COMS BC3262: Introduction to Cryptography

Lecture 11: Collision-Resistant Hash Functions

February 25, 2026

Logistics: Problem Sets

- PS 1 grades are out!
  - If you notice a mistake in the grading, please let me know by March 9
  - Regrades for PS 1 will be closed after that date!
- I expect to finish grading PS 2 before the midterm
- PS 3 is due next week Thursday, March 5

3262 students patiently waiting for their assignments to be graded

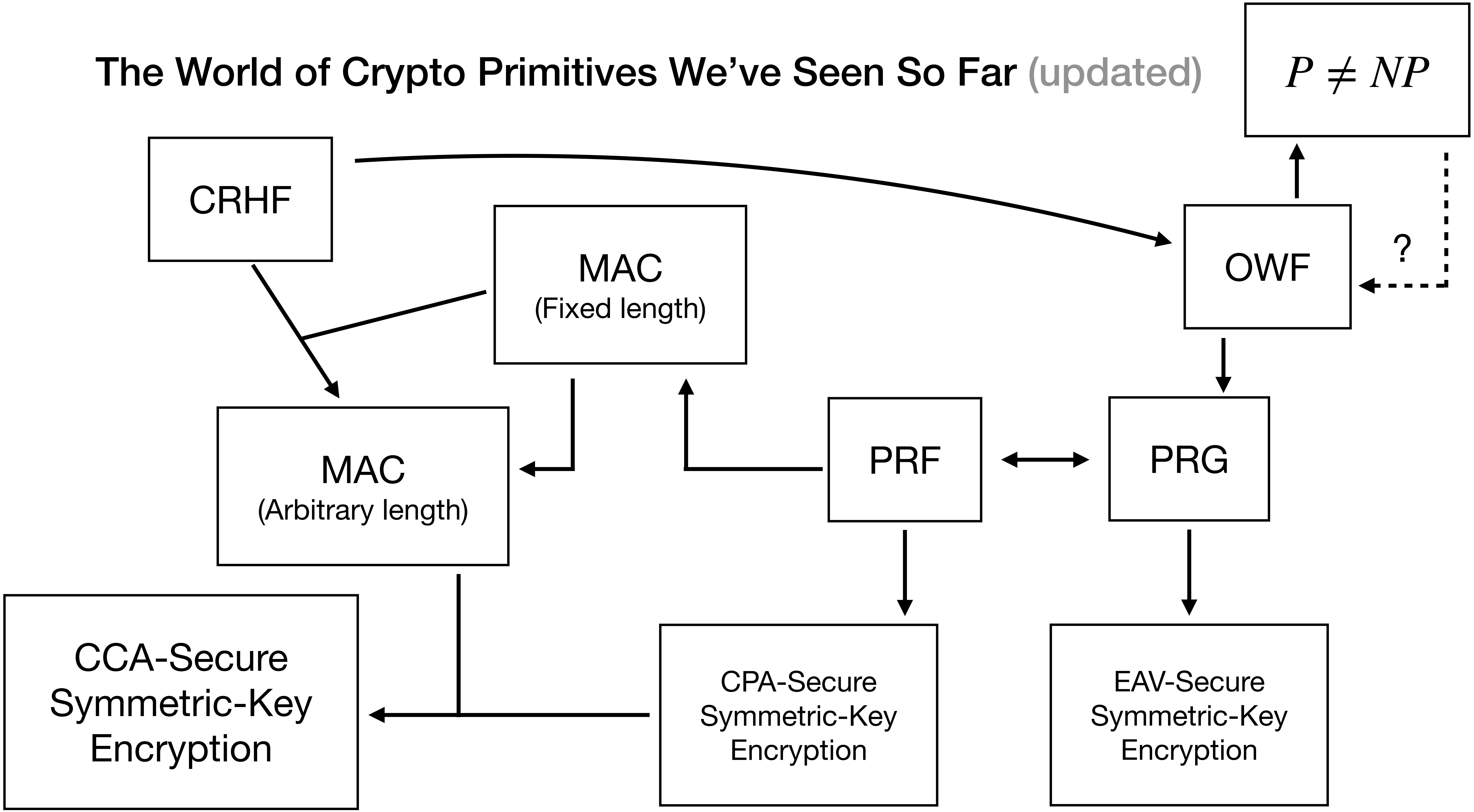
Logistics: Midterm

- There will be an **in-class written midterm** on **Wednesday, March 11**
  - *Please contact CARDS or ODS as soon as possible if you need accommodations*
  - It is worth 25% of your grade
- You may bring a single letter-sized reference sheet (double-sided)
  - You will be expected to submit your reference sheet along with your exam
- Exam is closed note, no technology, no collaboration
- Exam will not cover any material introduced *after* Lecture 12 (next one)

Looking back and looking forward

- Lecture before the exam (Monday, March 9) will be a review session
    - We’ll go over all of the problem set answers again
    - Come with questions!
    - Lecture may end early if no one has questions

The World of Crypto Primitives We've Seen So Far (updated)



Today’s Lecture

- CRHF Review
  - Merkle-Damgård (Domain Extension)
- Hash and Authenticate
- Random Oracle Model
- More on Applications

Collision-Resistant Hash Functions

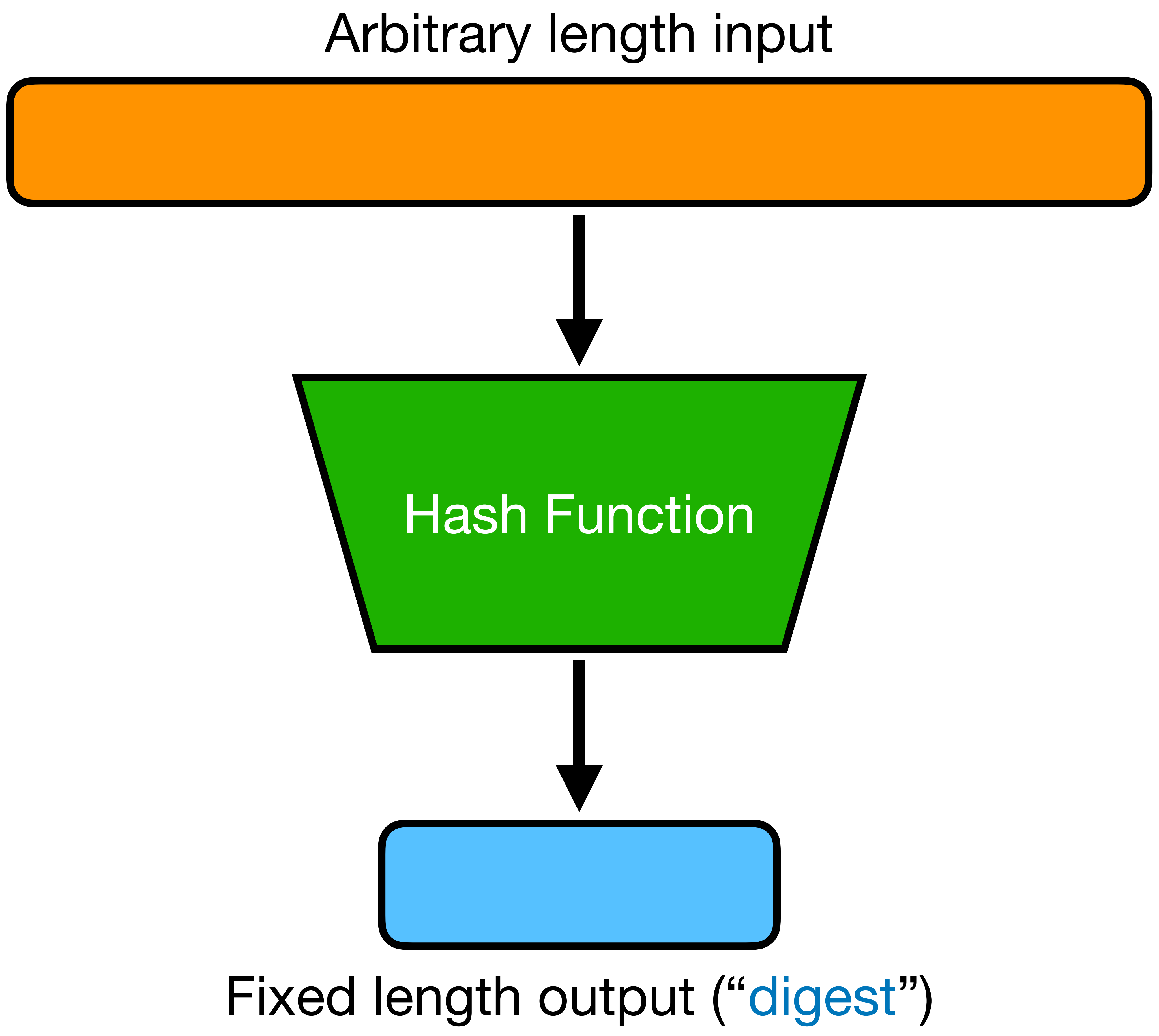
Hash Functions

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

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

Desired properties:

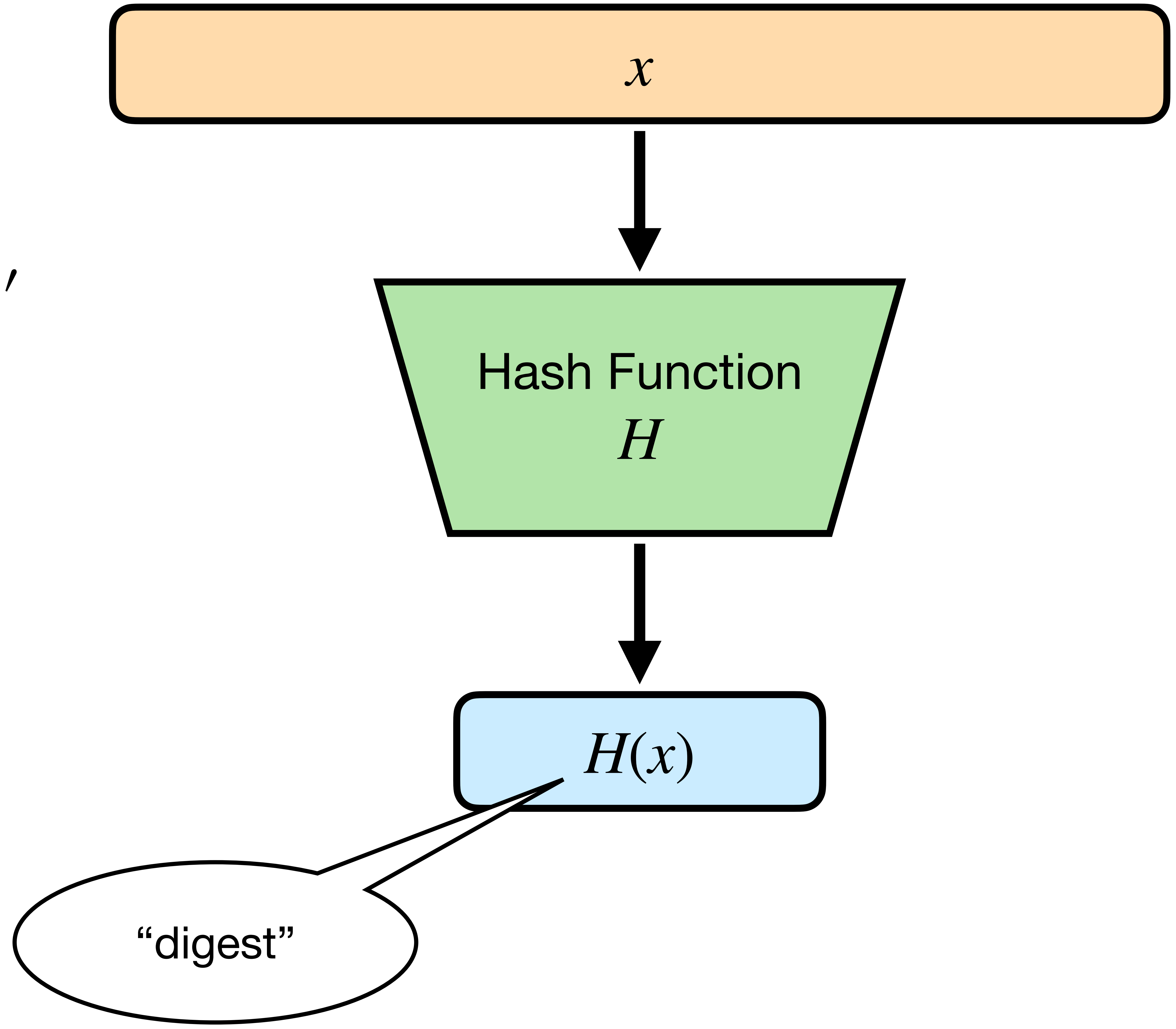
- Compressing
- Given $x \in \{0,1\}^*$ and $y \in \{0,1\}^n$ can verify whether $y = H(x)$
- Output is distributed "randomly" (minimizes collisions)



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

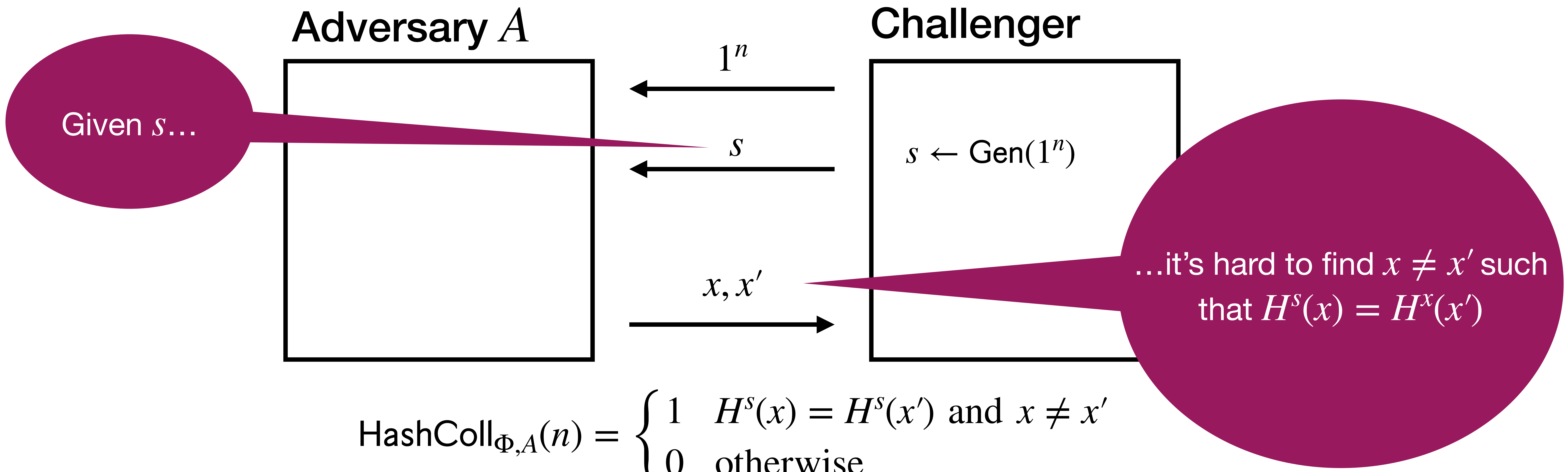
- **Key generation:** (randomized) algorithm $\text{Gen}(1^n)$ outputs a key s
- **Evaluation:** (deterministic) algorithm H takes input s and $x \in \{0,1\}^*$ and outputs $H^s(x) = H(s, x) \in \{0,1\}^{\ell(n)}$

If H^s is defined only for inputs $x \in \{0,1\}^{\ell'(n)}$ and $\ell'(x) > \ell(n)$, then we say that (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 $\Phi = (\text{Gen}, H)$ and an adversary A, consider the experiment $\text{HashColl}_{\Phi,A}(n)$:


Adversary A
Challenger
1^n
s
$s \leftarrow \text{Gen}(1^n)$
x, x'
Given s…
…it's hard to find $x \neq x'$ such that $H^s(x) = H^x(x')$

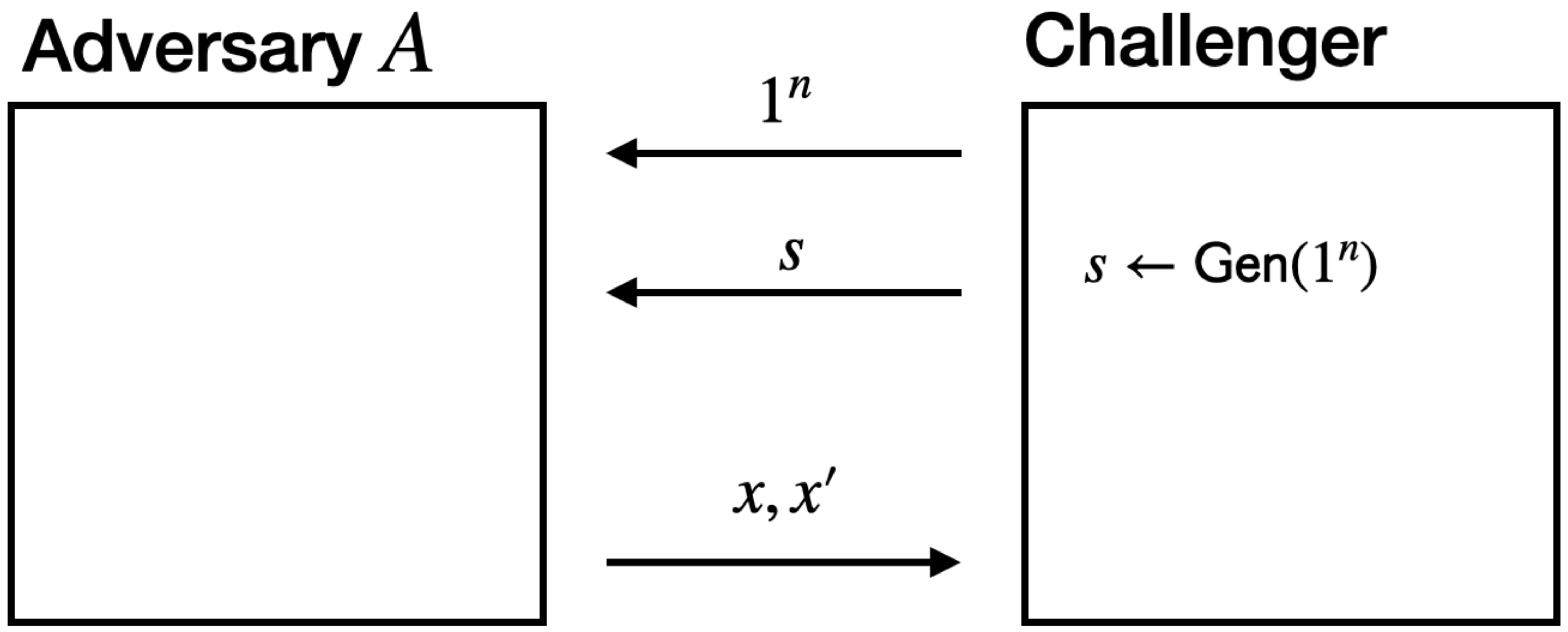
$$\text{HashColl}_{\Phi,A}(n) = \begin{cases} 1 & H^s(x) = H^s(x') \text{ and } x \neq x' \\ 0 & \text{otherwise} \end{cases}$$

Collision-Resistant Hash Functions

Definition:

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

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



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

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?

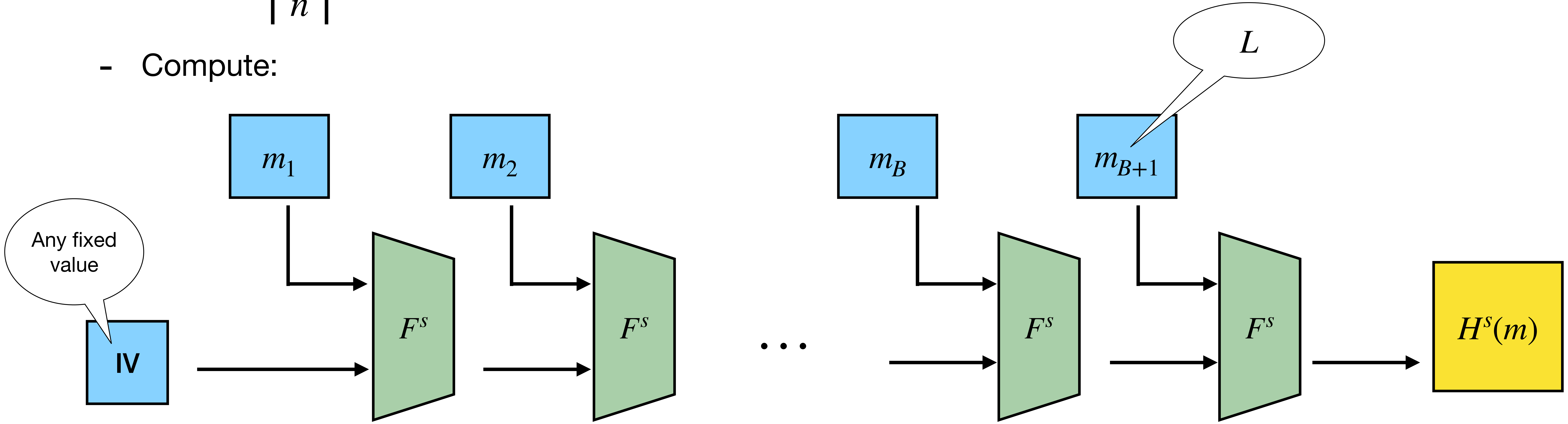
We will see the Merkle-Damgård transformation

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:



Hash and Authenticate

Authenticating Arbitrary-Length Messages

$$m = \begin{array}{|c|c|c|c|c|c|c|c|}\hline m_1 & m_2 & & \dots & & \dots & & m_d \\ \hline\end{array}$$

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 $(\widehat{\mathsf{Gen}}, \widehat{\mathsf{Mac}}, \widehat{\mathsf{Verify}})$ for arbitrary-length messages?

Hash-and-Authenticate

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).

Hash-and-Authenticate

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 $t = \mathsf{Mac}_k(H^s(m))$

Hash-and-Authenticate

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 $\mathsf{Verify}_k(H^s(m), t)$

Hash-and-Authenticate

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 Π is a secure MAC and Φ is collision resistant, then $\hat{\Pi}$ is a secure MAC.

Hash-and-Authenticate

Theorem:

If Π is a secure MAC and Φ is collision resistant, then $\hat{\Pi}$ is a secure MAC.

Hash-and-Authenticate

Theorem:

If Π is a secure MAC and Φ is collision resistant, then $\hat{\Pi}$ is a secure MAC.

Hash-and-Authenticate

Theorem:

If Π is a secure MAC and Φ is collision resistant, then $\hat{\Pi}$ is a secure MAC.

A and B implies C

Hash-and-Authenticate

Theorem:

If Π is a secure MAC and Φ is collision resistant, then $\hat{\Pi}$ is a secure MAC.

A and B implies C

What is the contrapositive of this?

Hash-and-Authenticate

Theorem:

If Π is a secure MAC and Φ is collision resistant, then $\hat{\Pi}$ is a secure MAC.

A and B implies C

What is the contrapositive of this?

