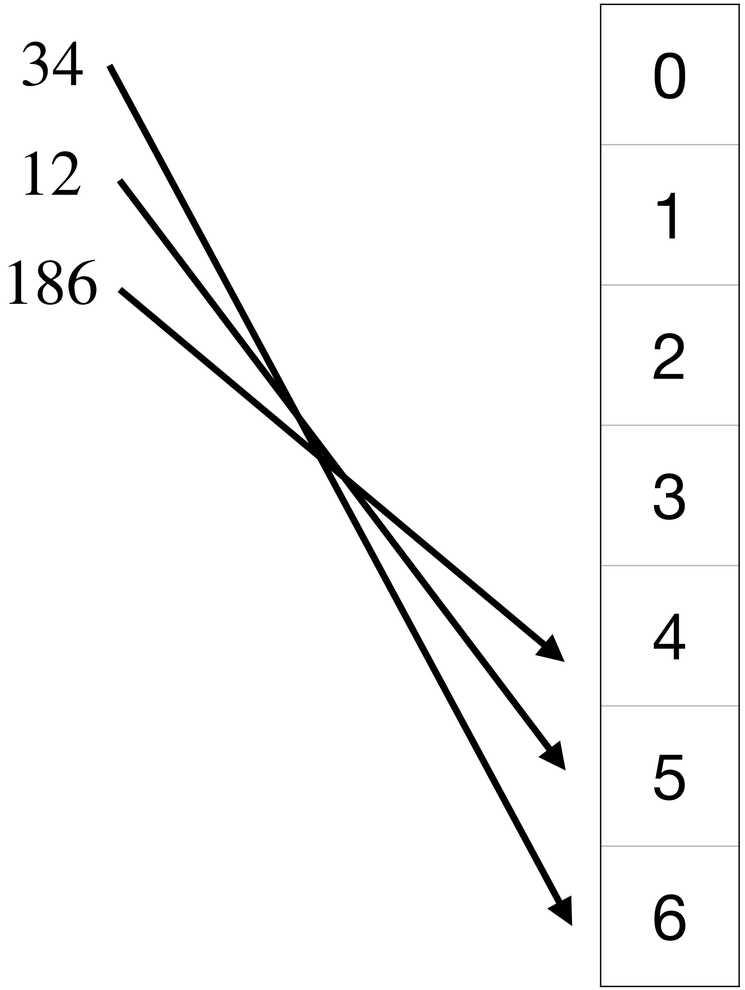
# Non-Cryptographic Hashes

- Heavily used in data structures
- Main emphasis is on reducing collisions and enabling efficient look-up
- Example: $x \mapsto x \mod 7$



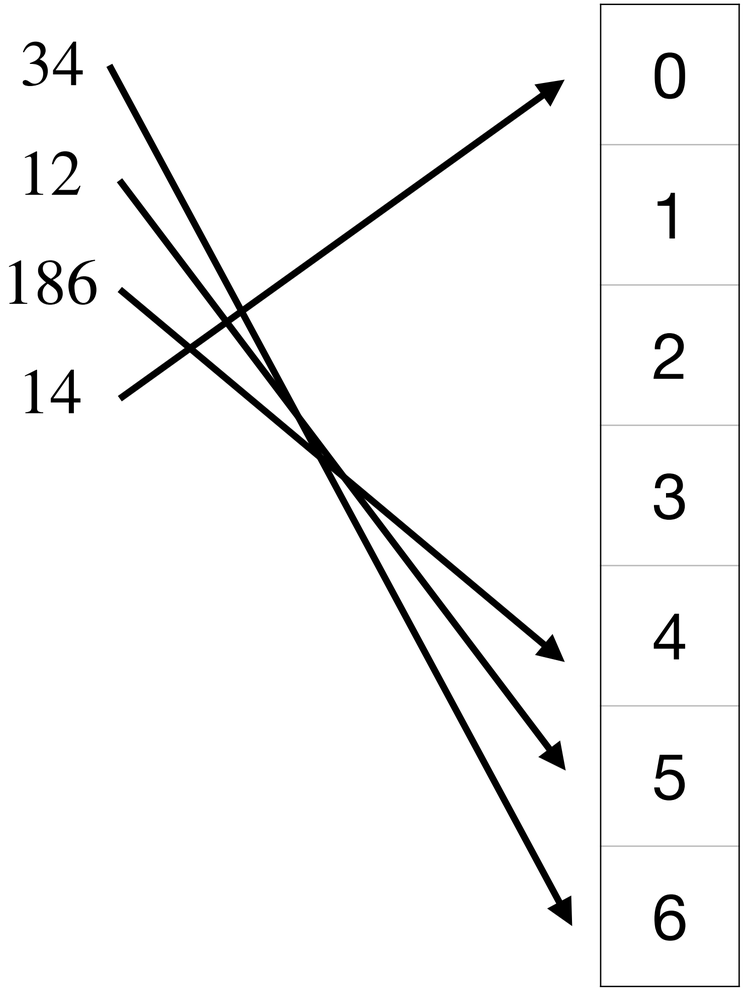
# Non-Cryptographic Hashes

- Heavily used in data structures
- Main emphasis is on reducing collisions and enabling efficient look-up
- Example: $x \mapsto x \mod 7$



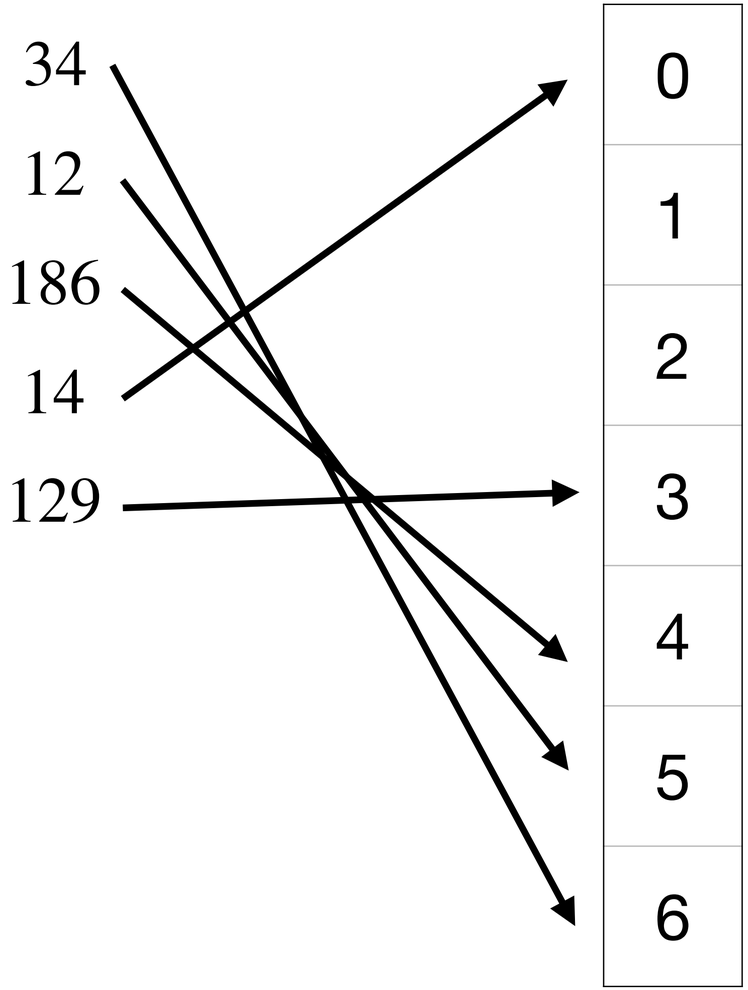
# Non-Cryptographic Hashes

- Heavily used in data structures
- Main emphasis is on reducing collisions and enabling efficient look-up
- Example: $x \mapsto x \mod 7$



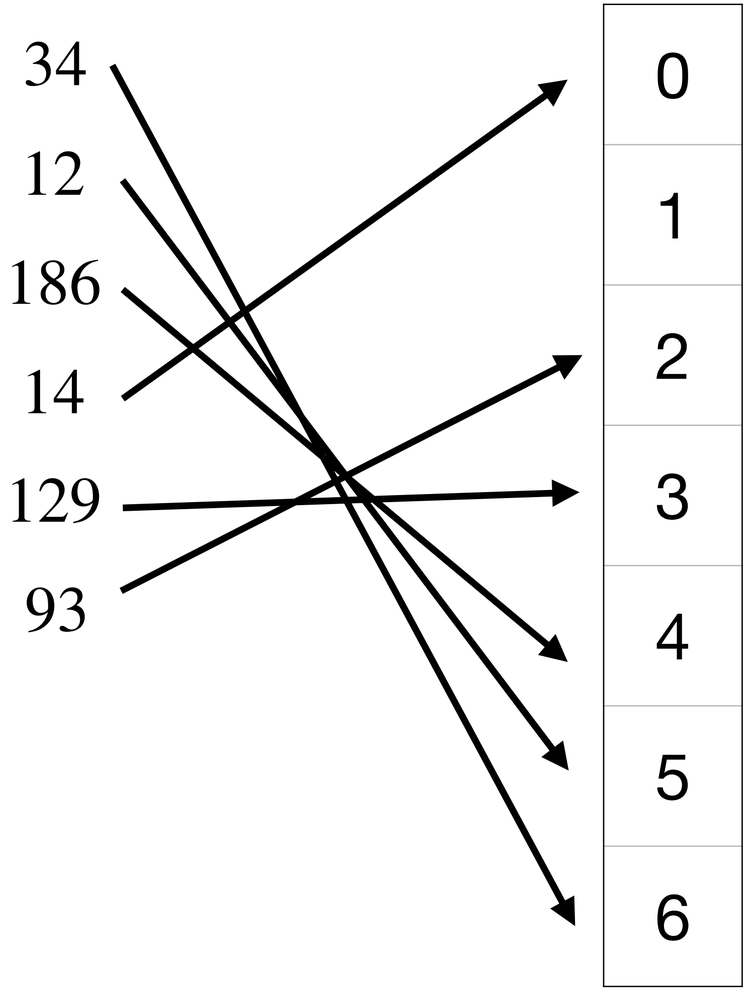
# Non-Cryptographic Hashes

- Heavily used in data structures
- Main emphasis is on reducing collisions and enabling efficient look-up
- Example: $x \mapsto x \mod 7$



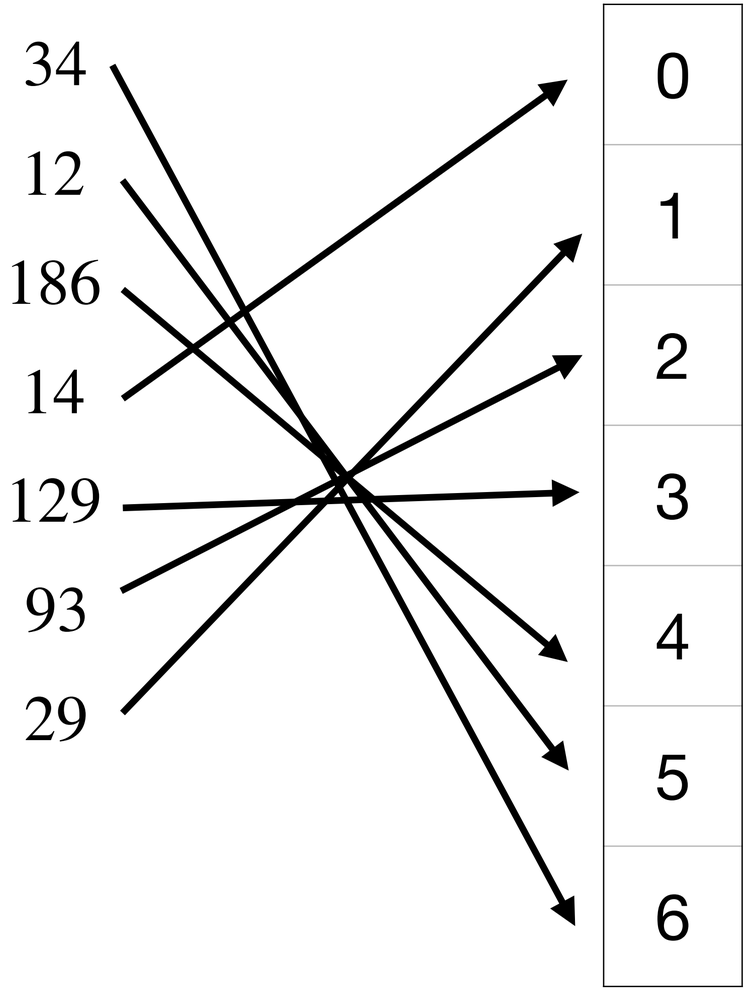
# Non-Cryptographic Hashes

- Heavily used in data structures
- Main emphasis is on reducing collisions and enabling efficient look-up
- Example: $x \mapsto x \mod 7$