Not C implies not A or not B

Hash-and-Authenticate

Theorem:

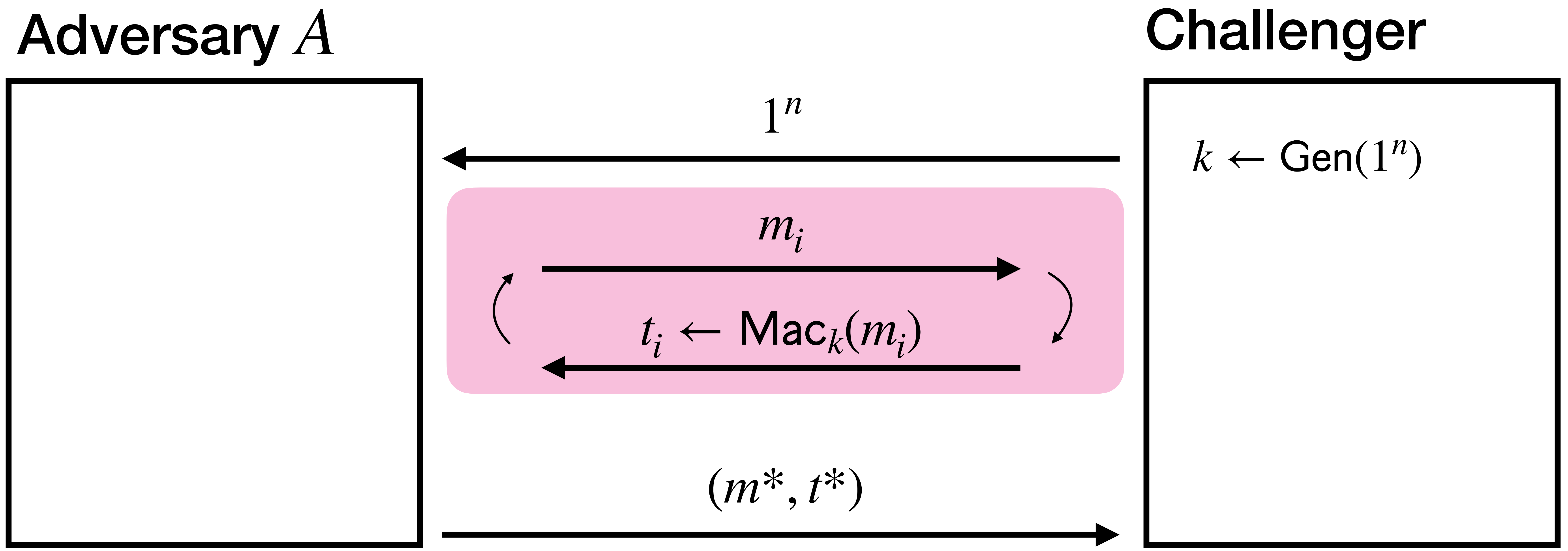
If Π is a secure MAC and Φ is collision resistant, then $\hat{\Pi}$ is a secure MAC.

Proof idea:

Given any forger $\hat{A}$ for $\hat{\Pi}$, you can construct either a forger A for Π or a collision-finder C for Φ

Recall: MAC Security

Let $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$. We define $\mathsf{MacForge}_{\mathscr{A},\Pi}(n)$ as follows



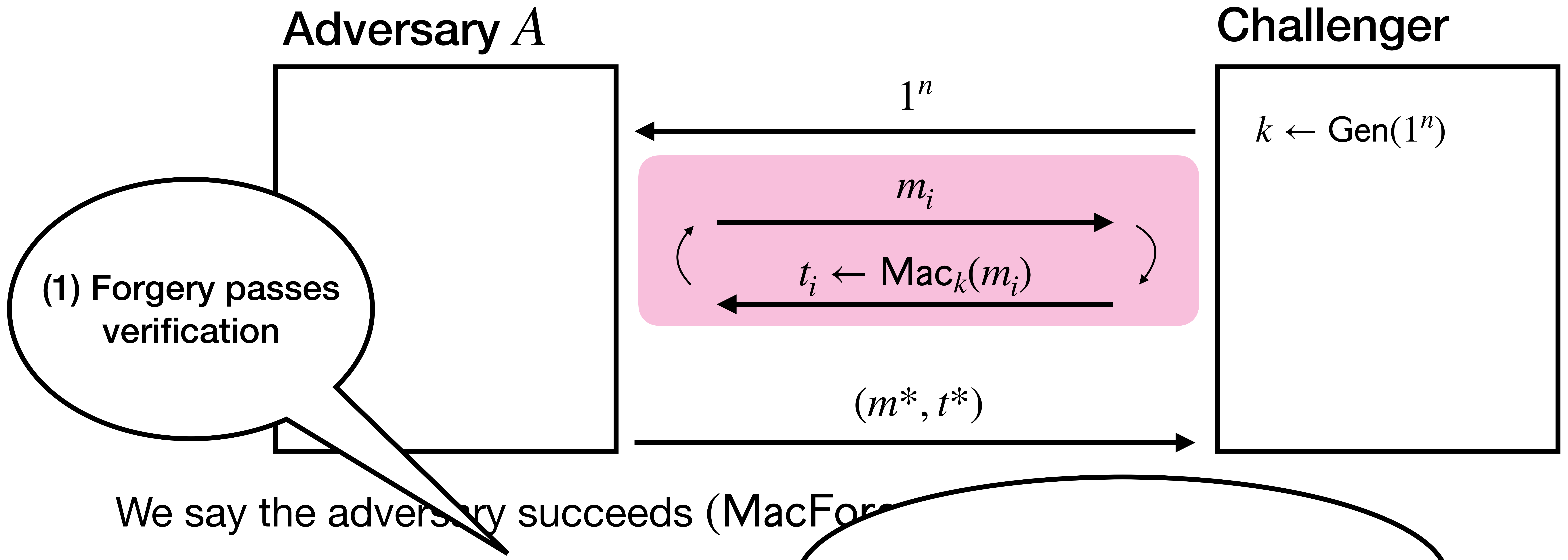
Adversary A
Challenger
1^n
$k \leftarrow \mathsf{Gen}(1^n)$
m_i
$t_i \leftarrow \mathsf{Mac}_k(m_i)$
(m^*, t^*)

We say the adversary succeeds ($\mathsf{MacForge}_{\mathscr{A},\Pi}(n) = 1$) if:

1. $\mathsf{Verify}_k(m^*, t^*) = 1$
2. $m^* \neq m_i$ for all queried m_i

Recall: MAC Security

Let $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$. We define $\mathsf{MacForge}_{\mathscr{A},\Pi}(n)$ as follows



Adversary A
Challenger
1^n
$k \leftarrow \mathsf{Gen}(1^n)$
m_i
$t_i \leftarrow \mathsf{Mac}_k(m_i)$
(m^*, t^*)
(1) Forgery passes verification

We say the advers ry succeeds (MacFor

1. $\mathsf{Verify}_k(m^*, t^*) = 1$
2. $m^* \neq m_i$ for all queried m_i

(2) Forgery is on a new message

Hash-and-Authenticate

Theorem:

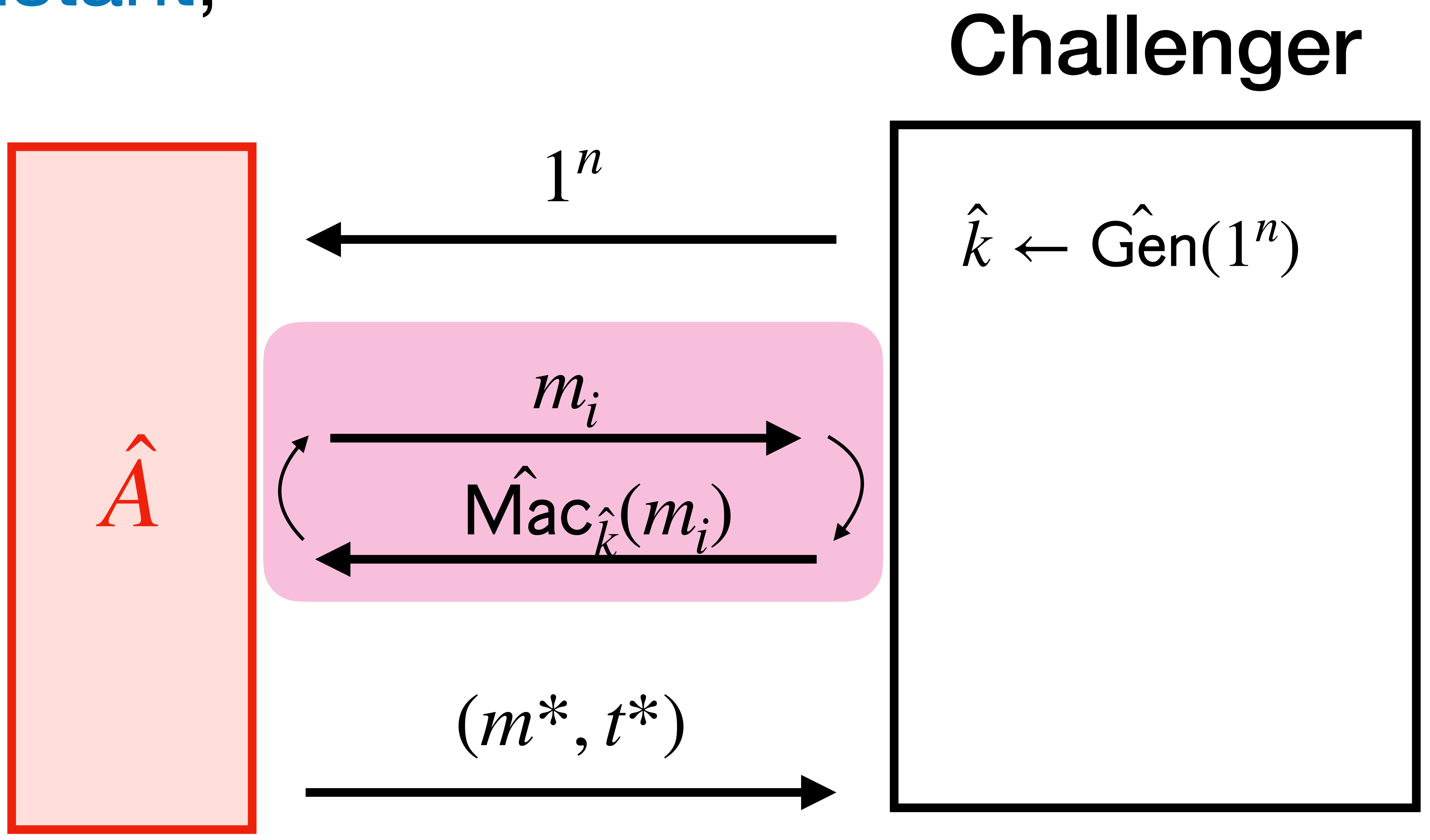
If Π is a secure MAC and Φ is collision resistant,
then $\hat{\Pi}$ is a secure MAC.

Proof idea:

Given any forger $\hat{A}$ for $\hat{\Pi}$, construct

either a forger A for Π

or a collision-finder C for Φ


Challenger
$\hat{k} \leftarrow \hat{\mathsf{Gen}}(1^n)$
$\hat{A}$
1^n
m_i
$\hat{\mathsf{Mac}}_{\hat{k}}(m_i)$
(m^*, t^*)

Forger wins if $\hat{\mathsf{Verify}}(m^*, t^*) = 1$
and $m^* \notin \{m_1, \ldots, m_q\}$

Hash-and-Authenticate

Theorem:

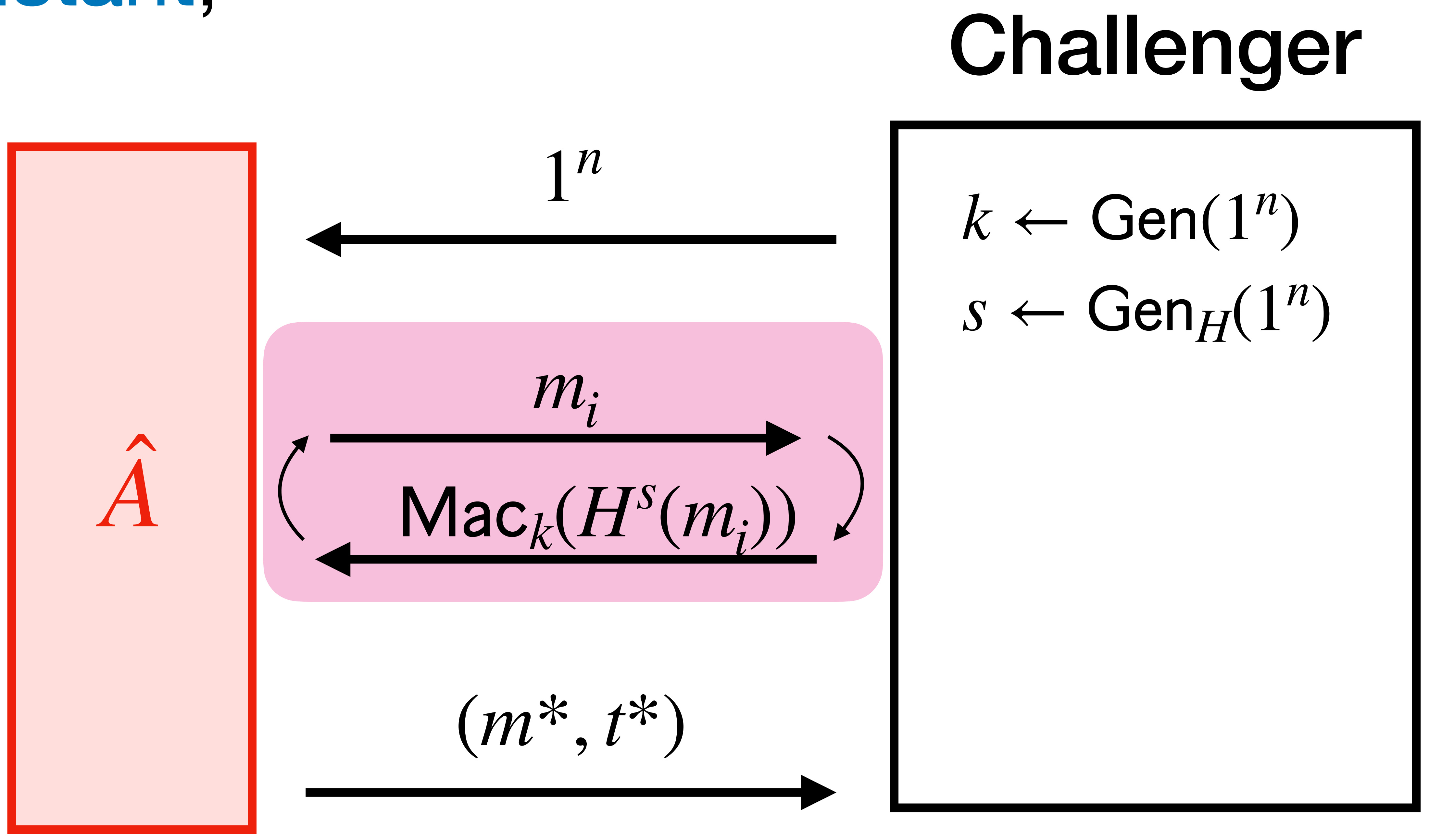
If Π is a secure MAC and Φ is collision resistant,
then $\hat{\Pi}$ is a secure MAC.

Proof idea:

Given any forger $\hat{A}$ for $\hat{\Pi}$, construct

either a forger A for Π

or a collision-finder C for Φ


Challenger
$k \leftarrow \mathsf{Gen}(1^n)$
$s \leftarrow \mathsf{Gen}_H(1^n)$
$\hat{A}$
1^n
m_i
$\mathsf{Mac}_k(H^s(m_i))$
(m^*, t^*)

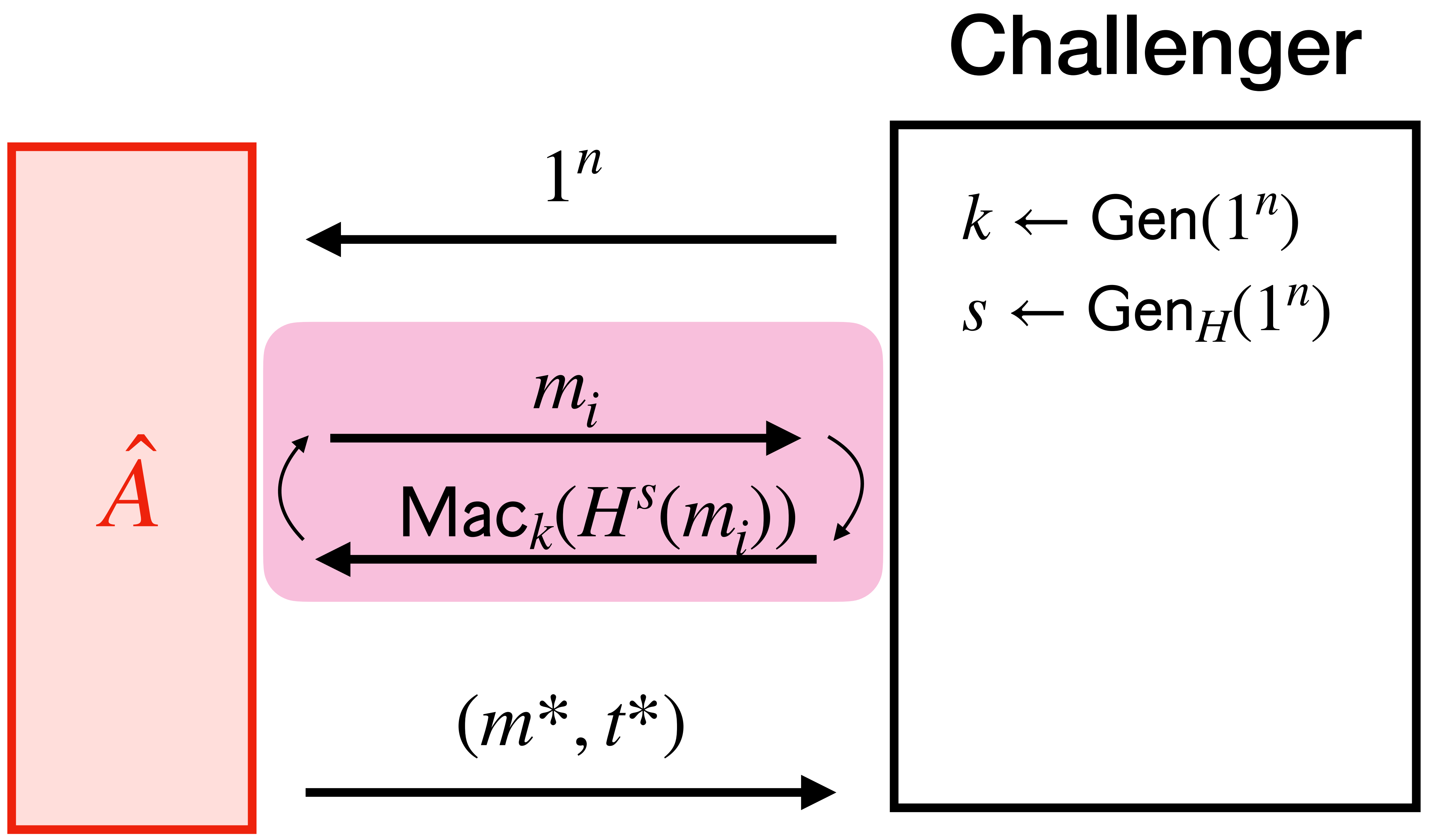
Forger wins if $\mathsf{Verify}(H^s(m^*), t^*) = 1$
and $m^* \notin \{m_1, \ldots, m_q\}$

Hash-and-Authenticate

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

Proof idea:

Forger $\hat{A}$ wins in the $\mathsf{MacForge}_{\hat{\mathscr{A}},\Pi}(n)$ game with a forgery (m^*, t^*) in one of two ways:


Challenger
$k \leftarrow \mathsf{Gen}(1^n)$
$s \leftarrow \mathsf{Gen}_H(1^n)$
$\hat{A}$
1^n
m_i
$\mathsf{Mac}_k(H^s(m_i))$
(m^*, t^*)

Forger wins if $\mathsf{Verify}(H^s(m^*), t^*) = 1$ and $m^* \notin \{m_1, \ldots, m_q\}$

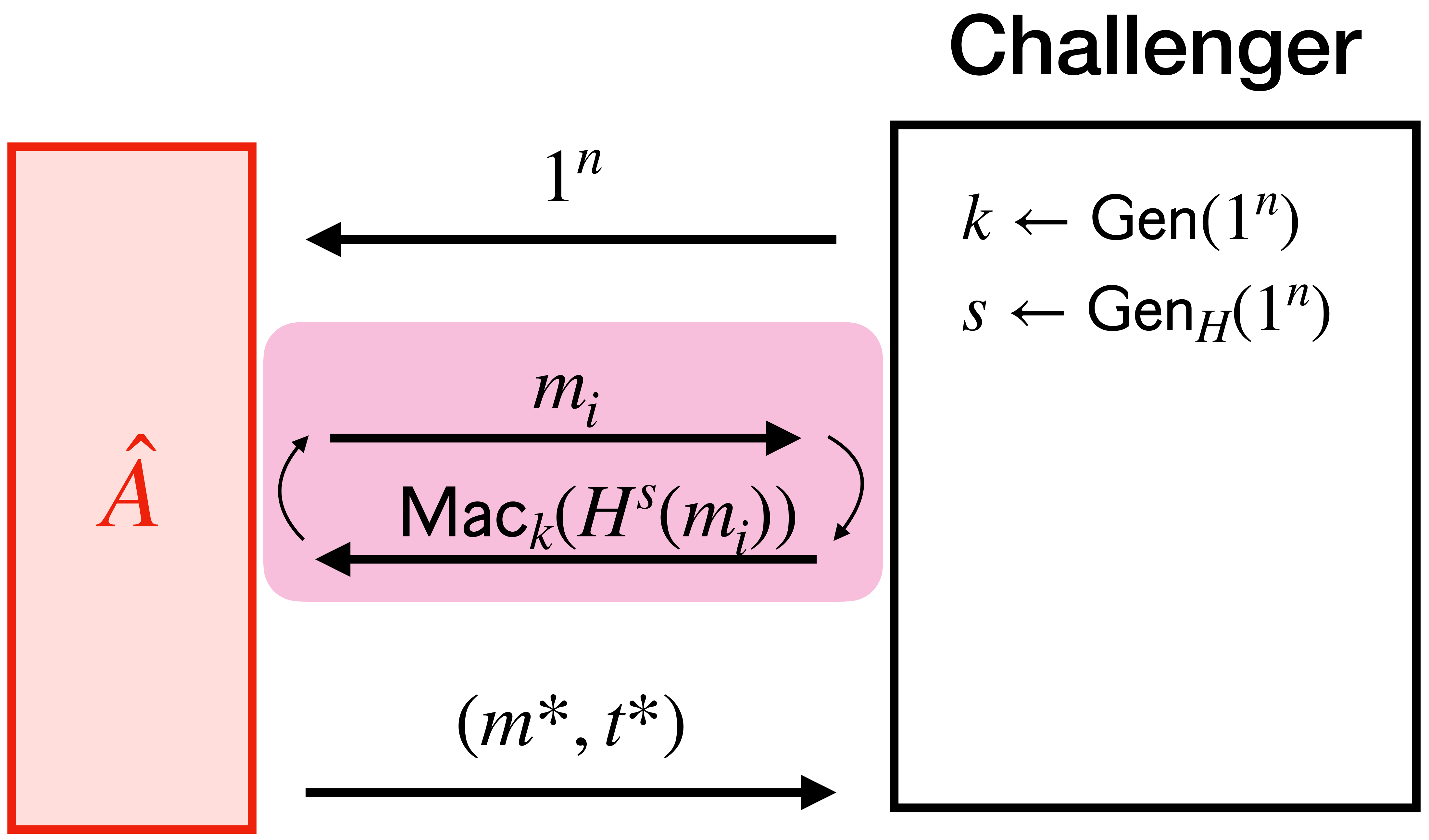
Hash-and-Authenticate

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

Proof idea:

Forger $\hat{A}$ wins in the $\mathsf{MacForge}_{\hat{\mathscr{A}},\Pi}(n)$ game with a forgery (m^*, t^*) in one of two ways:

1. There exists some previously queried m_i such that $H^s(m^*) = H^s(m_i)$ and $m_i \neq m^*$


Challenger
1^n
$k \leftarrow \mathsf{Gen}(1^n)$
$s \leftarrow \mathsf{Gen}_H(1^n)$
$\hat{A}$
m_i
$\mathsf{Mac}_k(H^s(m_i))$
(m^*, t^*)

Forger wins if $\mathsf{Verify}(H^s(m^*), t^*) = 1$ and $m^* \notin \{m_1, \ldots, m_q\}$

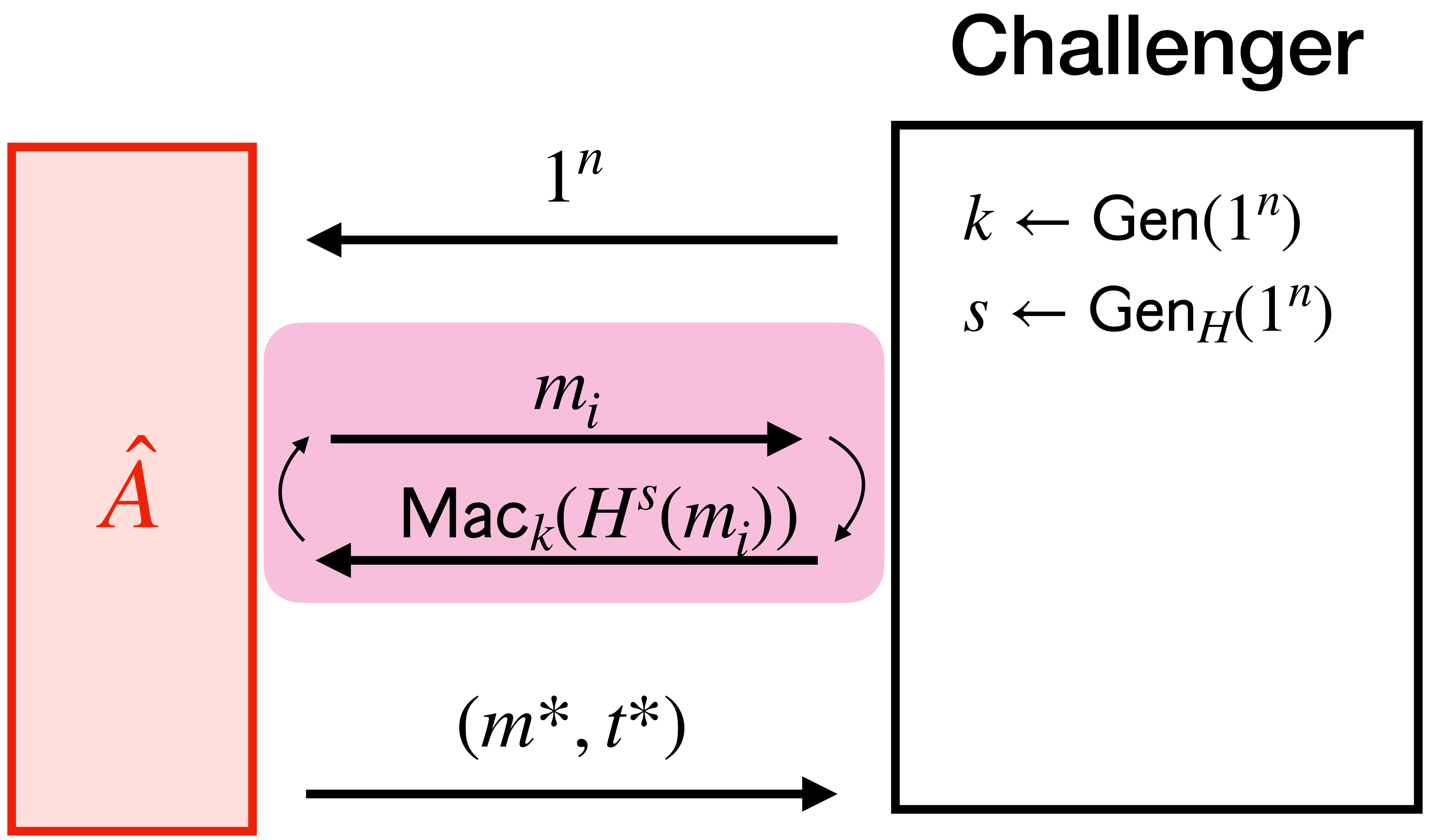
Hash-and-Authenticate

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

Proof idea:

Forger $\hat{A}$ wins in the $\mathsf{MacForge}_{\hat{\mathscr{A}},\Pi}(n)$ game with a forgery (m^*, t^*) in one of two ways:

1. There exists some previously queried m_i such that $H^s(m^*) = H^s(m_i)$ and $m_i \neq m^*$
2. $H^s(m^*) \notin \{H^s(m_1), \ldots, H^s(m_q)\}$ for all queried $m_1, \ldots, m_q$

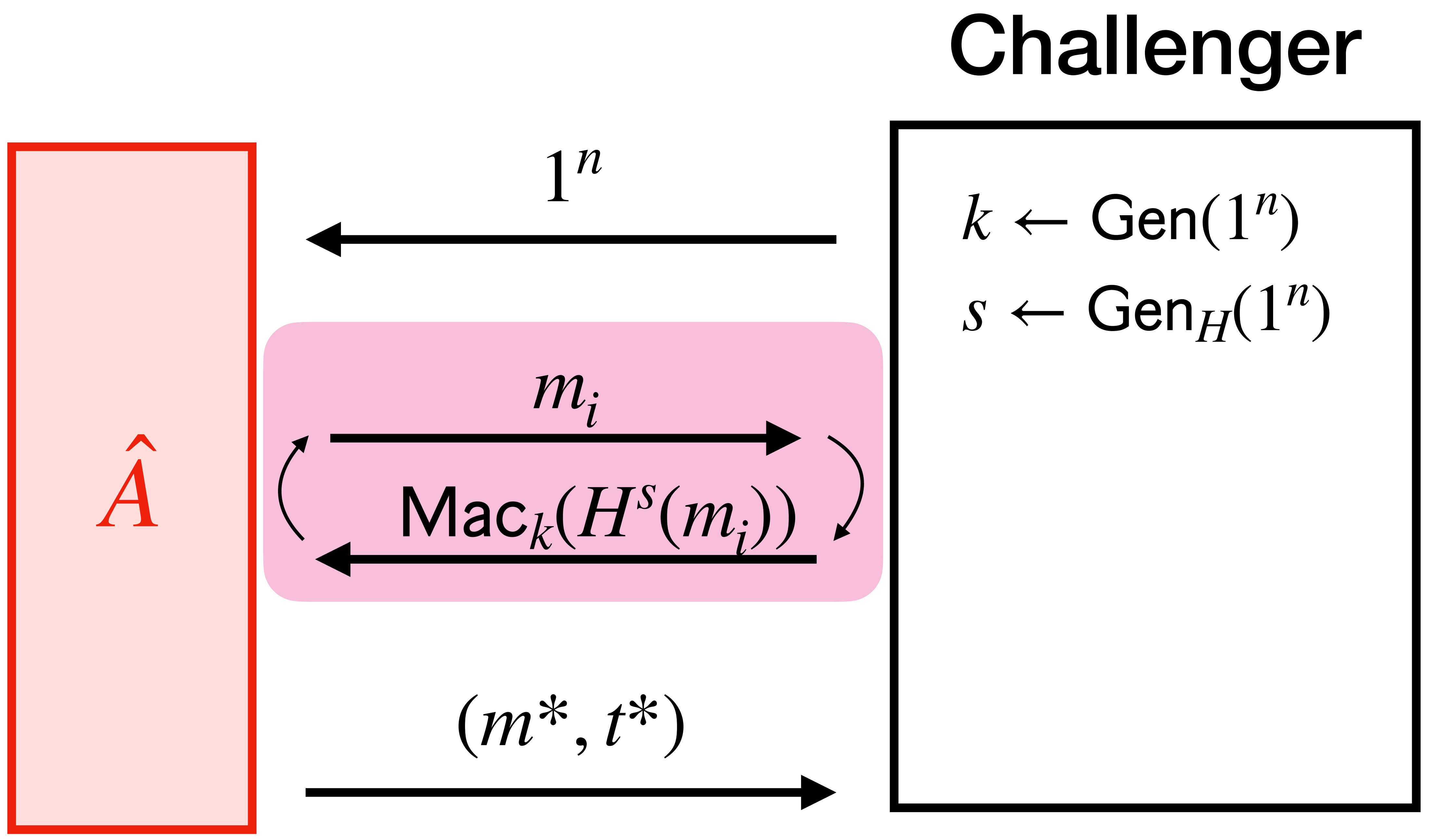

Challenger
$k \leftarrow \mathsf{Gen}(1^n)$
$s \leftarrow \mathsf{Gen}_H(1^n)$
$\hat{A}$
1^n
m_i
$\mathsf{Mac}_k(H^s(m_i))$
(m^*, t^*)

Forger wins if $\mathsf{Verify}(H^s(m^*), t^*) = 1$ and $m^* \notin \{m_1, \ldots, m_q\}$

Hash-and-Authenticate

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

$$\Pr[\mathsf{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1]$$
$$\leq \Pr[\mathit{collision}]$$
$$+ \Pr[\mathsf{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1 \wedge \neg \mathit{collision}]$$


Challenger
$\hat{A}$
1^n
$k \leftarrow \mathsf{Gen}(1^n)$
$s \leftarrow \mathsf{Gen}_H(1^n)$
m_i
$\mathsf{Mac}_k(H^s(m_i))$
(m^*, t^*)

Forger wins if $\mathsf{Verify}(H^s(m^*), t^*) = 1$ and $m^* \notin \{m_1, \ldots, m_q\}$

*Law of total probability: $\Pr[A] = \Pr[A \wedge B] + \Pr[A \wedge \neg B] \leq \Pr[B] + \Pr[A \wedge \neg B]$

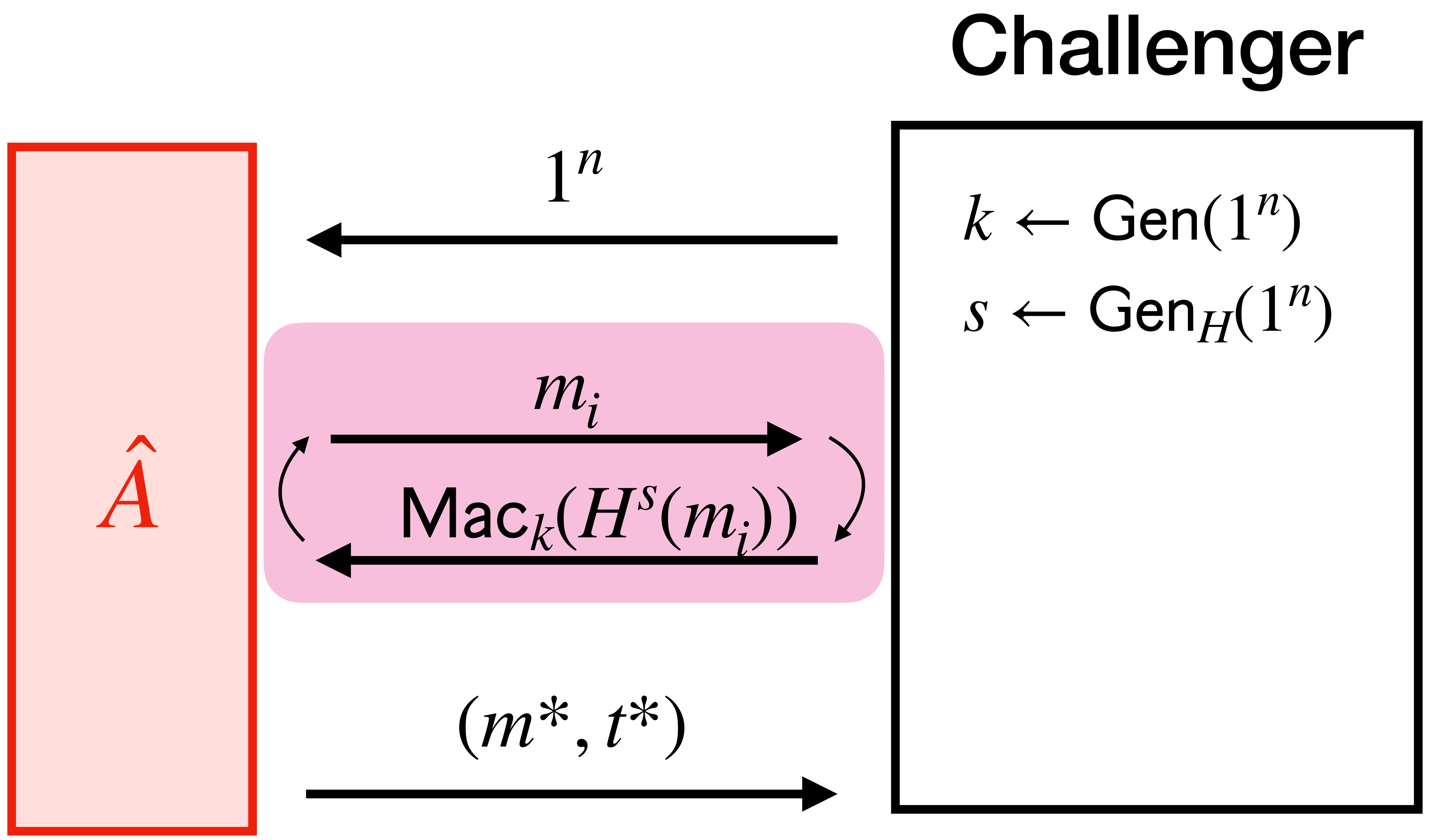
Hash-and-Authenticate

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

$$\begin{aligned}&\Pr[\text{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1]\\ &\quad \leq \Pr[\mathit{collision}]\\ &\qquad + \Pr[\text{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1 \wedge \neg \mathit{collision}]\end{aligned}$$

Claim 1: There exists a negligible function $\nu_1(n)$ such that

$$\Pr[\mathit{collision}] \leq \nu_1(n)$$



Forger wins if $\text{Verify}(H^s(m^*), t^*) = 1$ and $m^* \notin \{m_1, \ldots, m_q\}$

*Law of total probability: $\Pr[A] = \Pr[A \wedge B] + \Pr[A \wedge \neg B] \leq \Pr[B] + \Pr[A \wedge \neg B]$

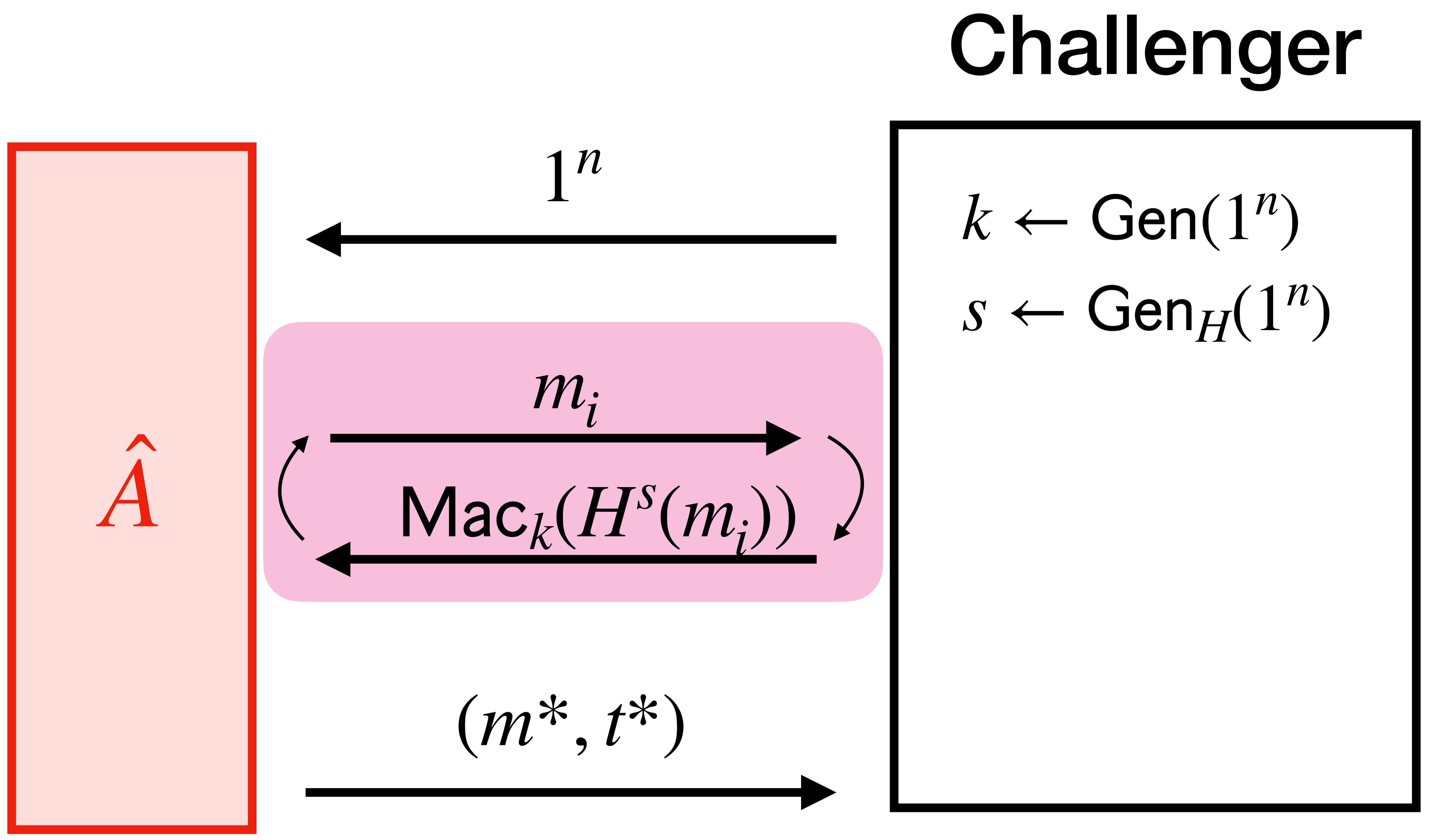
Hash-and-Authenticate

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

$$\Pr[\mathsf{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1]$$
$$\leq \Pr[collision]$$
$$+ \Pr[\mathsf{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1 \wedge \neg collision]$$

Claim 2: There exists a negligible function $\nu_2(n)$ such that

$$\Pr[\mathsf{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1 \wedge \neg collision] \leq \nu_2(n)$$



Challenger
$\hat{A}$
1^n
$k \leftarrow \mathsf{Gen}(1^n)$
$s \leftarrow \mathsf{Gen}_H(1^n)$
m_i
$\mathsf{Mac}_k(H^s(m_i))$
(m^*, t^*)

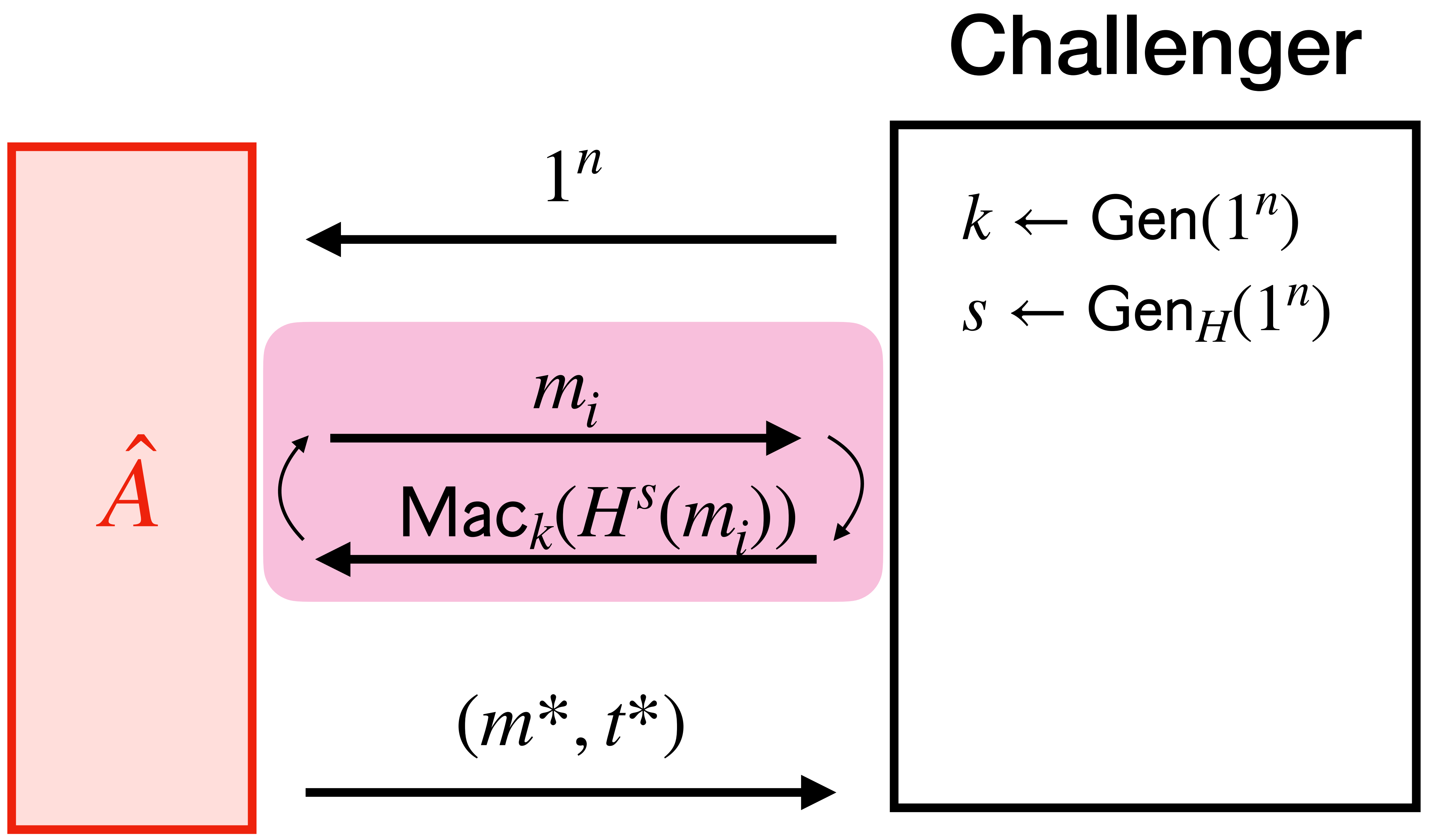
Forger wins if $\mathsf{Verify}(H^s(m^*), t^*) = 1$ and $m^* \notin \{m_1, \ldots, m_q\}$

*Law of total probability: $\Pr[A] = \Pr[A \wedge B] + \Pr[A \wedge \neg B] \leq \Pr[B] + \Pr[A \wedge \neg B]$

Hash-and-Authenticate

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

$$
\begin{aligned}
&\Pr[\mathsf{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1] \\
&\quad \leq \Pr[collision] \\
&\quad\quad + \Pr[\mathsf{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1 \wedge \neg collision] \\
&\quad \leq \nu_1(n) + \nu_2(n)
\end{aligned}
$$



Challenger
$\hat{A}$
1^n
$k \leftarrow \mathsf{Gen}(1^n)$
$s \leftarrow \mathsf{Gen}_H(1^n)$
m_i
$\mathsf{Mac}_k(H^s(m_i))$
(m^*, t^*)