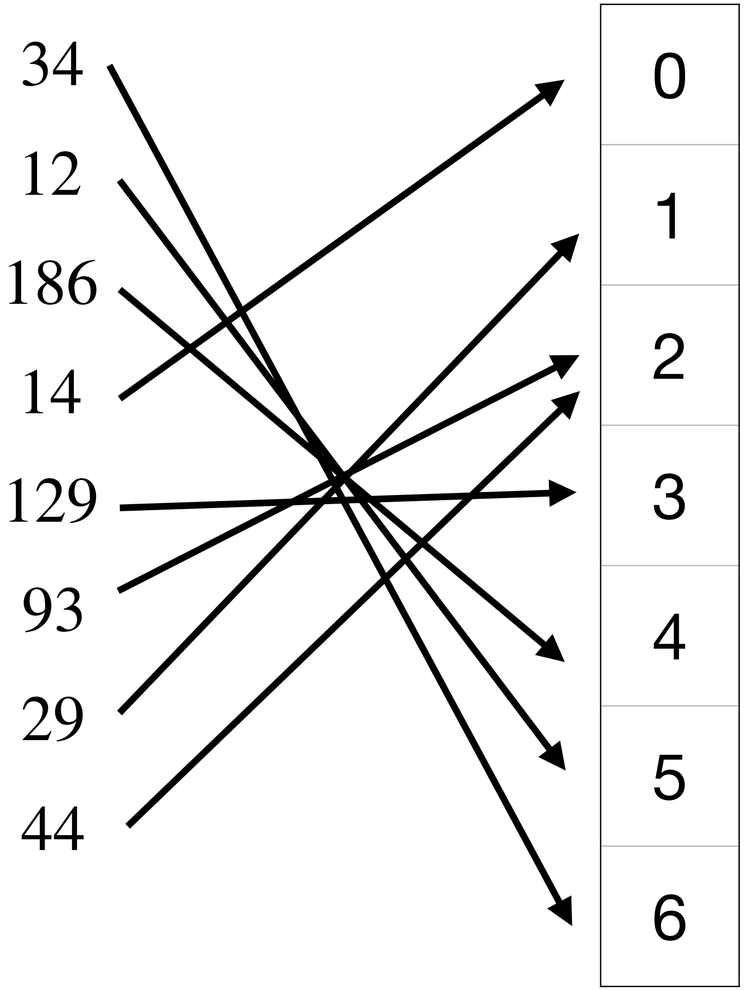
# Non-Cryptographic Hashes

- Heavily used in data structures
- Main emphasis is on reducing collisions and enabling efficient look-up
- Example: $x \mapsto x \mod 7$



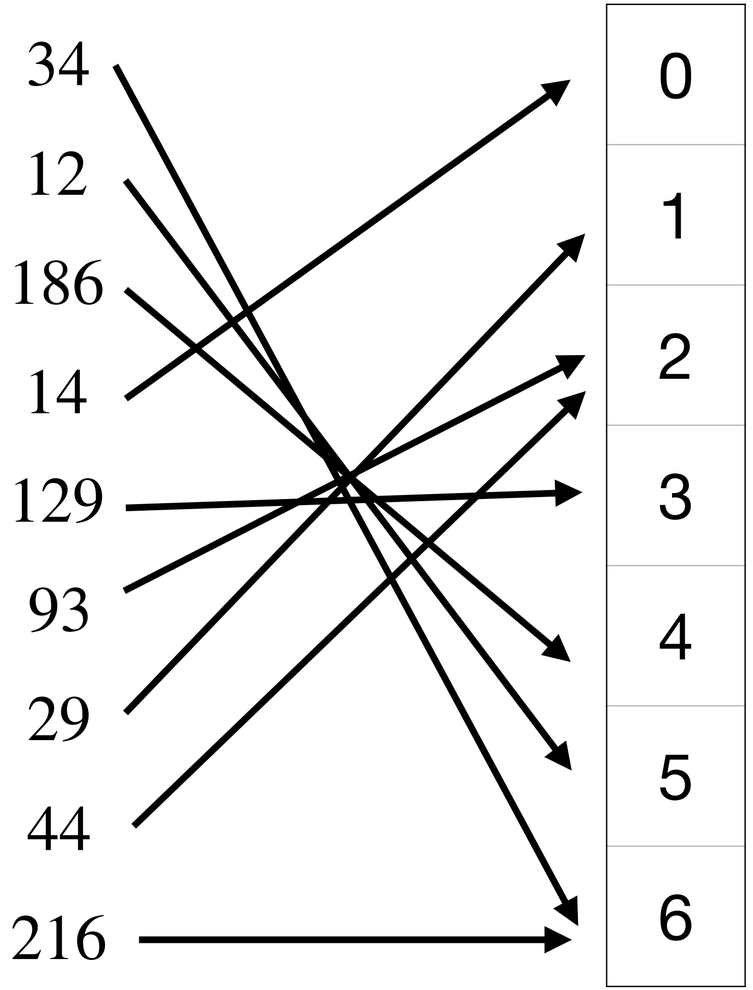
# Non-Cryptographic Hashes

- Heavily used in data structures
- Main emphasis is on reducing collisions and enabling efficient look-up
- Example: $x \mapsto x \mod 7$



# Non-Cryptographic Hashes

- Heavily used in data structures
- Main emphasis is on reducing collisions and enabling efficient look-up
- Example: $x \mapsto x \mod 7$