Forger wins if $\mathsf{Verify}(H^s(m^*), t^*) = 1$ and $m^* \notin \{m_1, \ldots, m_q\}$

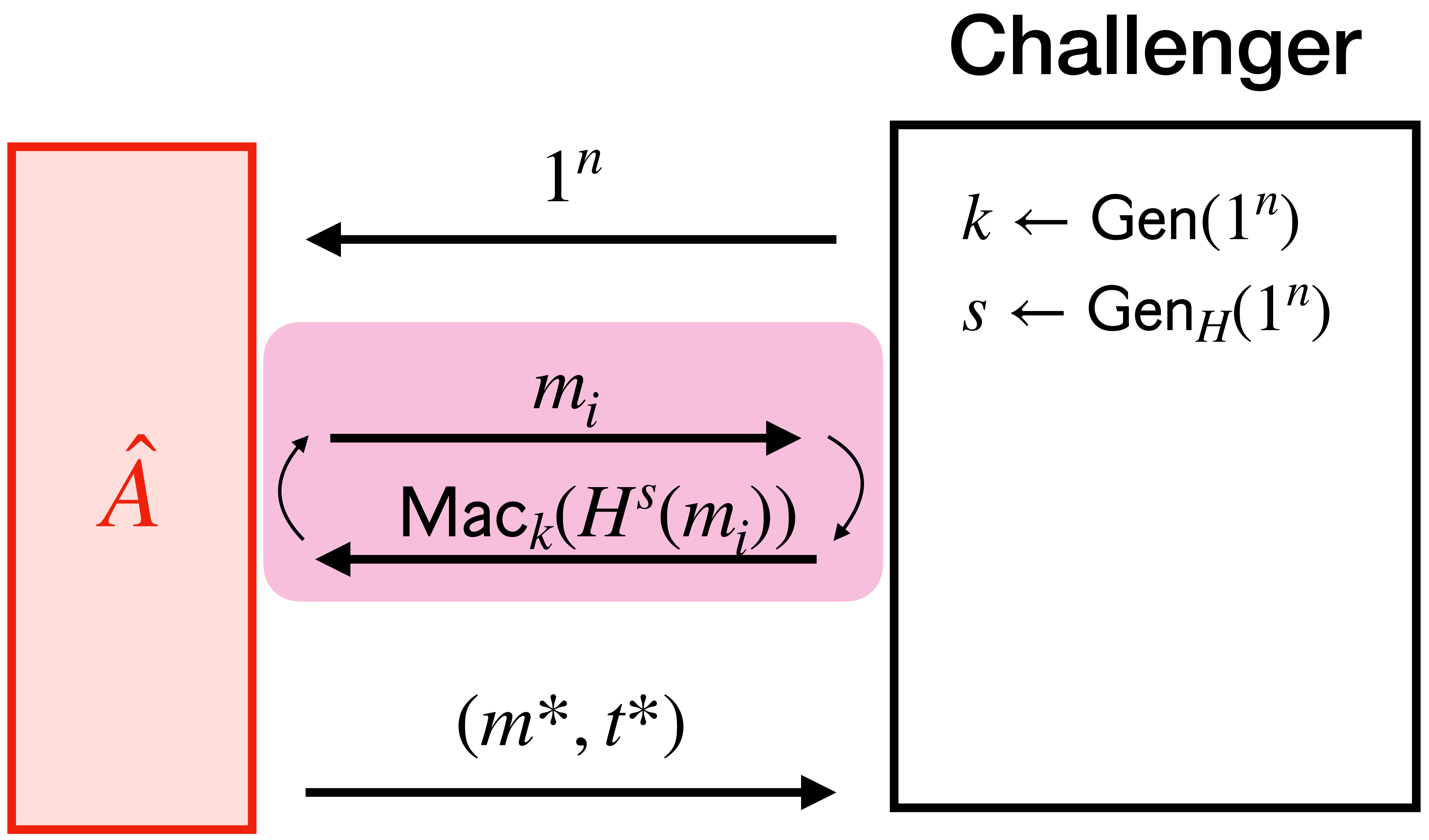
*Law of total probability: $\Pr[A] = \Pr[A \wedge B] + \Pr[A \wedge \neg B] \leq \Pr[B] + \Pr[A \wedge \neg B]$

Hash-and-Authenticate

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

$$\begin{aligned}
&\Pr[\mathsf{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1] \\
&\quad \leq \Pr[\mathit{collision}] \\
&\quad\quad + \Pr[\mathsf{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1 \wedge \neg \mathit{collision}] \\
&\quad \leq \nu_1(n) + \nu_2(n)
\end{aligned}$$

Now we just need to prove Claim 1 and 2



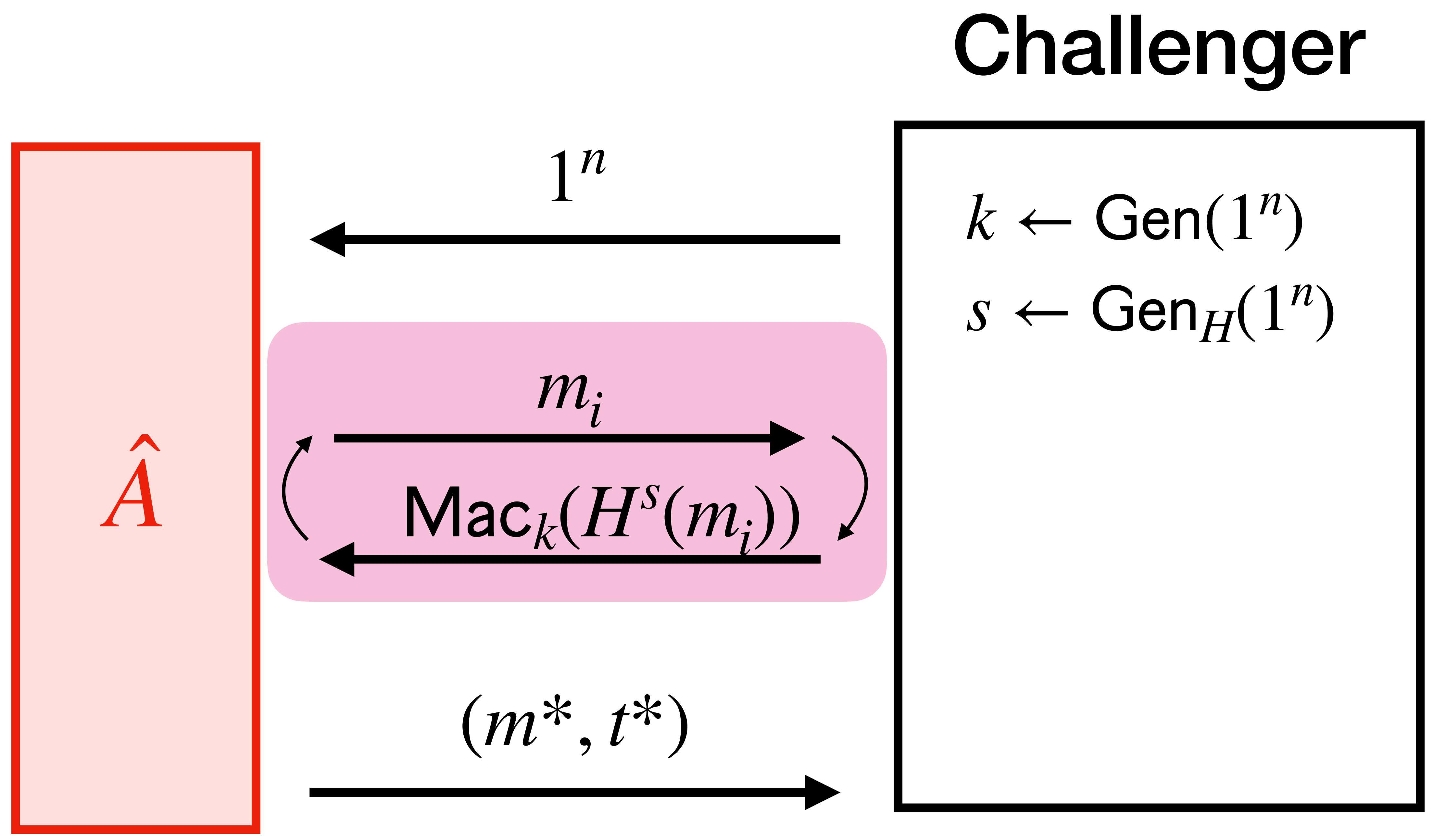
Challenger
$\hat{A}$
1^n
$k \leftarrow \mathsf{Gen}(1^n)$
$s \leftarrow \mathsf{Gen}_H(1^n)$
m_i
$\mathsf{Mac}_k(H^s(m_i))$
(m^*, t^*)

Forger wins if $\mathsf{Verify}(H^s(m^*), t^*) = 1$ and $m^* \notin \{m_1, \ldots, m_q\}$

*Law of total probability: $\Pr[A] = \Pr[A \wedge B] + \Pr[A \wedge \neg B] \leq \Pr[B] + \Pr[A \wedge \neg B]$

Proof of Claim 1

Claim 1: There exists a negligible function $\nu_1(n)$ such that $\Pr[collision] \leq \nu_1(n)$


Challenger
$k \leftarrow \mathsf{Gen}(1^n)$
$s \leftarrow \mathsf{Gen}_H(1^n)$
$\hat{A}$
1^n
m_i
$\mathsf{Mac}_k(H^s(m_i))$
(m^*, t^*)

Forger wins if $\mathsf{Verify}(H^s(m^*), t^*) = 1$
and $m^* \notin \{m_1, \ldots, m_q\}$

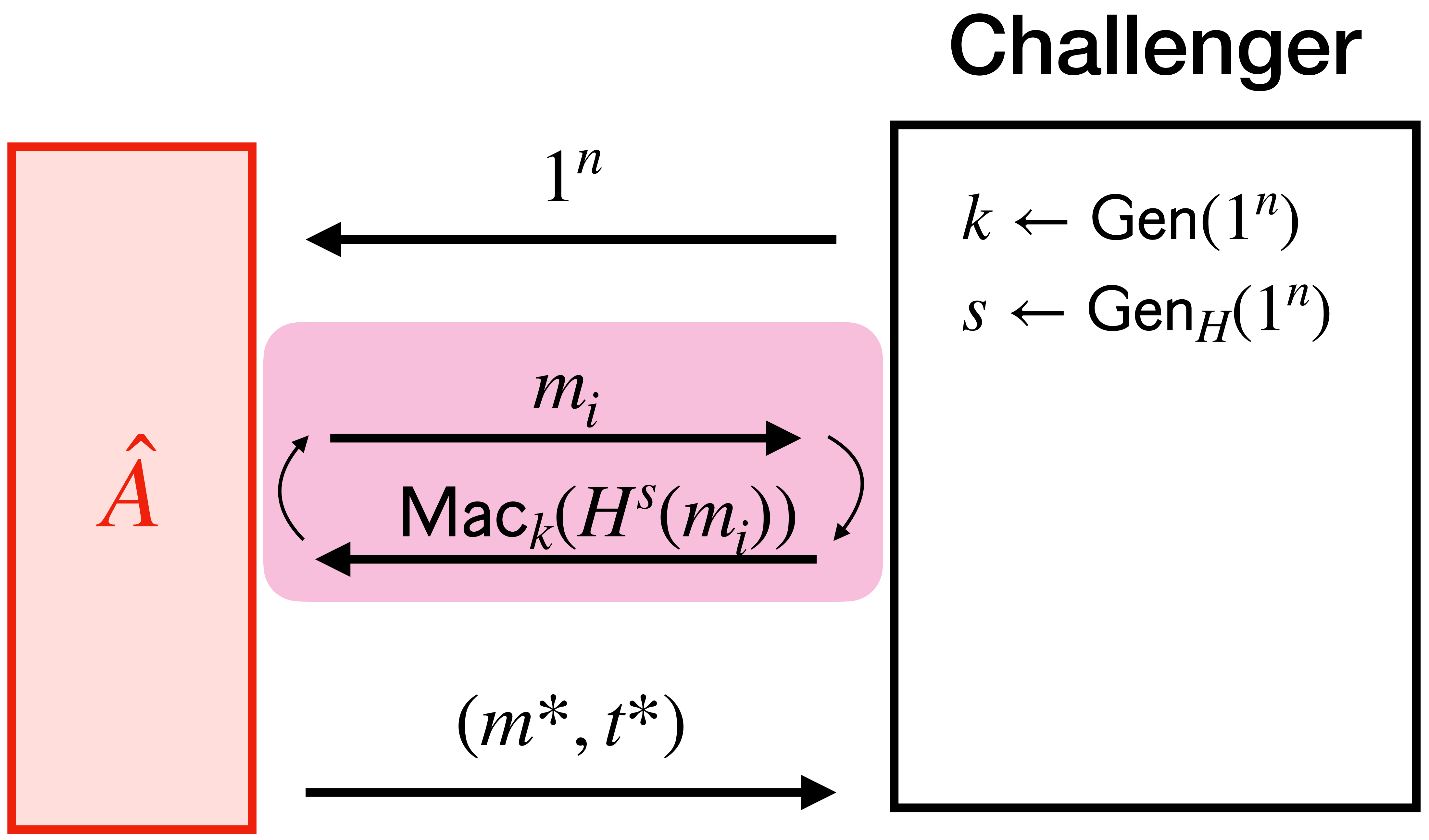
Proof of Claim 1

Claim 1: There exists a negligible function $\nu_1(n)$ such that $\Pr[collision] \leq \nu_1(n)$

Proof of claim 1:

Assume towards contradiction that there exists an adversary $\hat{A}$ and a polynomial $p(n)$ s.t. $\Pr[collision] \geq 1/p(n)$

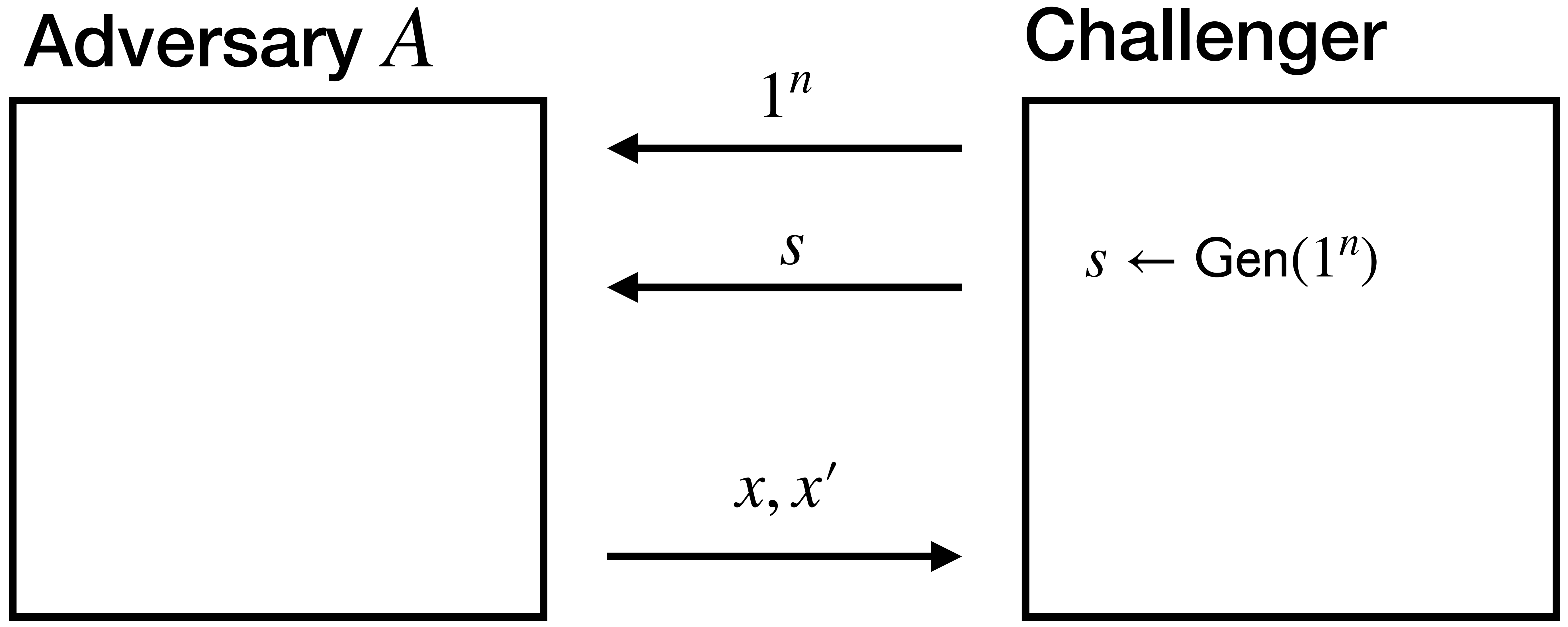
We will use $\hat{A}$ to construct a collision-finder C to break the CRHF security of Φ



Forger wins if $\text{Verify}(H^s(m^*), t^*) = 1$ and $m^* \notin \{m_1, \ldots, m_q\}$

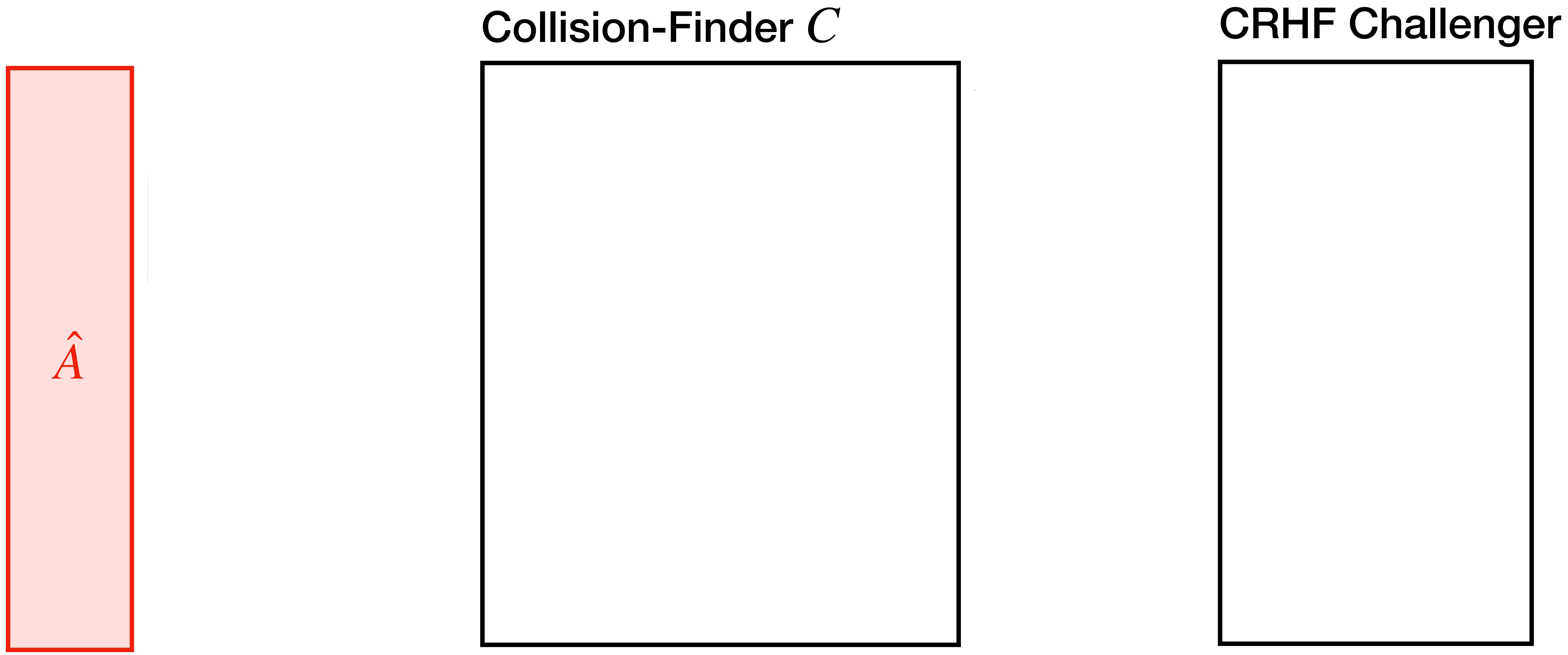
Recall: CRHF Security

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

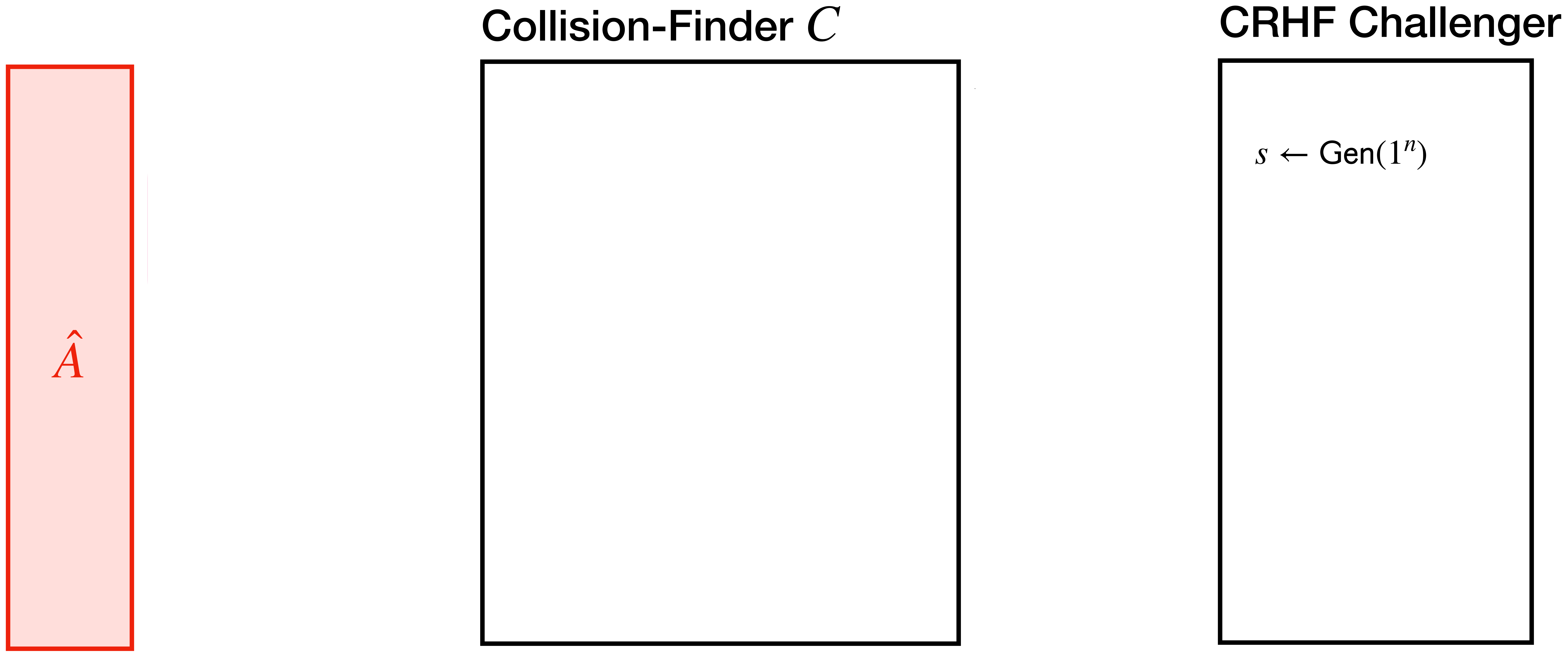


$$\mathsf{HashColl}_{\Phi,A}(n) = \begin{cases} 1 & H^s(x) = H^s(x') \text{ and } x \neq x' \\ 0 & \text{otherwise} \end{cases}$$