# Non-Cryptographic Hashes

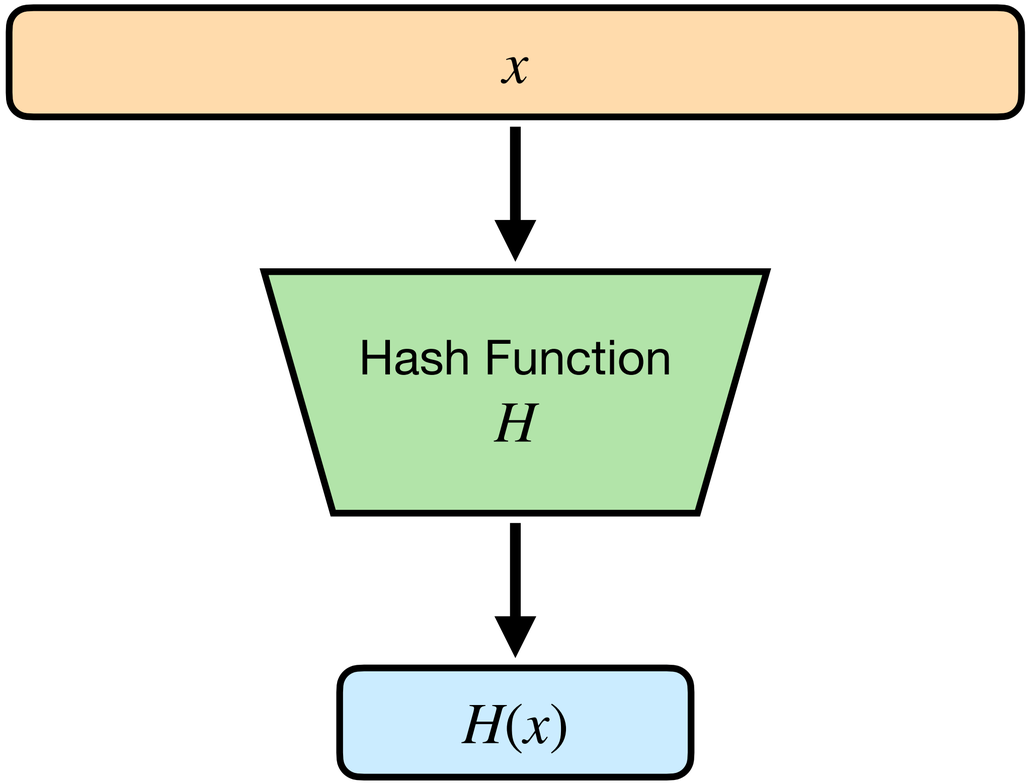
Why might we not want to use these types of hashes in cryptography?

- Design goals are different!
- In data structures:
  - Collision resistance is a desire (better performance) but not crucial
  - Elements are chosen independently of the hash function
- In cryptography:
  - The ability to find collisions yields attacks
  - Adversary chooses inputs with the explicit goal of finding a collision

# Collision-Resistant Hash Functions

Goals:

- Compress arbitrarily-long inputs into short fixed-length outputs
- It should be hard to find a collision $x \neq x'$ such that $H(x) = H(x')$



# Collision-Resistant Hash Functions

**Syntax**: $\Phi = (\text{Gen}, H)$

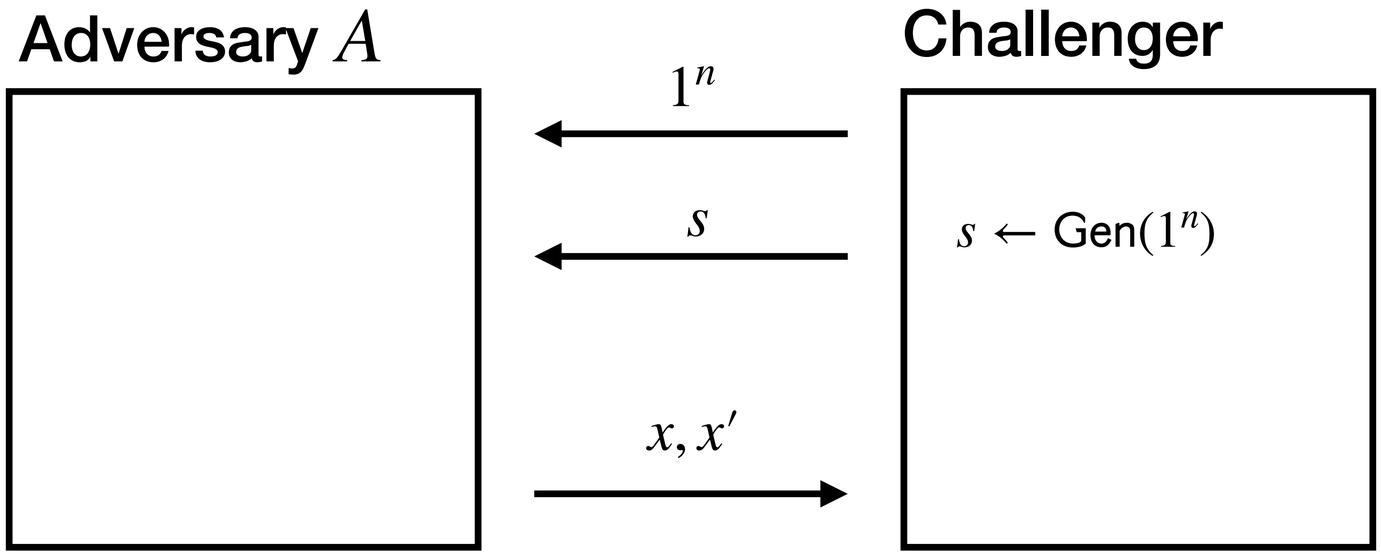
- **Key generation**: (randomized) algorithm $\text{Gen}(1^n)$ outputs a key $s$
- **Evaluation**: (deterministic) algorithm $H$ takes input $s$ and $x \in \{0,1\}^*$ and outputs $H^s(x) = H(s, x) \in \{0,1\}^{\ell(n)}$

If $H^s$ is defined only for inputs $x \in \{0,1\}^{\ell'(n)}$ and $\ell'(x) > \ell(n)$, then we say that $(\text{Gen}, H)$ is a fixed-length hash function for inputs of length $\ell'(n)$

- Also known as a compression function

# Collision-Resistant Hash Functions

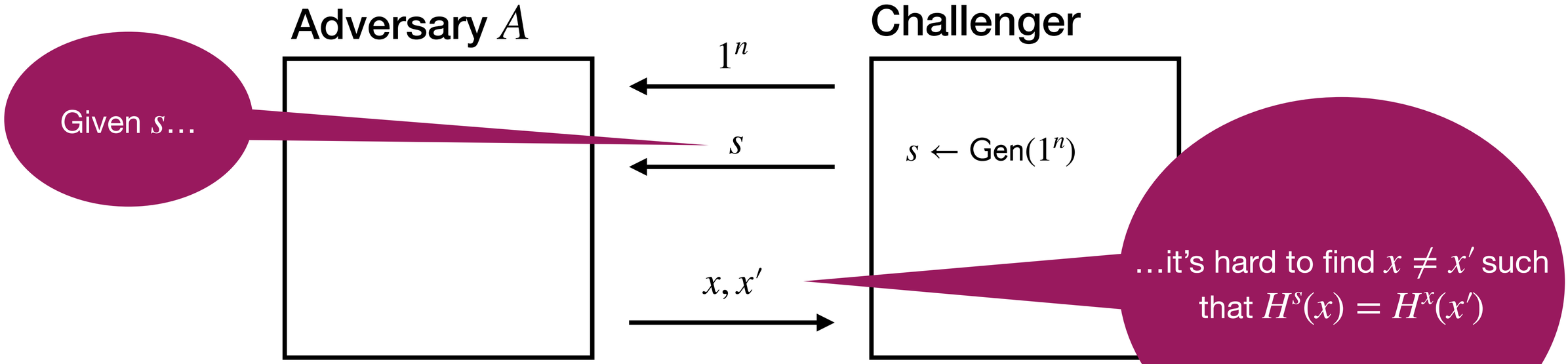
Given $\Phi = (\mathsf{Gen}, H)$ and an adversary $A$, consider the experiment $\mathsf{HashColl}_{\Phi,A}(n)$:




$$\mathsf{HashColl}_{\Phi,A}(n) = \begin{cases} 1 & H^s(x) = H^s(x') \text{ and } x \neq x' \\ 0 & \text{otherwise} \end{cases}$$

# Collision-Resistant Hash Functions

Given $\Phi = (\text{Gen}, H)$ and an adversary $A$, consider the experiment $\text{HashColl}_{\Phi,A}(n)$:



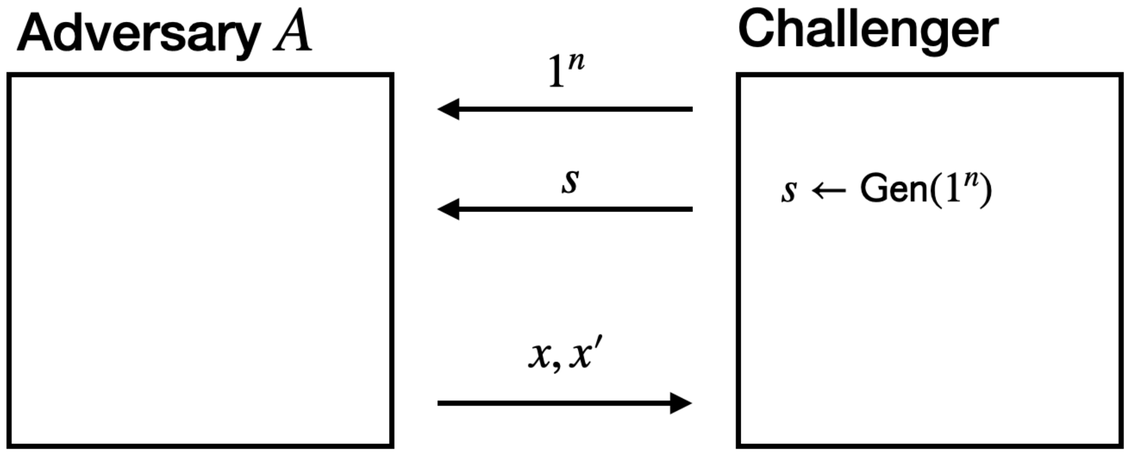

$$\text{HashColl}_{\Phi,A}(n) = \begin{cases} 1 & H^s(x) = H^s(x') \text{ and } x \neq x' \\ 0 & \text{otherwise} \end{cases}$$

# Collision-Resistant Hash Functions

**Definition**:

A **collision-resistant hash function (CRHF)** is a pair of polynomial-time algorithms $\Phi = (\text{Gen}, H)$ s.t. $\Phi$ satisfies collision resistance. That is, for every PPT adversary $A$ there exists a negligible function $\epsilon(\cdot)$ such that

$$\Pr[\text{HashColl}_{\Phi,A}(n) = 1] \leq \epsilon(n)$$




$$\text{HashColl}_{\Phi,A}(n) = \begin{cases} 1 & H^s(x) = H^s(x') \text{ and } x \neq x' \\ 0 & \text{otherwise} \end{cases}$$

# An aside: Why do we need $s$?

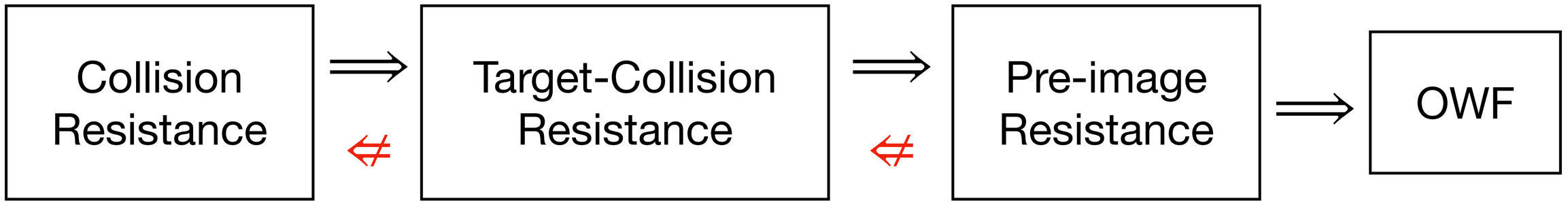
- In practice, hash functions are unkeyed (e.g., SHA256)
- However, in our definition, if we don't have $s$, we can consider a class of adversaries that have a collision hard-coded
  - Let $A_{x_1,x_2}$ be an algorithm that outputs $(x_1, x_2)$. There *exists* some $x_1, x_2$ s.t. $x_1, x_2$ is a collision (This gets into non-uniform vs uniform adversaries, which we won't talk about in this class)
- In practice, we don't know how to find this collision, so we define the keyed notion with the key known
  - Can think of hashes in practice as implicitly having a key that is published as part of the hash specification

# An aside: An application of hash functions

- Equality checking / fingerprinting
  - e.g., Storage or deduplication
  - Password storage
    - Storing in the clear is bad
    - Could store $H^s(pwd)$
      - Better, but still not great
    - Could store $(salt, H^s(salt||pwd))$
      - Typically what happens

# Weaker Notions of Security (informally)

- **Collision-resistant (CR)**
  - It is hard to find $x \neq x'$ such that $H^s(x) = H^s(x')$
- **Second pre-image resistance / Target-collision resistant (TCR)**
  - Given $x$ it is hard to find $x'$ such that $H^s(x) = H^s(x')$
- **Pre-image resistance / One-Wayness (OW)**
  - Given $y$ it is hard to find $x$ such that $H^s(x) = y$



# Generic Attacks

# Generic Attacks: Exhaustive Search

Given a hash function $H^s : \{0,1\}^* \rightarrow \{0,1\}^{\ell(n)}$, can brute force trying inputs until a collision

- There are $2^{\ell(n)}$ potential outputs for $H^s$
- By pigeonhole principle, a collision is guaranteed after testing $2^{\ell(n)} + 1$ values

# Generic Attacks: Birthday Attack

Given a hash function $H^s : \{0,1\}^* \rightarrow \{0,1\}^{\ell(n)}$,
can try $O(\sqrt{2^{\ell(n)}})$ random inputs

- By birthday paradox, finds a collision with probability $> 0.99$

- Since $\sqrt{2^{\ell(n)}} = 2^{\ell(n)/2}$, would need to choose the output length to be double whatever "level of security" we want

# Implication to Common Hash Functions

| Hash Function | Year | Digest Length | Security |
|---|---|---|---|
| MD4 | 1990 | 128 bits | 64 bits |
| MD5 | 1992 | 128 bits | 64 bits |
| SHA-1 | 1995 | 160 bits | 80 bits |
| SHA-256 | 2001 | 256 bits | 128 bits |
| SHA-512 | 2001 | 512 bits | 256 bits |
| SHA-3 | 2015 | 256/512 bits | 128/256 bits |

# Implication to Common Hash Functions

Completely broken!