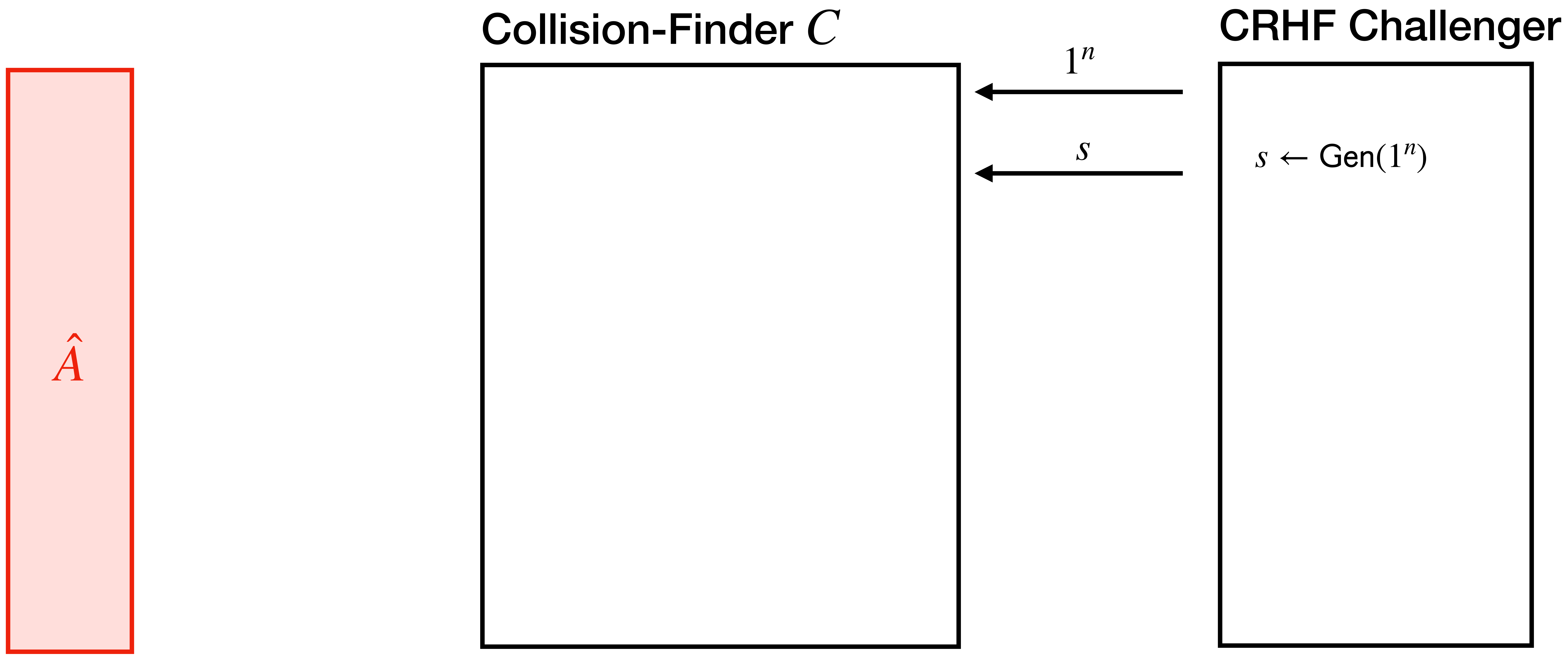
Proof of Claim 1



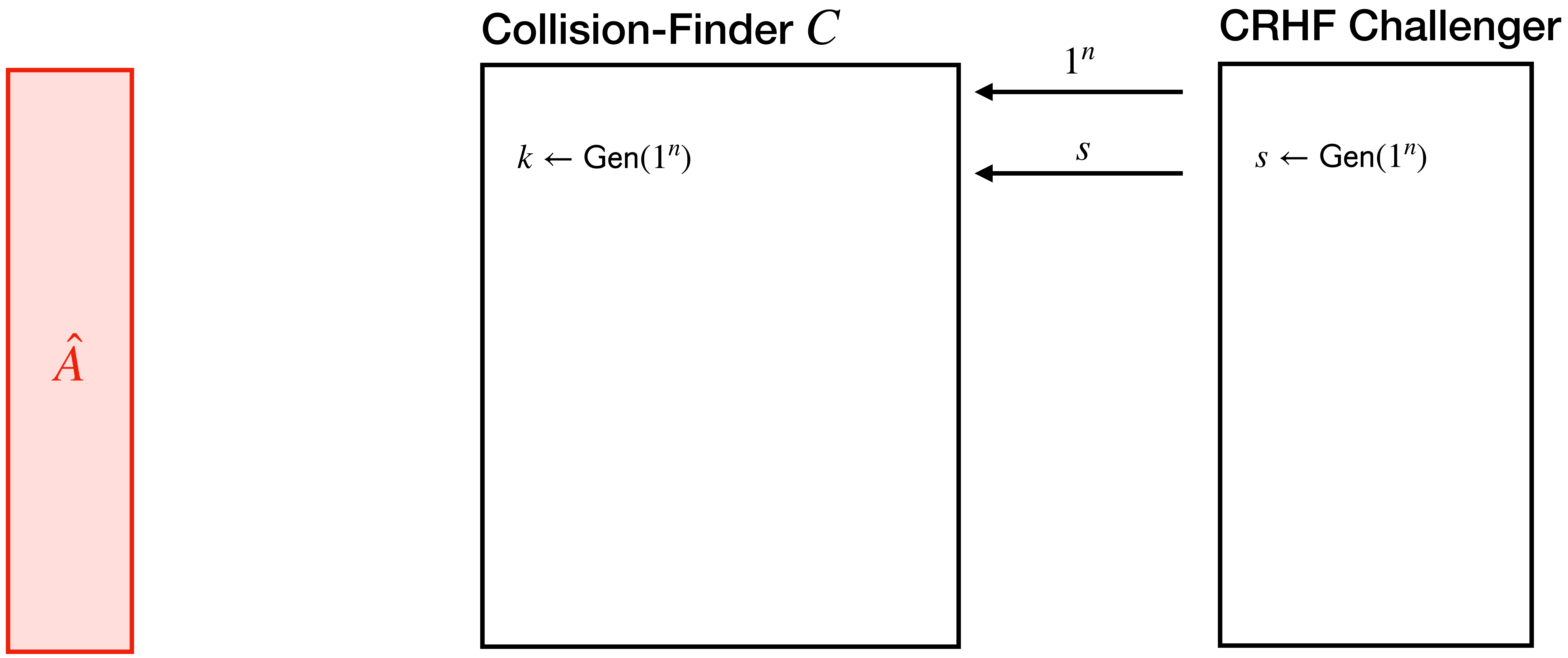
Proof of Claim 1



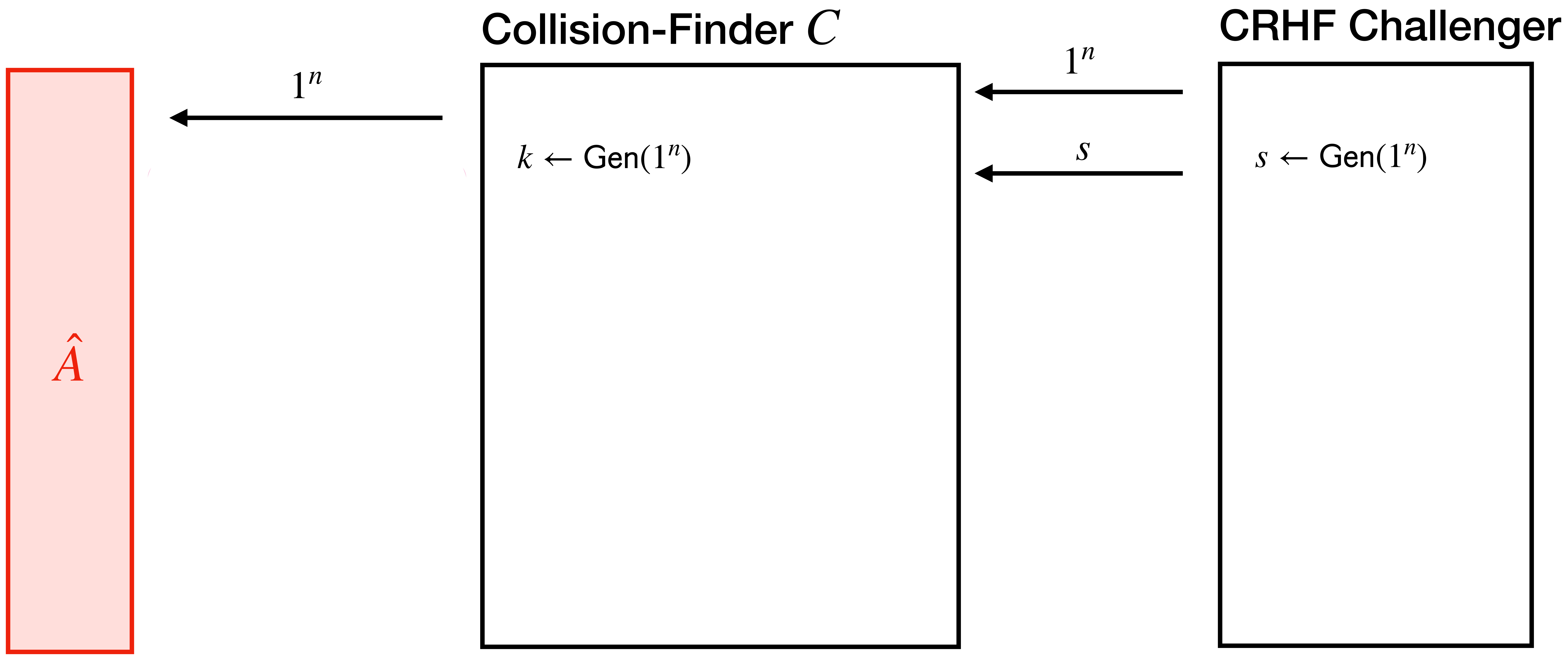
Proof of Claim 1



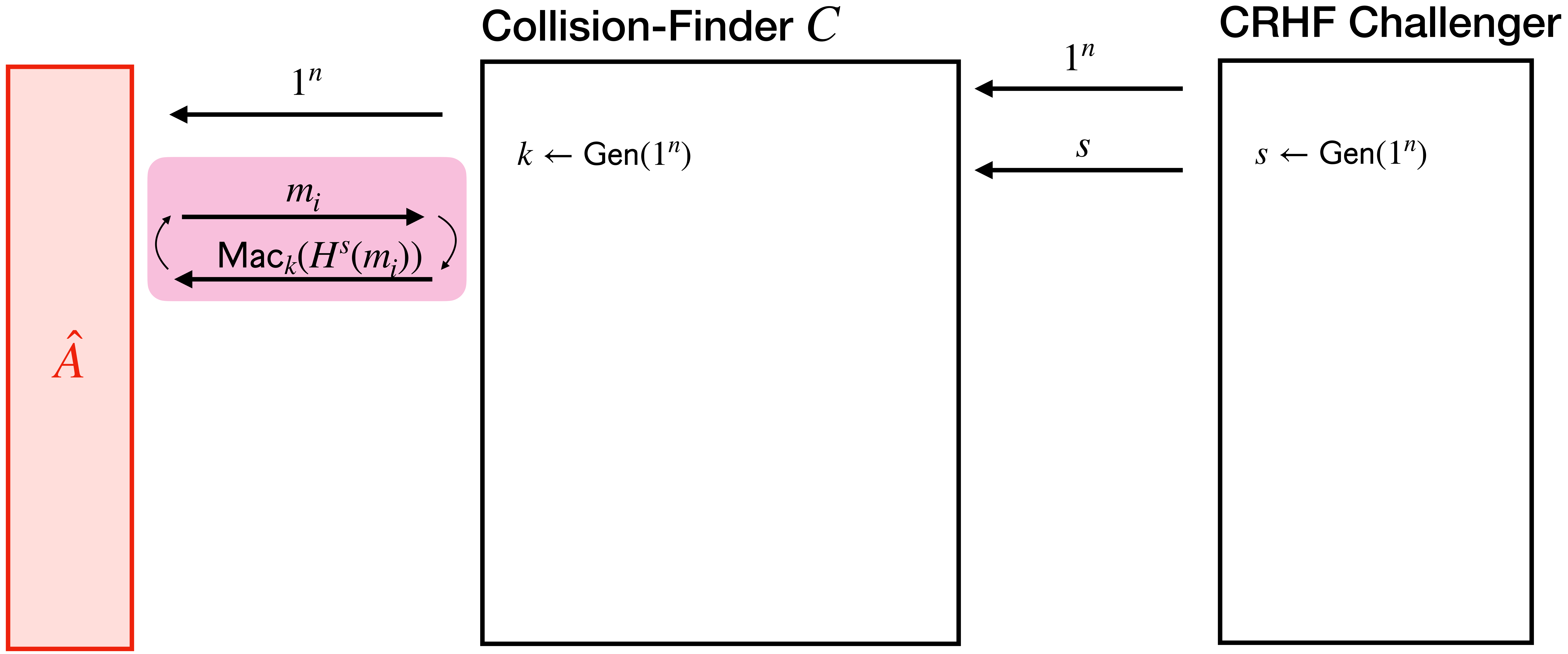
Proof of Claim 1


Collision-Finder C
CRHF Challenger
$\hat{A}$
$k \leftarrow \mathsf{Gen}(1^n)$
1^n
s
$s \leftarrow \mathsf{Gen}(1^n)$

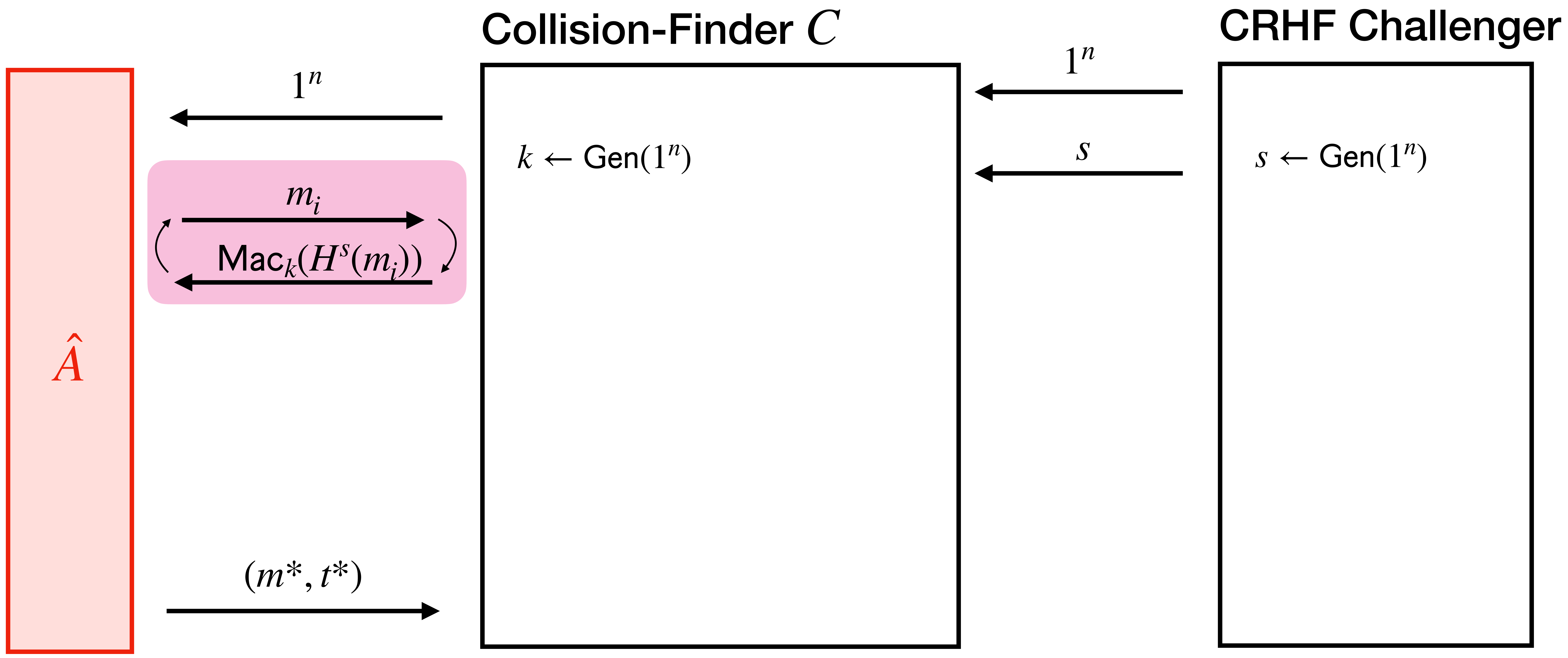
Proof of Claim 1


Collision-Finder C
CRHF Challenger
$\hat{A}$
1^n
1^n
s
$k \leftarrow \mathsf{Gen}(1^n)$
$s \leftarrow \mathsf{Gen}(1^n)$

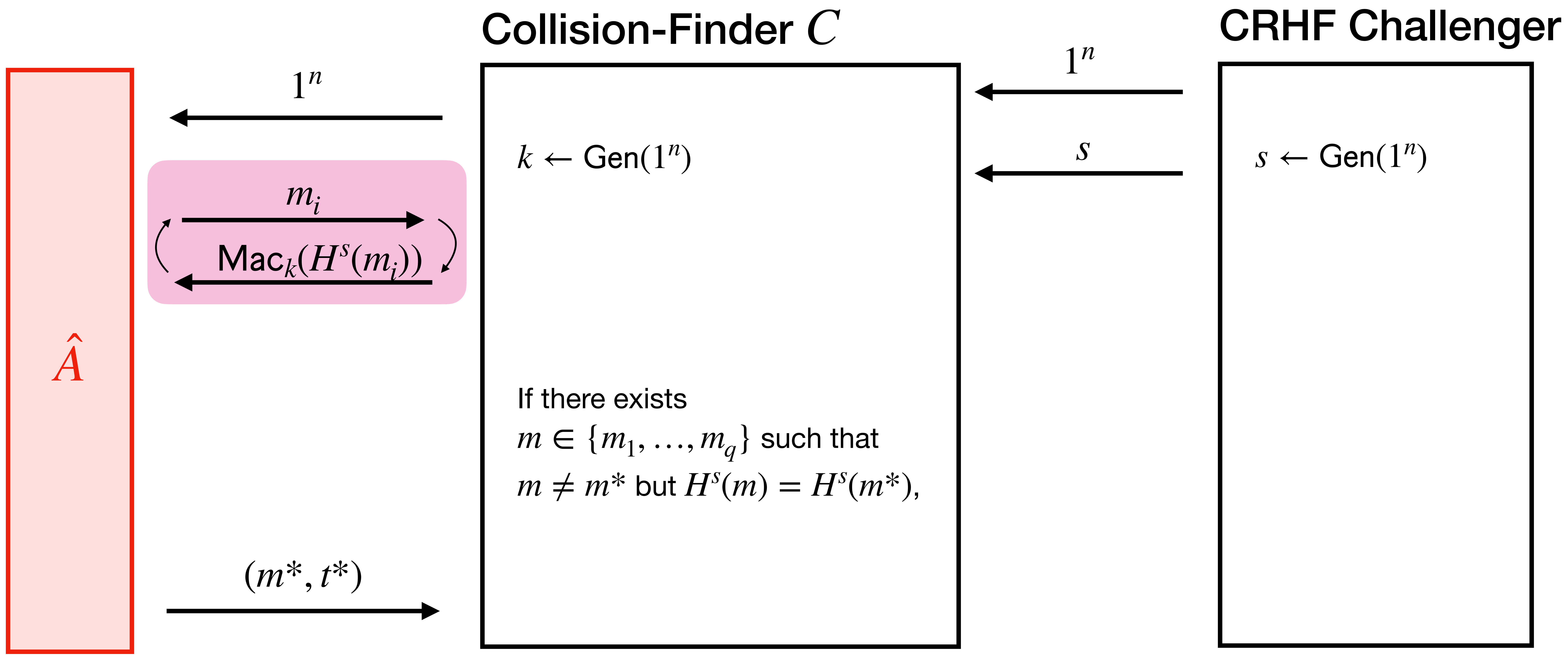
Proof of Claim 1



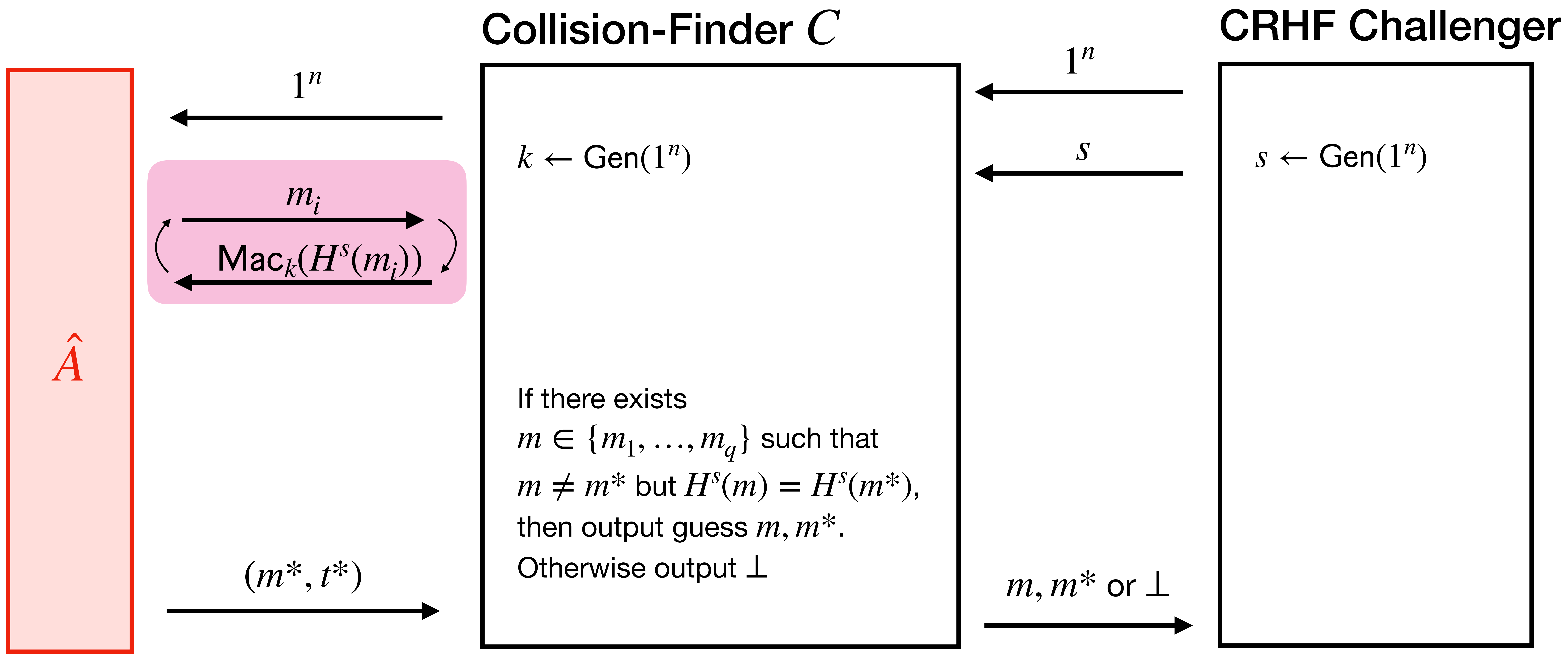
Proof of Claim 1



Proof of Claim 1


Collision-Finder C
CRHF Challenger
$\hat{A}$
1^n
m_i
$\mathsf{Mac}_k(H^s(m_i))$
(m^*, t^*)
$k \leftarrow \mathsf{Gen}(1^n)$
If there exists
$m \in \{m_1, \ldots, m_q\}$ such that
$m \neq m^*$ but $H^s(m) = H^s(m^*)$,
1^n
s
$s \leftarrow \mathsf{Gen}(1^n)$