SHA-2

| Hash Function | Year | Digest Length | Security |
|---|---|---|---|
| MD4 | 1990 | 128 bits | 64 bits |
| MD5 | 1992 | 128 bits | 64 bits |
| SHA-1 | 1995 | 160 bits | 80 bits |
| SHA-256 | 2001 | 256 bits | 128 bits |
| SHA-512 | 2001 | 512 bits | 256 bits |
| SHA-3 | 2015 | 256/512 bits | 128/256 bits |

# Domain Extension

# Domain Extension

Suppose we had a $F^s : \{0,1\}^{2n} \rightarrow \{0,1\}^n$ for fixed-length input.

(We'll see later some constructions for fixed-length messages)

Can we construct a $H^s : \{0,1\}^* \rightarrow \{0,1\}^n$ for arbitrary-length input?

# Domain Extension

Suppose we had a $F^s : \{0,1\}^{2n} \to \{0,1\}^n$ for fixed-length input.

(We'll see later some constructions for fixed-length messages)

Can we construct a $H^s : \{0,1\}^* \to \{0,1\}^n$ for arbitrary-length input?

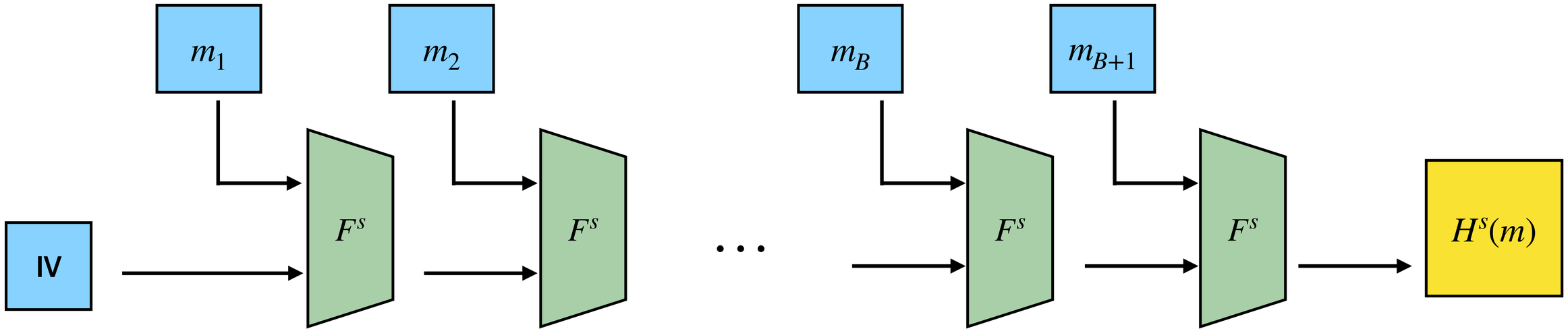
We will see the Merkle-Damgård transformation

# Merkle-Damgård Transform

Let $(\mathsf{Gen}, F)$ be a fixed-length CRHF such that $F^s : \{0,1\}^{2n} \to \{0,1\}^n$.

Construct $H^s : \{0,1\}^* \to \{0,1\}^n$ as follows:

- Given $m \in \{0,1\}^*$ of length $L = |m| \leq 2^n$.
- Let $B = \left\lceil \frac{L}{n} \right\rceil$ (pad $m$ with zeros if needed). Write $m = m_1 \ldots m_B$ where $m_i \in \{0,1\}^n$.
- Compute:

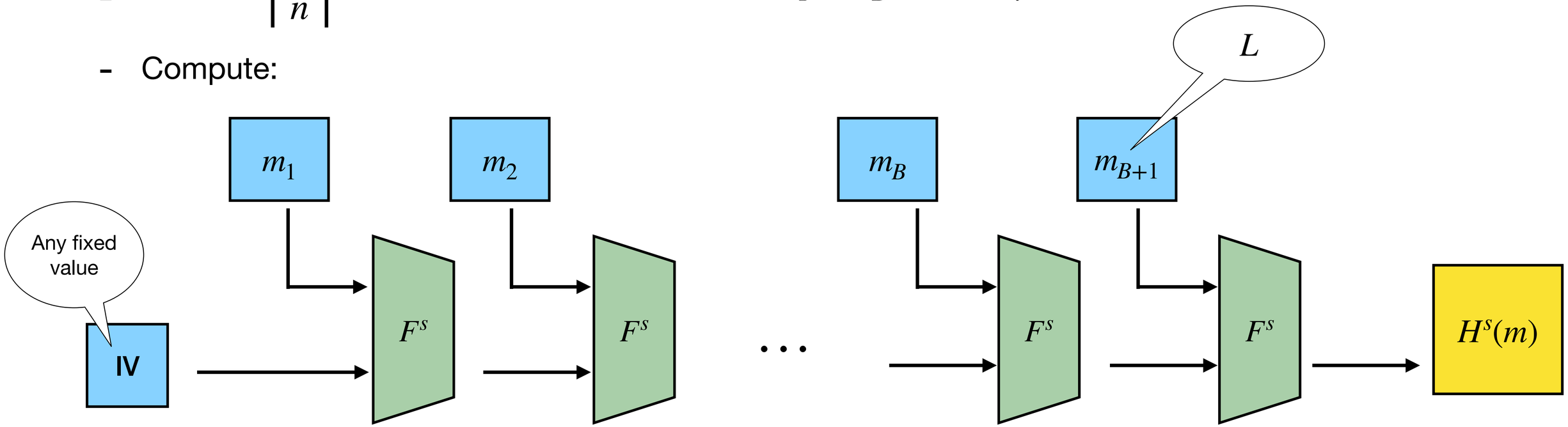


# Merkle-Damgård Transform

Let (Gen, $F$) be a fixed-length CRHF such that $F^s : \{0,1\}^{2n} \to \{0,1\}^n$.

Construct $H^s : \{0,1\}^* \to \{0,1\}^n$ as follows:

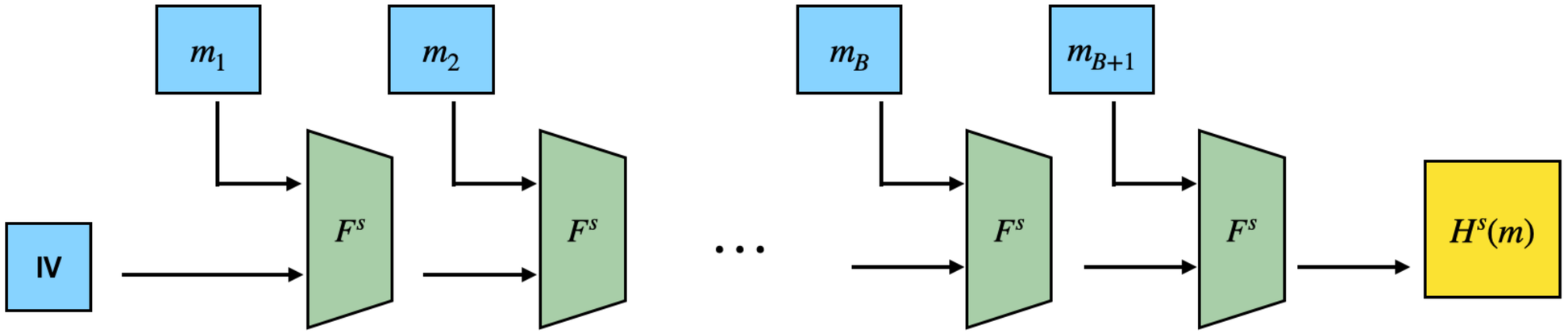
- Given $m \in \{0,1\}^*$ of length $L = |m| \leq 2^n$.
- Let $B = \left\lceil \frac{L}{n} \right\rceil$ (pad $m$ with zeros if needed). Write $m = m_1 \ldots m_B$ where $m_i \in \{0,1\}^n$.
- Compute:



# Merkle-Damgård Transform

**Theorem**: Let (Gen, $F$) be a fixed-length CRHF such that $F : \{0,1\}^{2n} \to \{0,1\}^n$. Then (Gen, $H$) is a CRHF where $H^s : \{0,1\}^* \to \{0,1\}^n$.

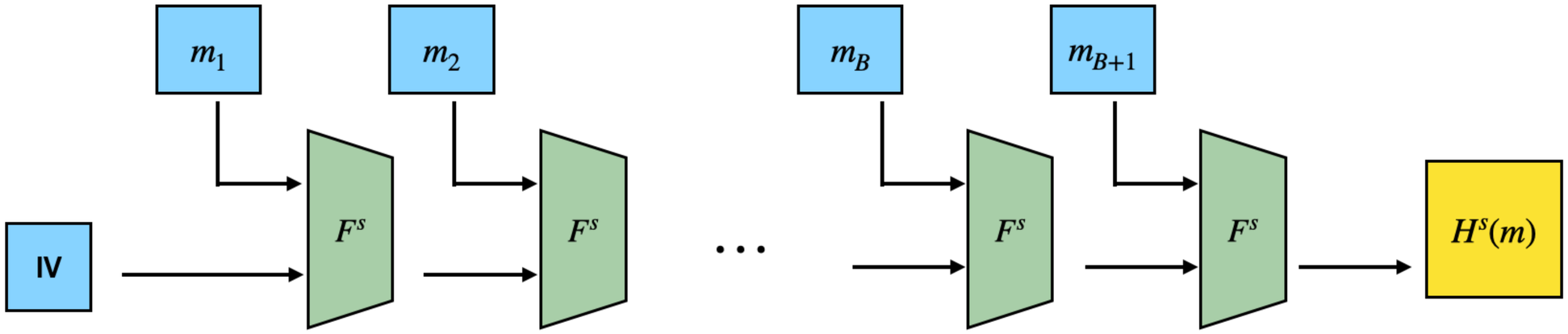
**Proof idea:** Assume $H^s$ is *not* collision-resistant. That is, there exists an adversary that can output a collision with non-negl probability. We can that adversary to break collision resistance of $F^s$.



# Merkle-Damgård Transform

**Proof idea:** Assume $H^s$ is *not* collision-resistant. That is, there exists an adversary that can output a collision with non-negl probability. We can that adversary to break collision resistance of $F^s$.

If $H^s$ is not collision-resistant, we can find $m \neq m'$ such that $H^s(m) = H^s(m')$

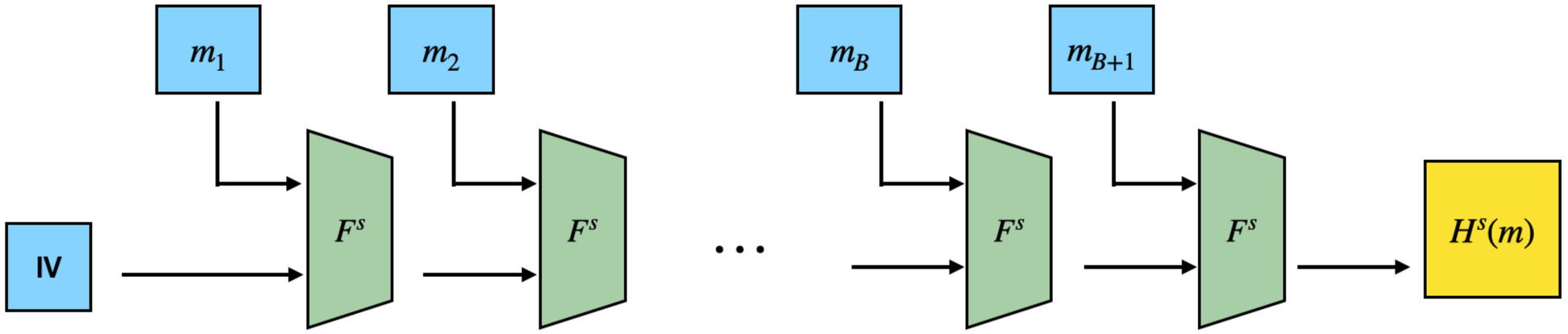


# Merkle-Damgård Transform

**Proof idea:** Assume $H^s$ is *not* collision-resistant. That is, there exists an adversary that can output a collision with non-negl probability. We can that adversary to break collision resistance of $F^s$.