Proof of Claim 1

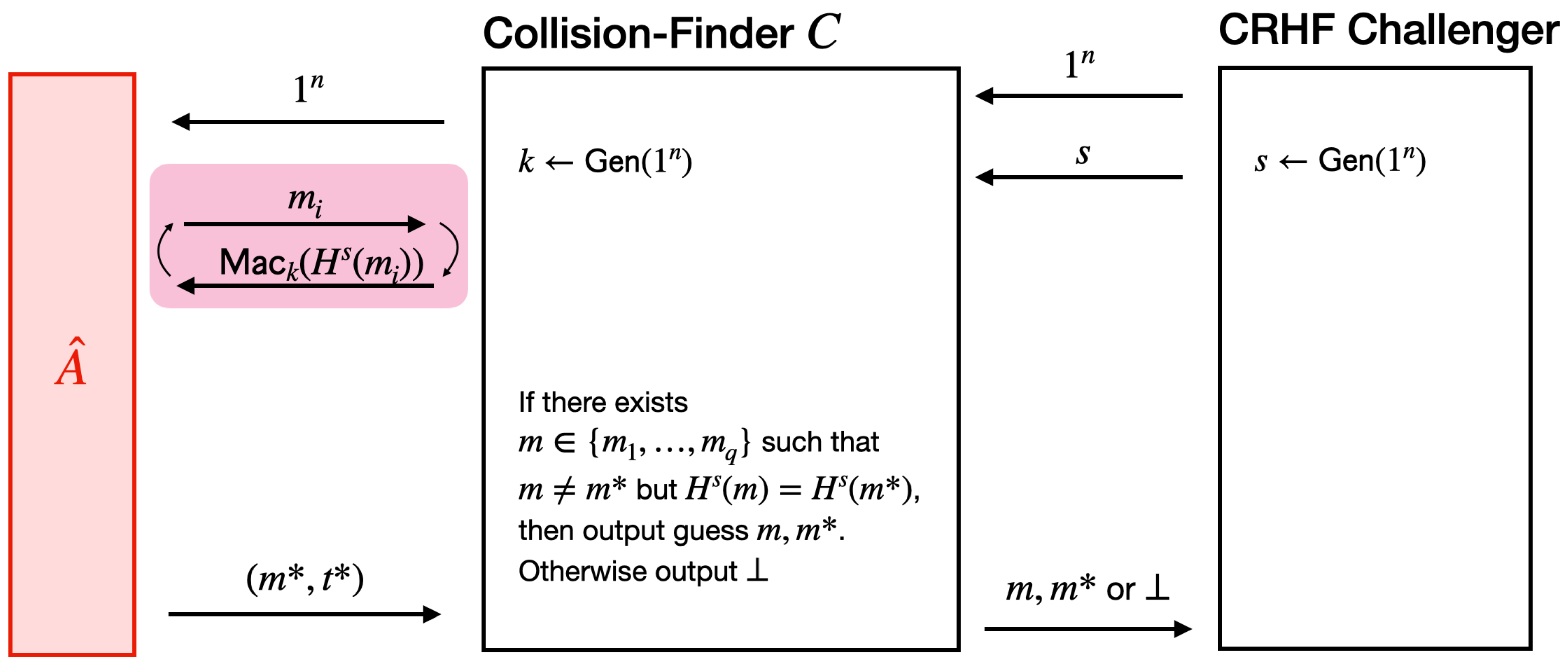


Collision-Finder C
CRHF Challenger
$\hat{A}$
1^n
m_i
$\text{Mac}_k(H^s(m_i))$
(m^*, t^*)
$k \leftarrow \text{Gen}(1^n)$
If there exists
$m \in \{m_1, \ldots, m_q\}$ such that
$m \neq m^*$ but $H^s(m) = H^s(m^*)$,
then output guess m, m^*.
Otherwise output $\perp$
1^n
s
m, m^* or $\perp$
$s \leftarrow \text{Gen}(1^n)$

Proof of Claim 1

Assume towards contradiction that there exists an adversary $\hat{A}$ and a polynomial $p(n)$ s.t. $\Pr[collision] \geq 1/p(n)$

$\Pr[\text{HashColl}_{\Phi,C} = 1]$

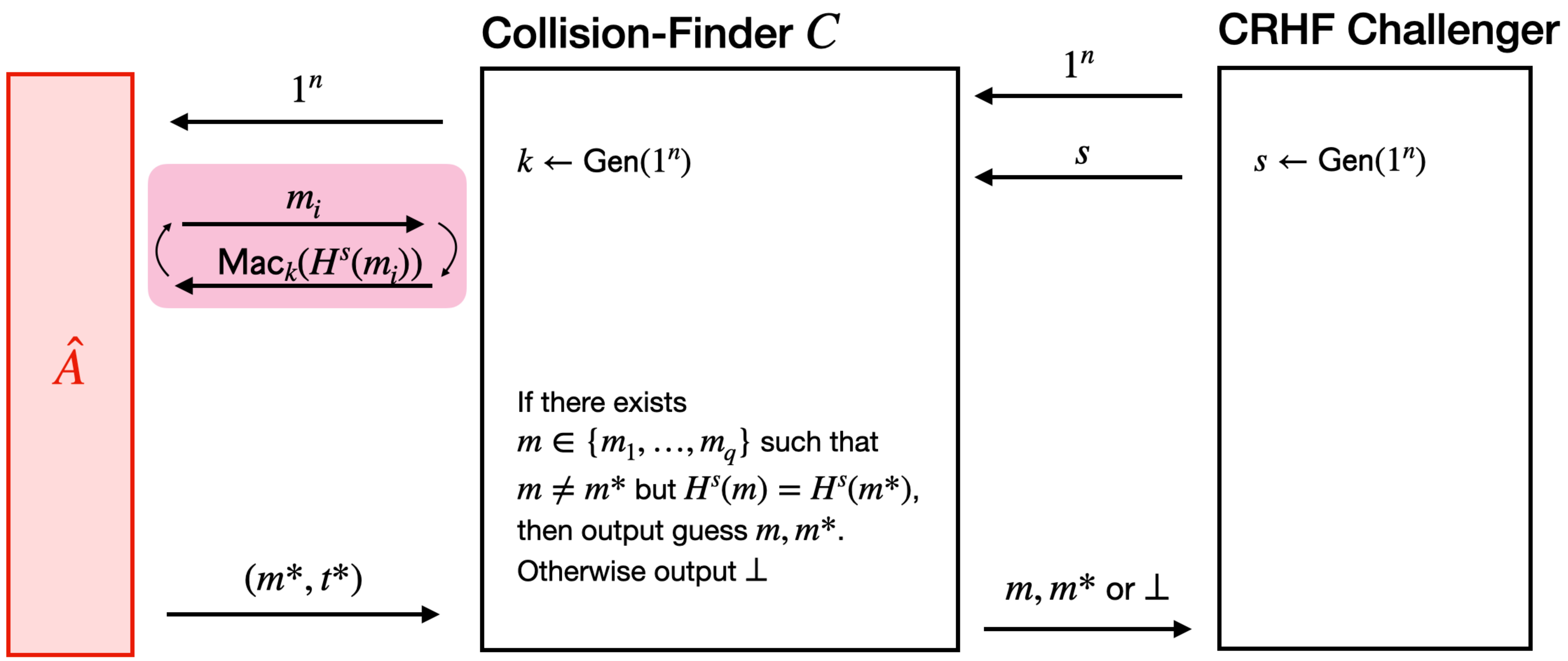


If there exists $m \in \{m_1, \ldots, m_q\}$ such that $m \neq m^*$ but $H^s(m) = H^s(m^*)$, then output guess m, m^*. Otherwise output $\perp$

Proof of Claim 1

Assume towards contradiction that there exists an adversary $\hat{A}$ and a polynomial $p(n)$ s.t. $\Pr[collision] \geq 1/p(n)$

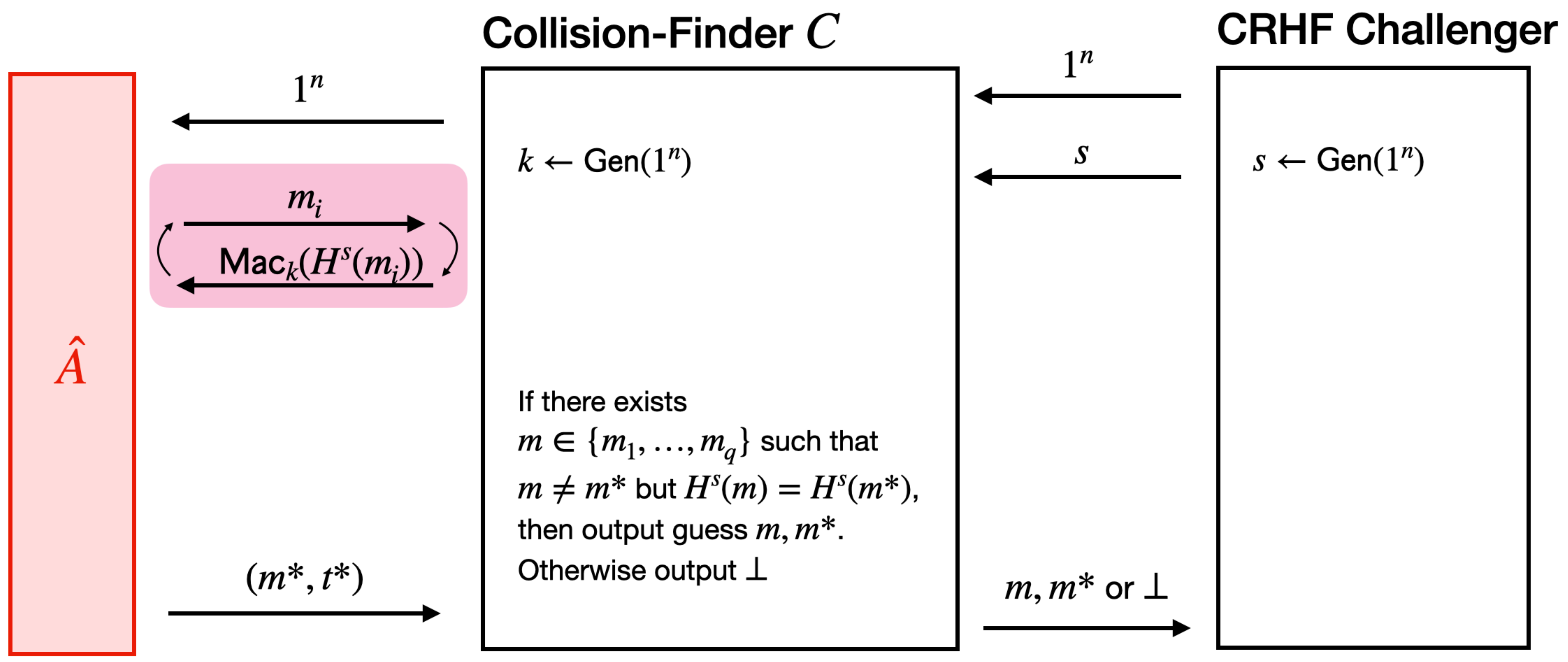
$$\Pr[\text{HashColl}_{\Phi,C} = 1]$$
$$= \Pr[collision]$$


Collision-Finder C
CRHF Challenger
$\hat{A}$
1^n
m_i
$\text{Mac}_k(H^s(m_i))$
(m^*, t^*)
$k \leftarrow \text{Gen}(1^n)$
If there exists $m \in \{m_1, \ldots, m_q\}$ such that $m \neq m^*$ but $H^s(m) = H^s(m^*)$, then output guess m, m^*. Otherwise output $\perp$
1^n
s
m, m^* or $\perp$
$s \leftarrow \text{Gen}(1^n)$

Proof of Claim 1

Assume towards contradiction that there exists an adversary $\hat{A}$ and a polynomial $p(n)$ s.t. $\Pr[collision] \geq 1/p(n)$

$$\Pr[\text{HashColl}_{\Phi,C} = 1]$$
$$= \Pr[collision]$$
$$\geq 1/p(n)$$



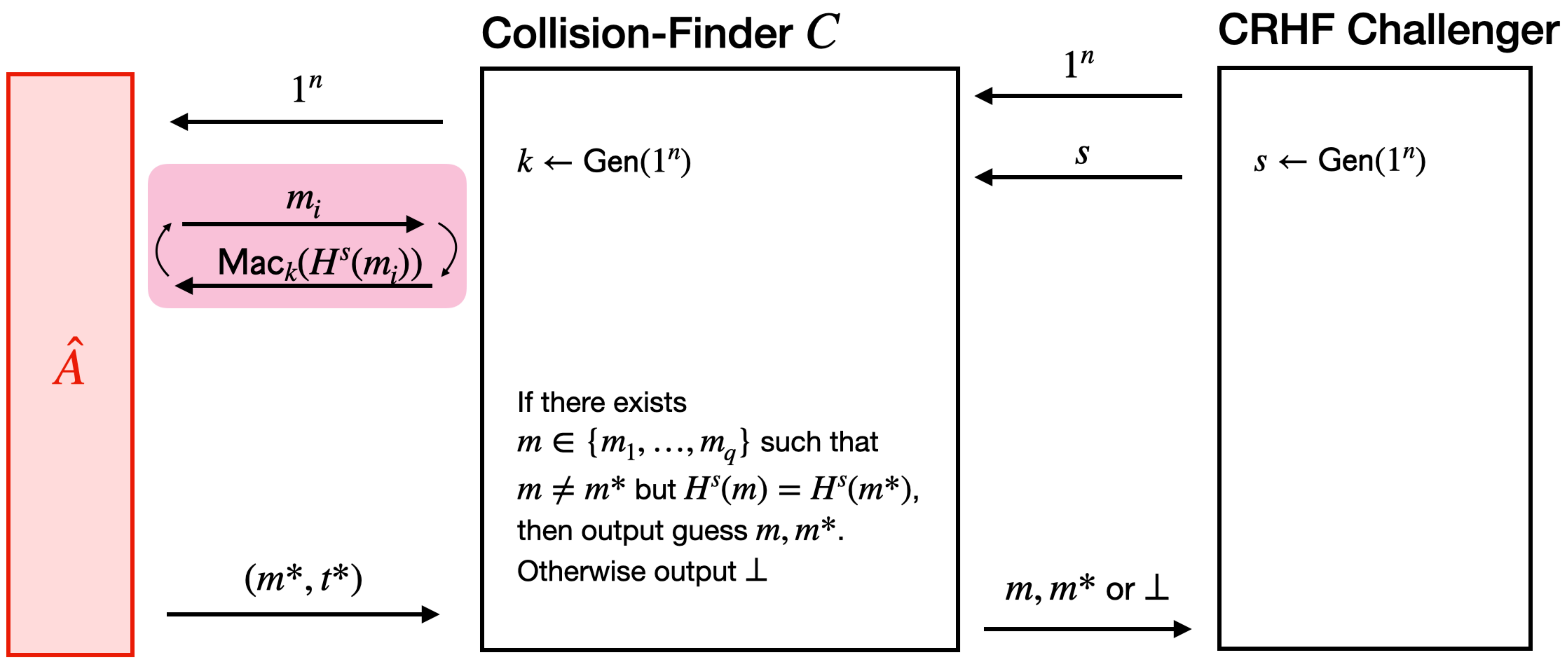
Collision-Finder C
CRHF Challenger
$\hat{A}$
1^n
m_i
$\text{Mac}_k(H^s(m_i))$
(m^*, t^*)
$k \leftarrow \text{Gen}(1^n)$
If there exists $m \in \{m_1, \ldots, m_q\}$ such that $m \neq m^*$ but $H^s(m) = H^s(m^*)$, then output guess m, m^*. Otherwise output $\perp$
1^n
s
m, m^* or $\perp$
$s \leftarrow \text{Gen}(1^n)$

Proof of Claim 1

Assume towards contradiction that there exists an adversary $\hat{A}$ and a polynomial $p(n)$ s.t. $\Pr[collision] \geq 1/p(n)$

$$\Pr[\text{HashColl}_{\Phi,C} = 1]$$
$$= \Pr[collision]$$
$$\geq 1/p(n)$$

Contradiction!



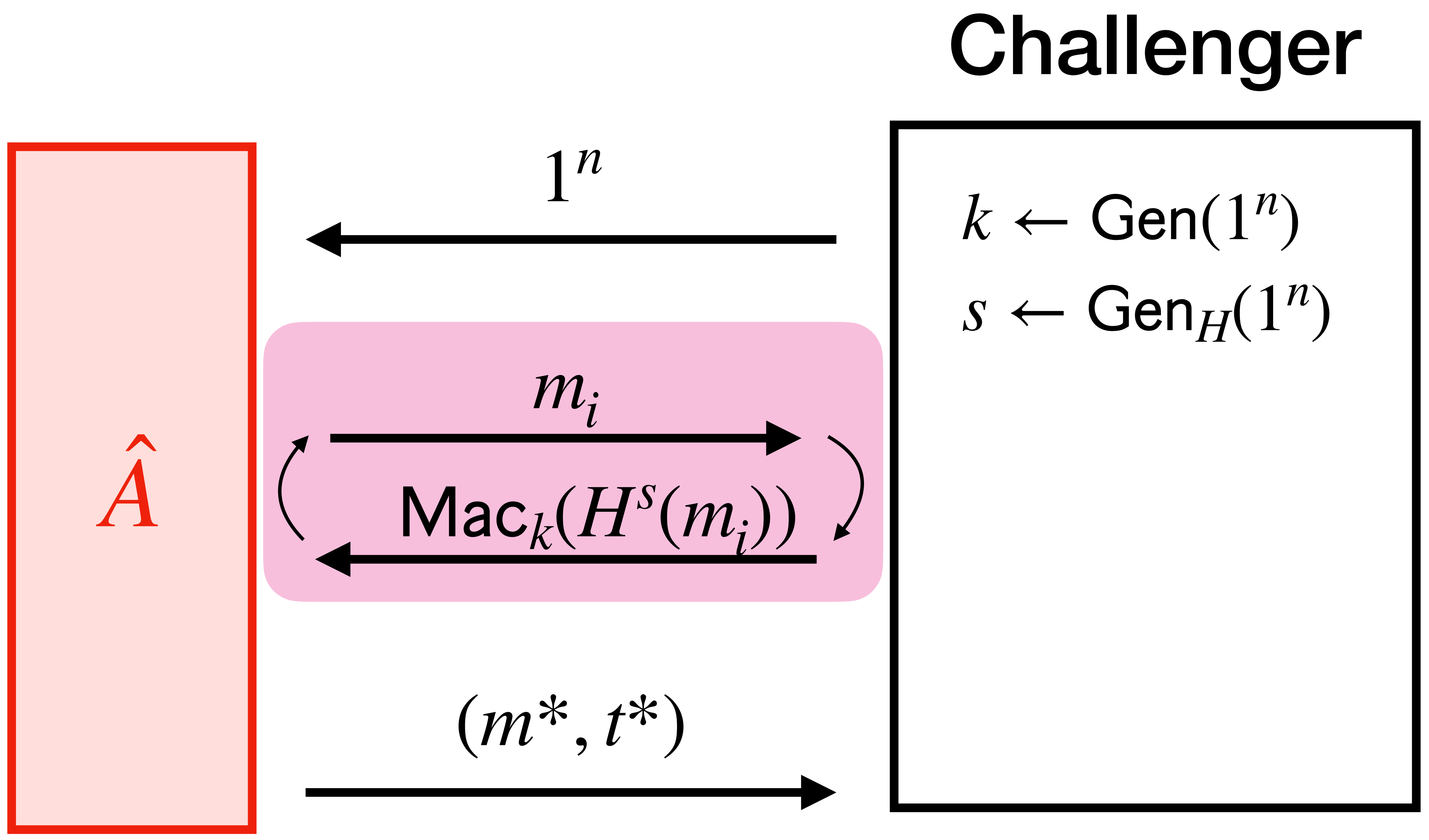
If there exists $m \in \{m_1, \ldots, m_q\}$ such that $m \neq m^*$ but $H^s(m) = H^s(m^*)$, then output guess m, m^*. Otherwise output $\perp$

Hash-and-Authenticate

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

$$\begin{aligned}
&\Pr[\text{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1] \\
&\quad \leq \Pr[\mathit{collision}] \\
&\quad\quad + \Pr[\text{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1 \wedge \neg \mathit{collision}] \\
&\quad \leq \nu_1(n) + \nu_2(n)
\end{aligned}$$

Now we just need to prove Claim 1 and 2



Forger wins if $\text{Verify}(H^s(m^*), t^*) = 1$ and $m^* \notin \{m_1, \ldots, m_q\}$

*Law of total probability: $\Pr[A] = \Pr[A \wedge B] + \Pr[A \wedge \neg B] \leq \Pr[B] + \Pr[A \wedge \neg B]$

Proof of Claim 2

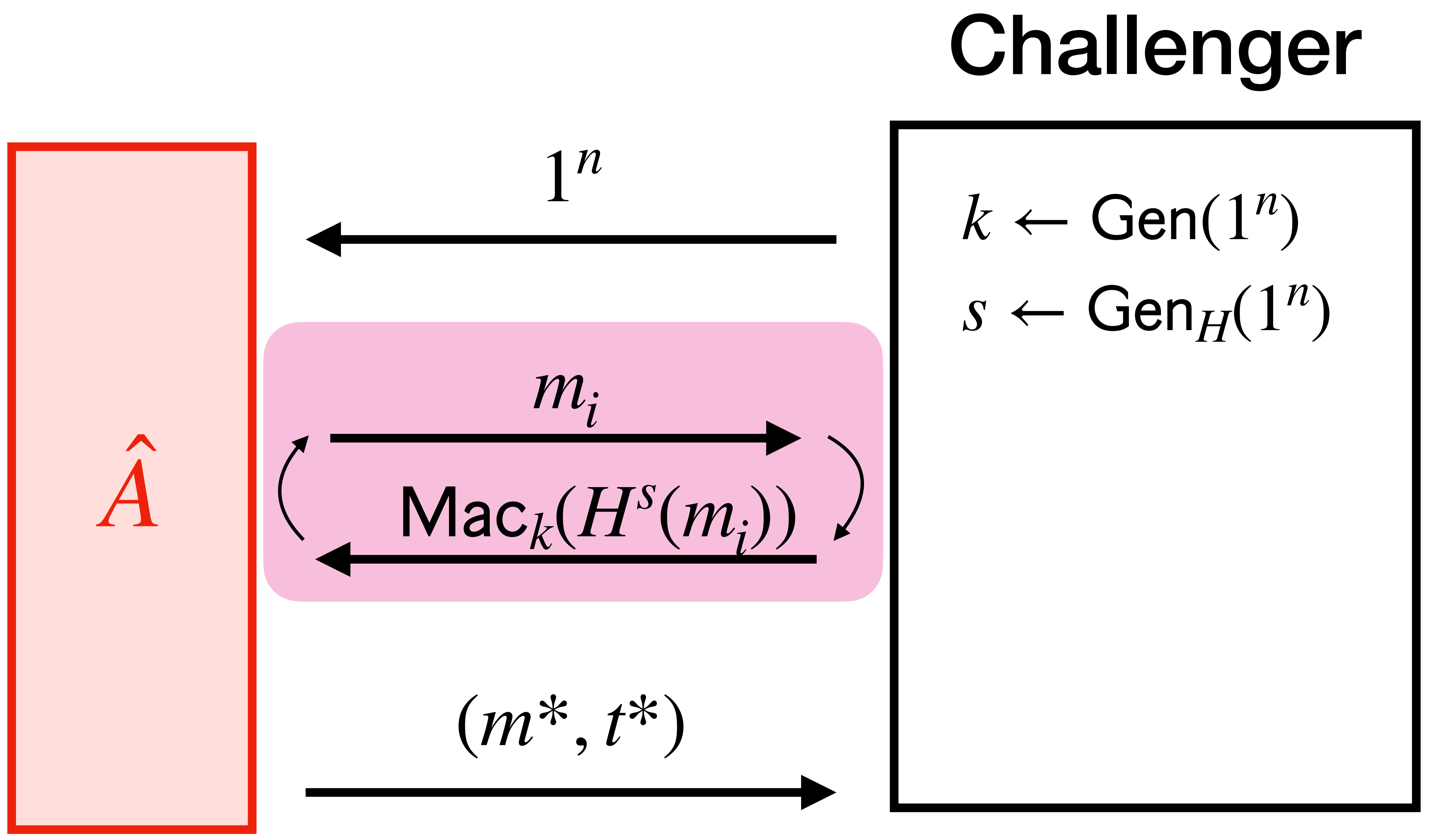
Claim 2: There exists a negligible function $\nu_2(n)$ such that
$\Pr[\mathsf{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1 \wedge \neg collision] \leq \nu_2(n)$

Proof of claim 2:

Assume towards contradiction that there exists an $\hat{A}$ and polynomial $p(n)$ s.t.