If $H^s$ is not collision-resistant, we can find $m \neq m'$ such that $H^s(m) = H^s(m')$

**Case 1**: If $|m| \neq |m'|$, then there is a collision for $F^s$ in the last block that contains the size



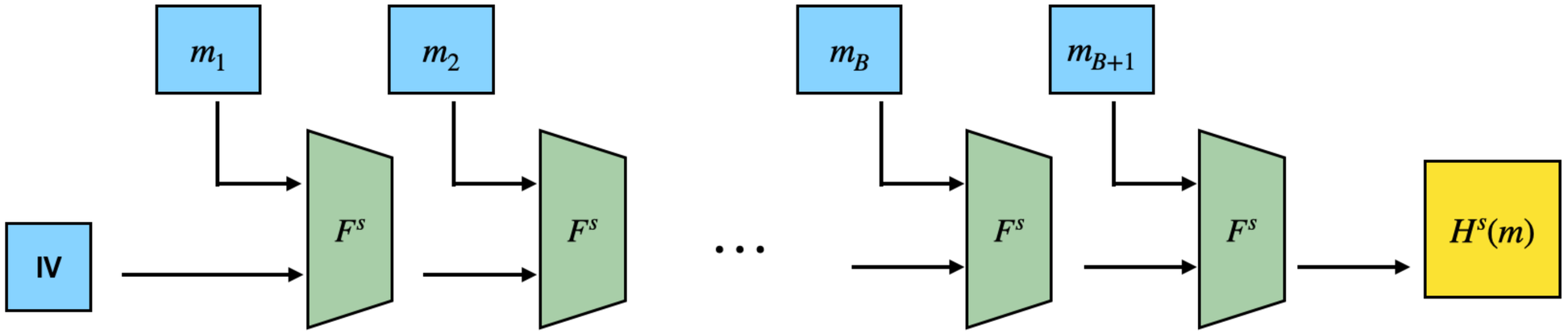
# Merkle-Damgård Transform

**Proof idea:** Assume $H^s$ is *not* collision-resistant. That is, there exists an adversary that can output a collision with non-negl probability. We can that adversary to break collision resistance of $F^s$.

If $H^s$ is not collision-resistant, we can find $m \neq m'$ such that $H^s(m) = H^s(m')$

**Case 1**: If $|m| \neq |m'|$, then there is a collision for $F^s$ in the last block that contains the size

**Case 2**: If $|m| = |m'|$, then go backwards and check each block. A collision for $F^s$ must exist in an earlier block otherwise $m = m'$.



# Hash and Authenticate

# Authenticating Arbitrary-Length Messages

$$m = \boxed{m_1}\boxed{m_2}\boxed{\phantom{m}}\boxed{\ldots}\boxed{\phantom{m}}\boxed{\ldots}\boxed{\phantom{m}}\boxed{m_d}$$

Suppose we had a $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ for fixed-length messages and a collision-resistant hash function $\Phi = (\mathsf{Gen}_H, H)$.

Can we construct a $(\hat{\mathsf{Gen}}, \hat{\mathsf{Mac}}, \hat{\mathsf{Verify}})$ for arbitrary-length messages?

# Authenticating Arbitrary-Length Messages

Let $\Pi = (\text{Gen}, \text{Mac}, \text{Verify})$ be a fixed-length MAC and let $\Phi = (\text{Gen}_H, H)$ be a keyed hash function.

Consider the following MAC scheme $\hat{\Phi} = (\hat{\text{Gen}}, \hat{\text{Mac}}, \hat{\text{Verify}})$ for arbitrary-length messages:

- **Key generation**: On input $1^n$, sample $k \leftarrow \text{Gen}(1^n)$ and $s \leftarrow \text{Gen}_H(1^n)$ and output $(k, s)$.

# Authenticating Arbitrary-Length Messages

Let $\Pi = (\text{Gen}, \text{Mac}, \text{Verify})$ be a fixed-length MAC and let $\Phi = (\text{Gen}_H, H)$ be a keyed hash function.

Consider the following MAC scheme $\hat{\Phi} = (\hat{\text{Gen}}, \hat{\text{Mac}}, \hat{\text{Verify}})$ for arbitrary-length messages:

- **Key generation**: On input $1^n$, sample $k \leftarrow \text{Gen}(1^n)$ and $s \leftarrow \text{Gen}_H(1^n)$ and output $(k, s)$.
- **Tag generation**: On input $(k, s)$ and $m \in \{0,1\}^*$ output $t = \text{Mac}_k(H^s(m))$

# Authenticating Arbitrary-Length Messages

Let $\Pi = (\text{Gen}, \text{Mac}, \text{Verify})$ be a fixed-length MAC and let $\Phi = (\text{Gen}_H, H)$ be a keyed hash function.

Consider the following MAC scheme $\hat{\Phi} = (\hat{\text{Gen}}, \hat{\text{Mac}}, \hat{\text{Verify}})$ for arbitrary-length messages:

- **Key generation**: On input $1^n$, sample $k \leftarrow \text{Gen}(1^n)$ and $s \leftarrow \text{Gen}_H(1^n)$ and output $(k, s)$.
- **Tag generation**: On input $(k, s)$ and $m \in \{0,1\}^*$ output $t = \text{Mac}_k(H^s(m))$
- **Verification**: On input $(k, s)$, $m \in \{0,1\}^*$, and $t \in \{0,1\}^*$, output $\text{Verify}_k(H^s(m), t)$

# Authenticating Arbitrary-Length Messages

Let $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ be a fixed-length MAC and let $\Phi = (\mathsf{Gen}_H, H)$ be a keyed hash function.

Consider the following MAC scheme $\hat{\Phi} = (\hat{\mathsf{Gen}}, \hat{\mathsf{Mac}}, \hat{\mathsf{Verify}})$ for arbitrary-length messages:

- **Key generation**: On input $1^n$, sample $k \leftarrow \mathsf{Gen}(1^n)$ and $s \leftarrow \mathsf{Gen}_H(1^n)$ and output $(k, s)$.
- **Tag generation**: On input $(k, s)$ and $m \in \{0,1\}^*$ output $t = \mathsf{Mac}_k(H^s(m))$
- **Verification**: On input $(k, s)$, $m \in \{0,1\}^*$, and $t \in \{0,1\}^*$, output $\mathsf{Verify}_k(H^s(m), t)$

**Theorem**: If $\Pi$ is a secure MAC and $\Phi$ is collision resistant, then $\hat{\Pi}$ is a secure MAC.

**Proof idea**: Given any forger for $\hat{\Pi}$, you can either forge for $\Pi$ or find a collision for $\Pi$

# Next Time

- Monday
  - Hash functions
- Wednesday
  - More hash functions