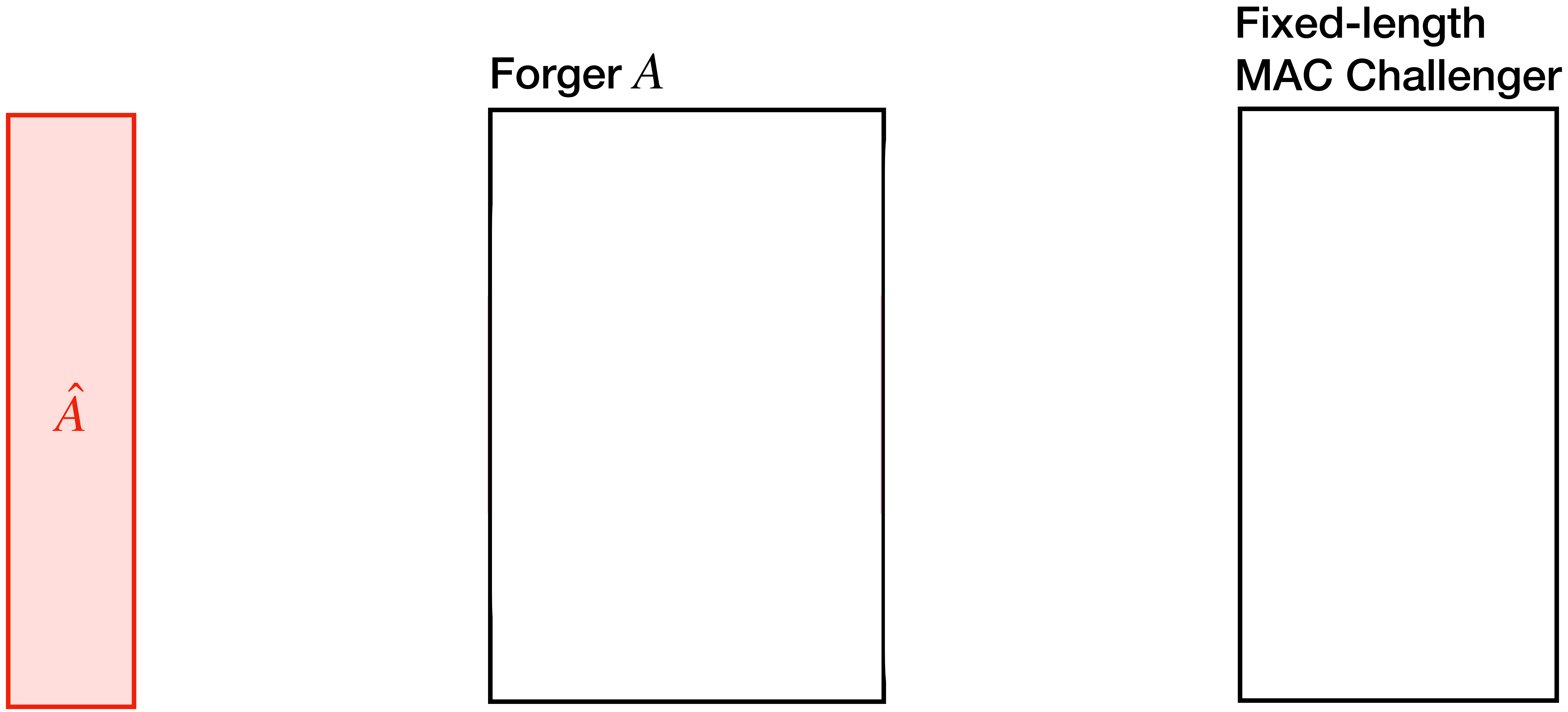
$\Pr[\mathsf{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1 \wedge \neg collision] \geq 1/p(n)$

We will use $\hat{A}$ to construct a forger A to break the MAC security of Π

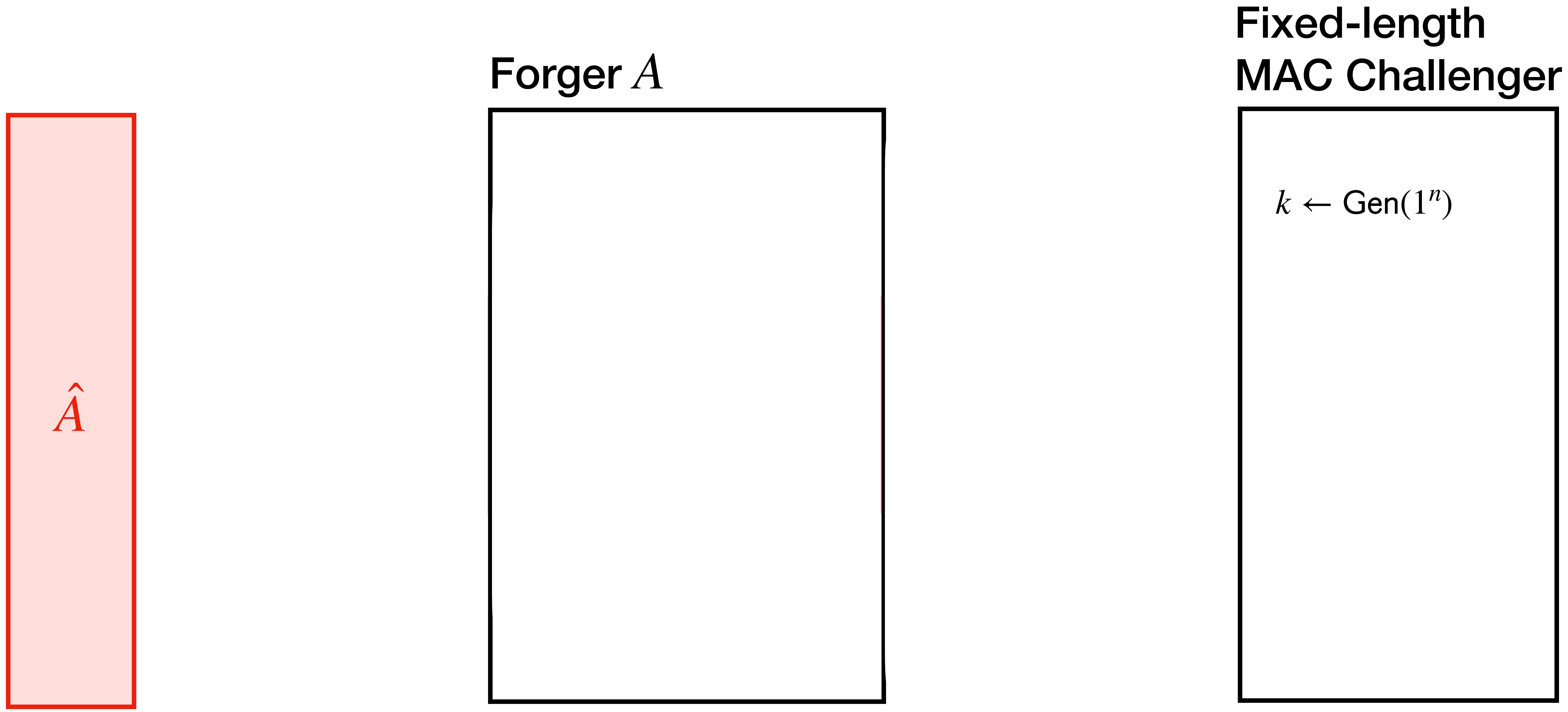

Challenger
$k \leftarrow \mathsf{Gen}(1^n)$
$s \leftarrow \mathsf{Gen}_H(1^n)$
$\hat{A}$
1^n
m_i
$\mathsf{Mac}_k(H^s(m_i))$
(m^*, t^*)

Forger wins if $\mathsf{Verify}(H^s(m^*), t^*) = 1$ and $m^* \notin \{m_1, \ldots, m_q\}$

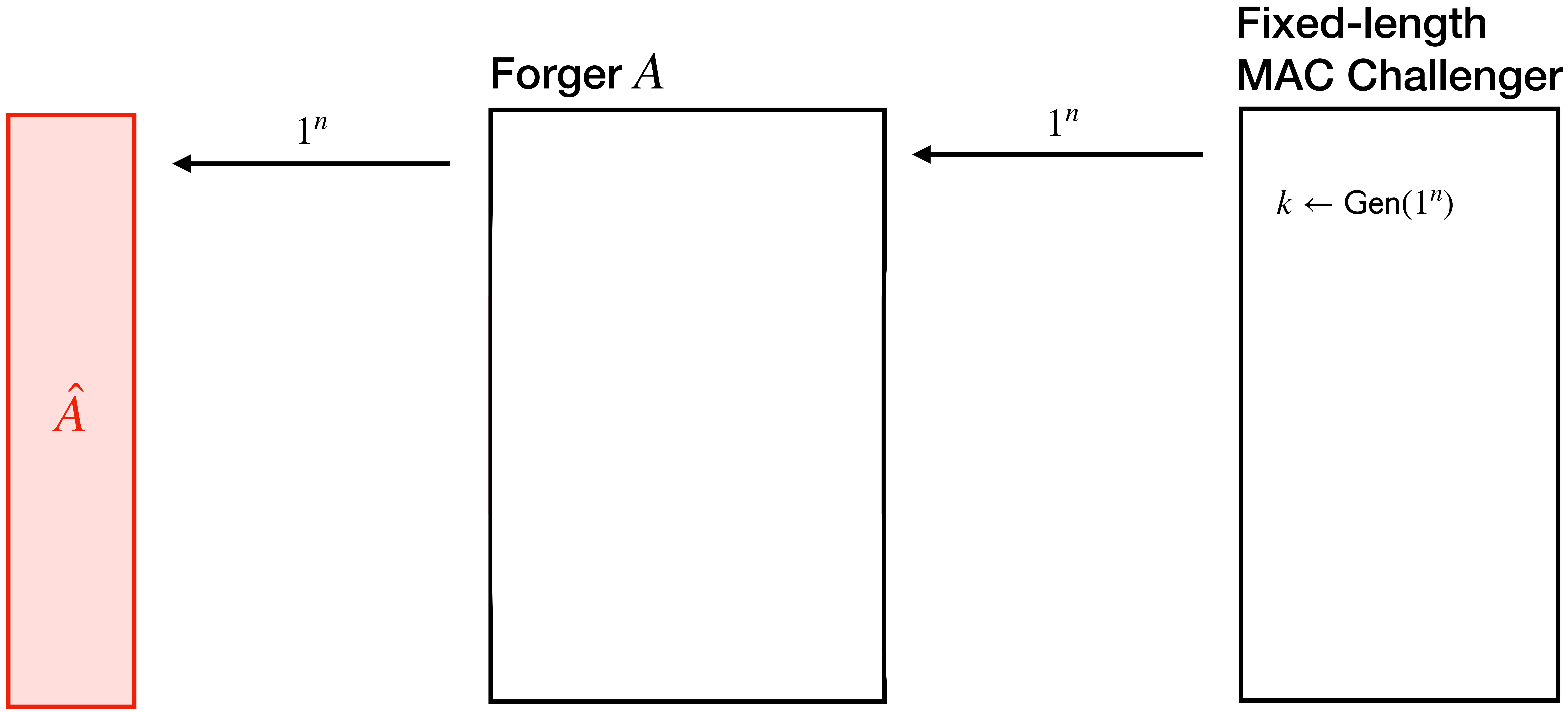
Proof of Claim 2



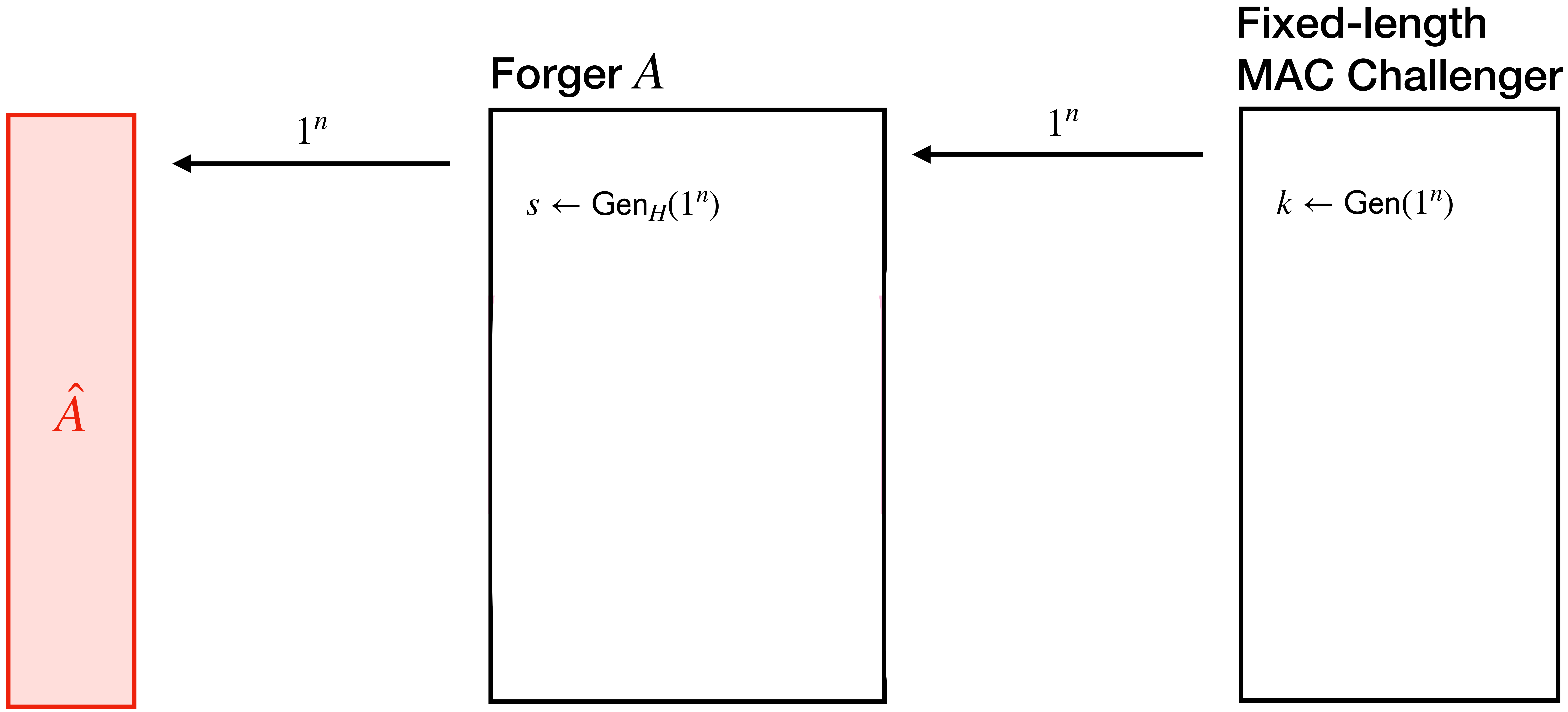
Proof of Claim 2



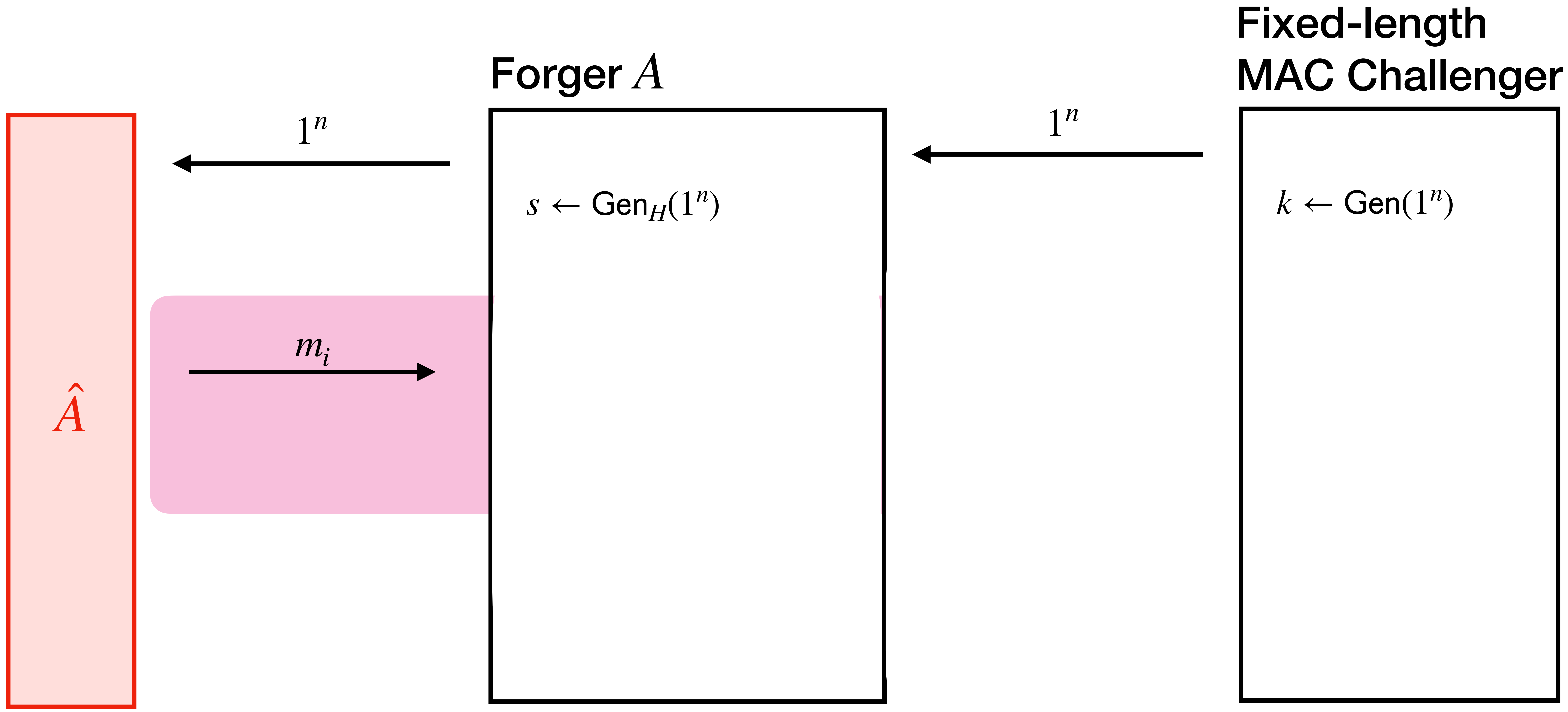
Proof of Claim 2



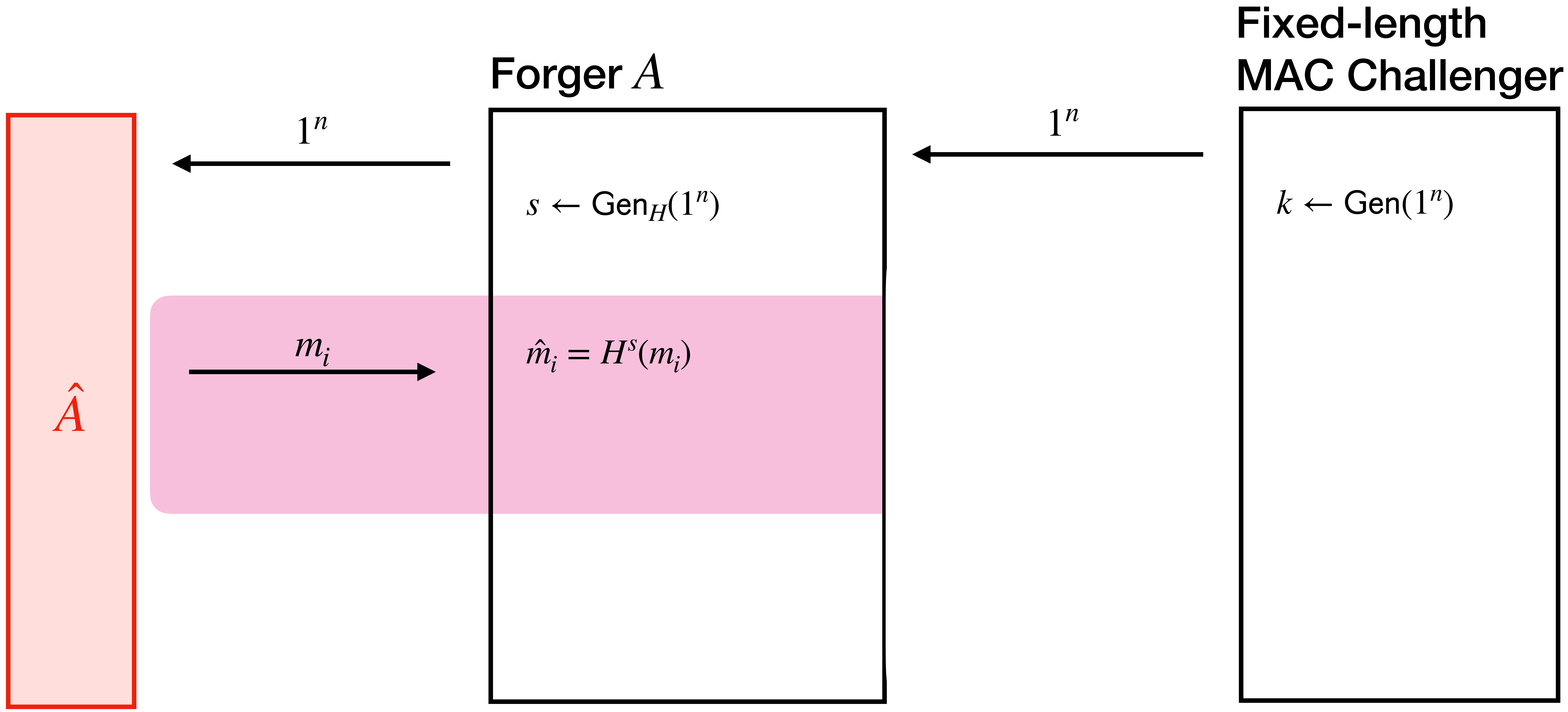
Proof of Claim 2


Forger A
Fixed-length MAC Challenger
$\hat{A}$
1^n
1^n
$s \leftarrow \mathsf{Gen}_H(1^n)$
$k \leftarrow \mathsf{Gen}(1^n)$

Proof of Claim 2

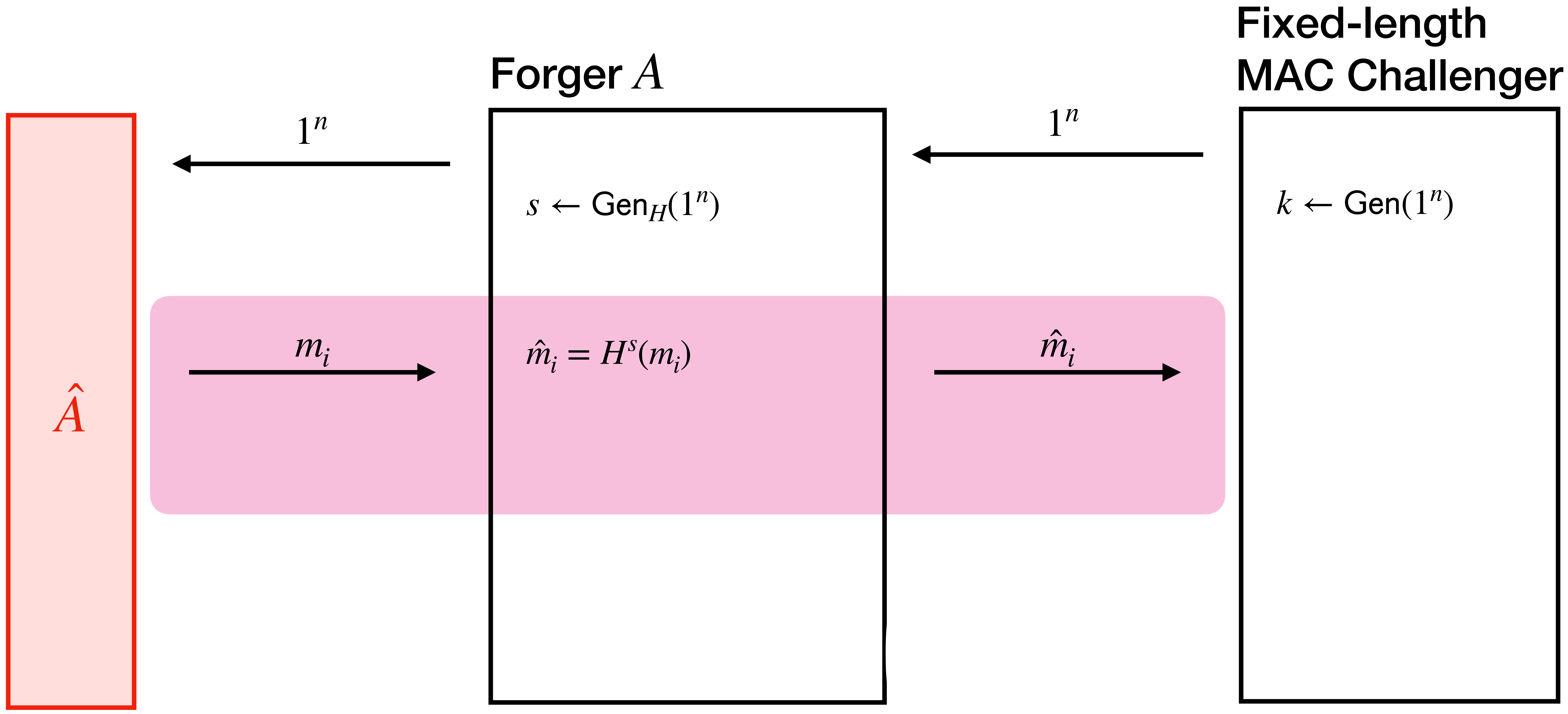

Forger A
Fixed-length MAC Challenger
1^n
1^n
$s \leftarrow \mathsf{Gen}_H(1^n)$
$k \leftarrow \mathsf{Gen}(1^n)$
m_i
$\hat{A}$

Proof of Claim 2

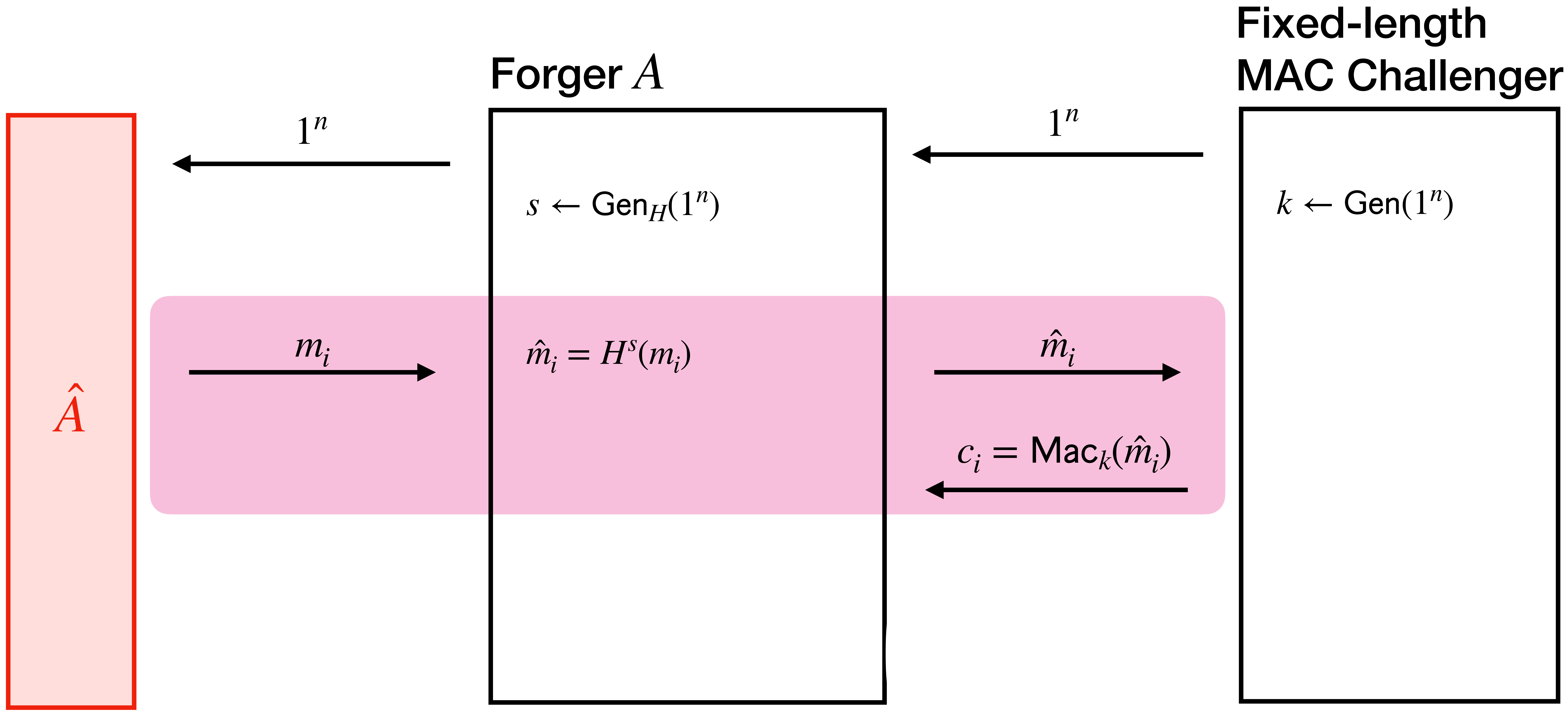


Forger A
Fixed-length MAC Challenger
1^n
1^n
$s \leftarrow \mathsf{Gen}_H(1^n)$
$k \leftarrow \mathsf{Gen}(1^n)$
m_i
$\hat{m}_i = H^s(m_i)$
$\hat{A}$

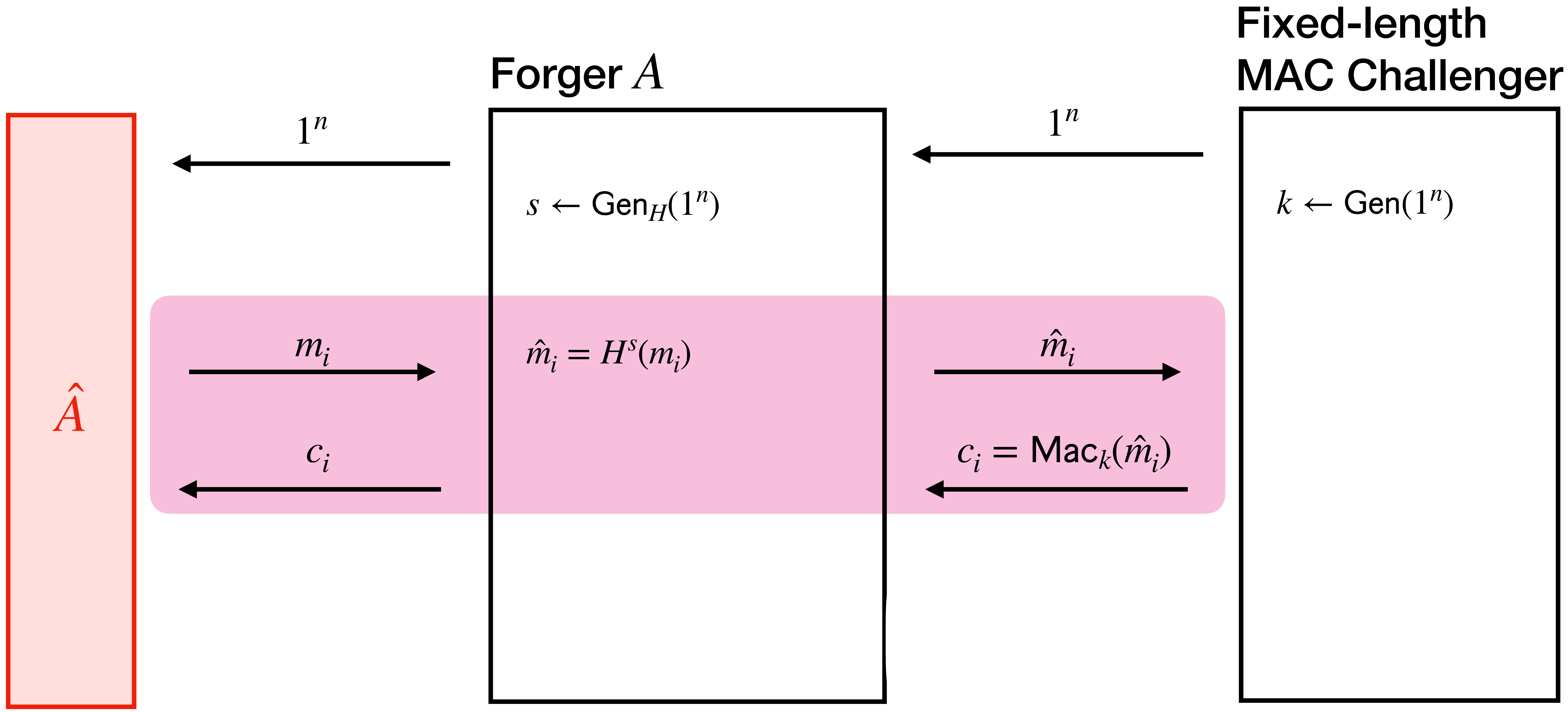
Proof of Claim 2


Forger A
Fixed-length MAC Challenger
1^n
1^n
$s \leftarrow \text{Gen}_H(1^n)$
$k \leftarrow \text{Gen}(1^n)$
m_i
$\hat{m}_i = H^s(m_i)$
$\hat{m}_i$
$\hat{A}$

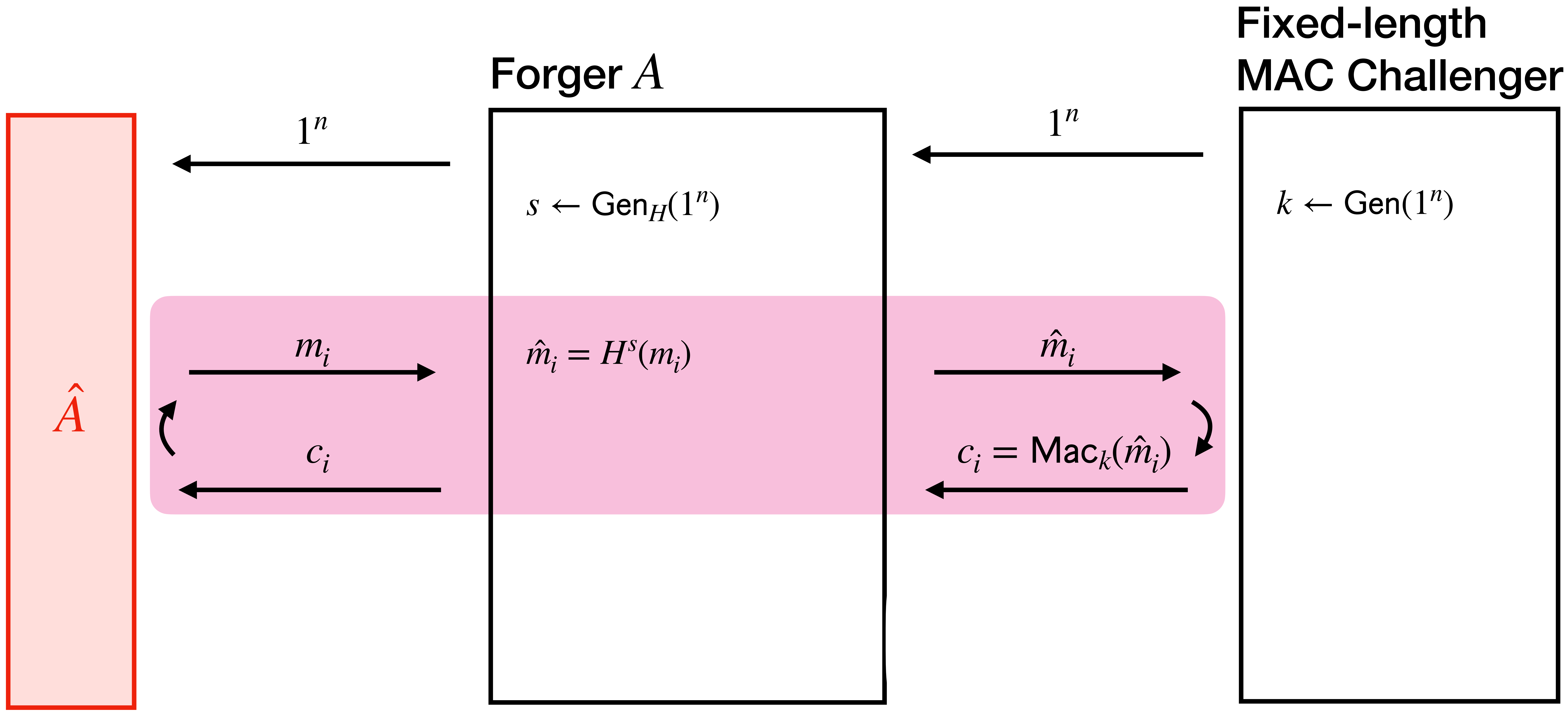
Proof of Claim 2


Forger A
Fixed-length MAC Challenger
$\hat{A}$
1^n
1^n
$s \leftarrow \mathsf{Gen}_H(1^n)$
$k \leftarrow \mathsf{Gen}(1^n)$
m_i
$\hat{m}_i = H^s(m_i)$
$\hat{m}_i$
$c_i = \mathsf{Mac}_k(\hat{m}_i)$

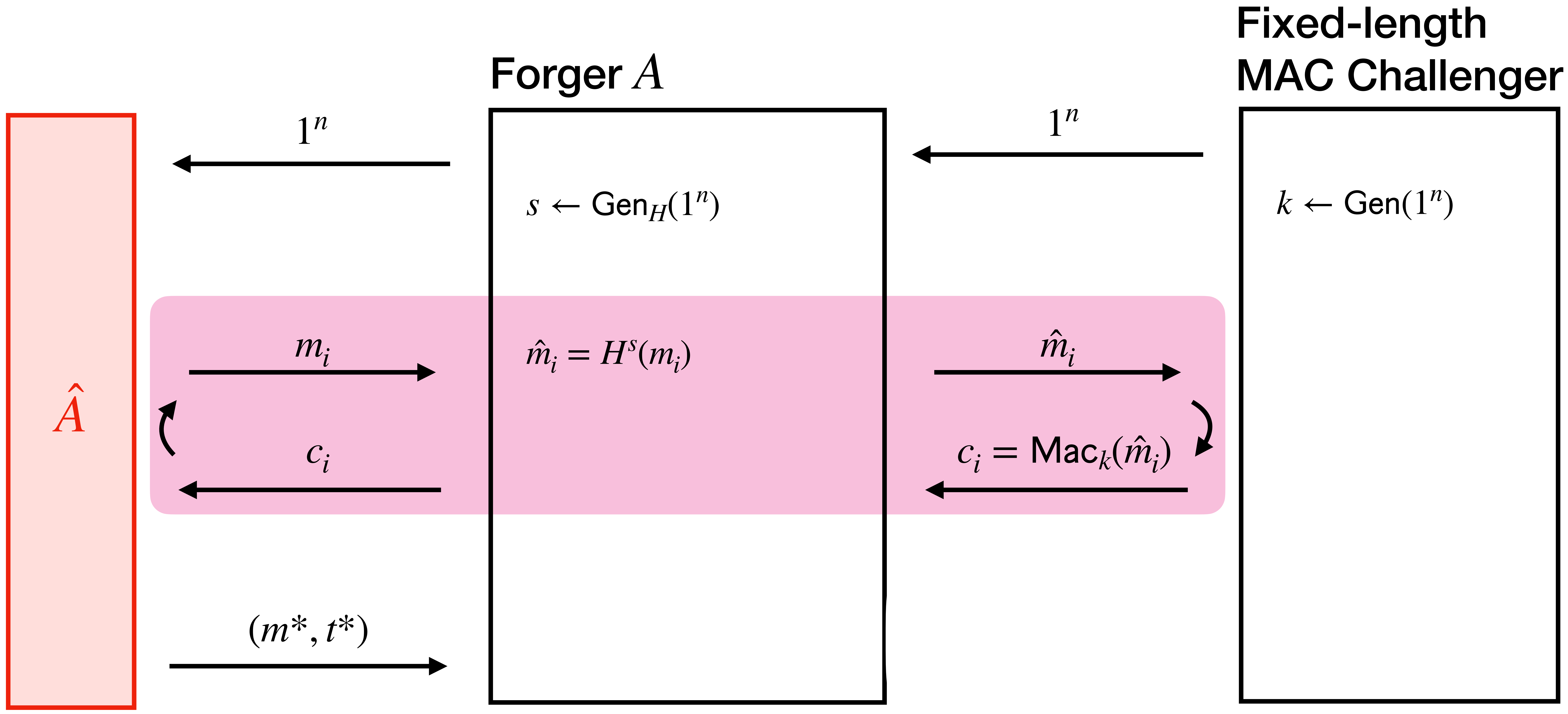
Proof of Claim 2


Forger A
Fixed-length MAC Challenger
$\hat{A}$
1^n
1^n
$s \leftarrow \mathsf{Gen}_H(1^n)$
$k \leftarrow \mathsf{Gen}(1^n)$
m_i
$\hat{m}_i = H^s(m_i)$
$\hat{m}_i$
c_i
$c_i = \mathsf{Mac}_k(\hat{m}_i)$

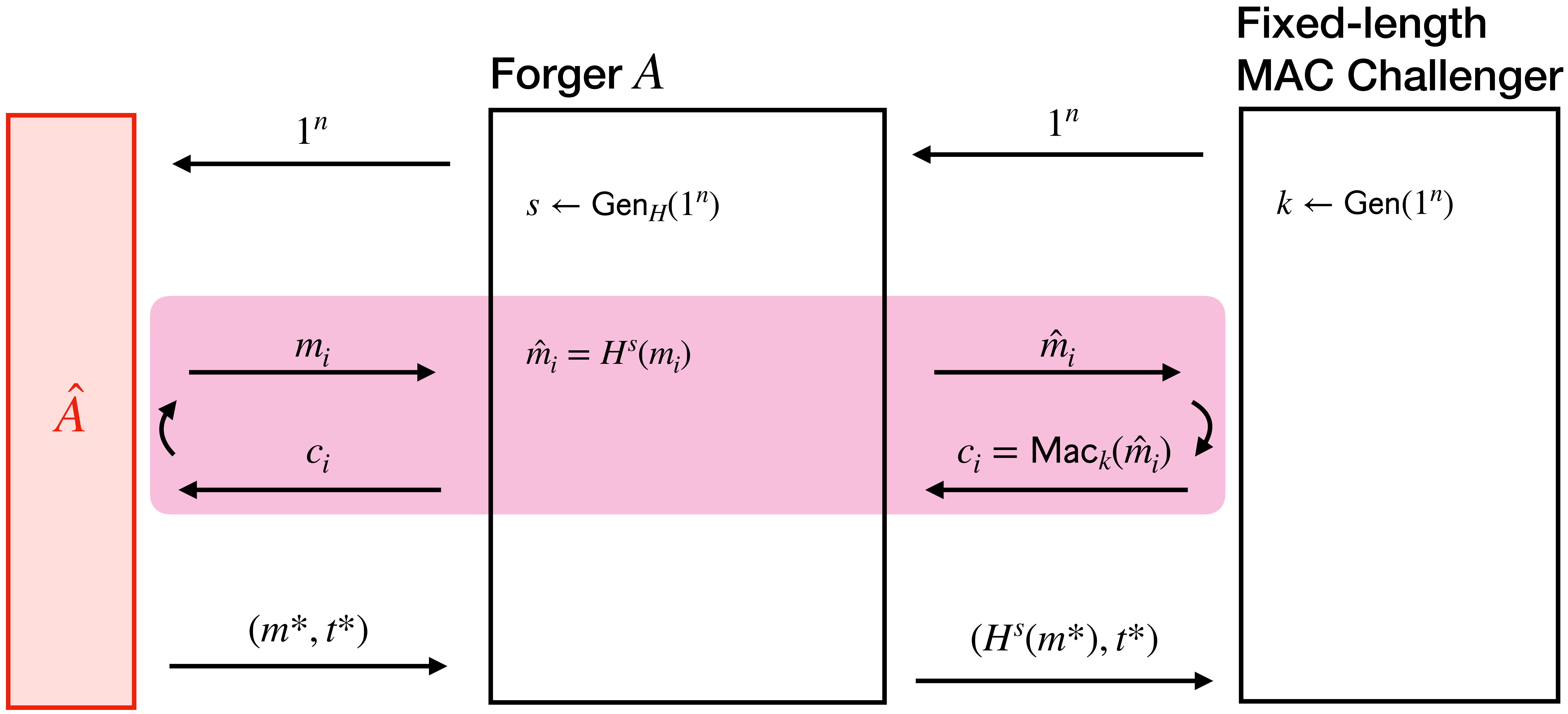
Proof of Claim 2


Forger A
Fixed-length MAC Challenger
$\hat{A}$
1^n
1^n
$s \leftarrow \mathsf{Gen}_H(1^n)$
$k \leftarrow \mathsf{Gen}(1^n)$
m_i
$\hat{m}_i = H^s(m_i)$
$\hat{m}_i$
c_i
$c_i = \mathsf{Mac}_k(\hat{m}_i)$

Proof of Claim 2


Forger A
Fixed-length MAC Challenger
$\hat{A}$
1^n
1^n
$s \leftarrow \mathsf{Gen}_H(1^n)$
$k \leftarrow \mathsf{Gen}(1^n)$
m_i
$\hat{m}_i = H^s(m_i)$
$\hat{m}_i$
c_i
$c_i = \mathsf{Mac}_k(\hat{m}_i)$
(m^*, t^*)

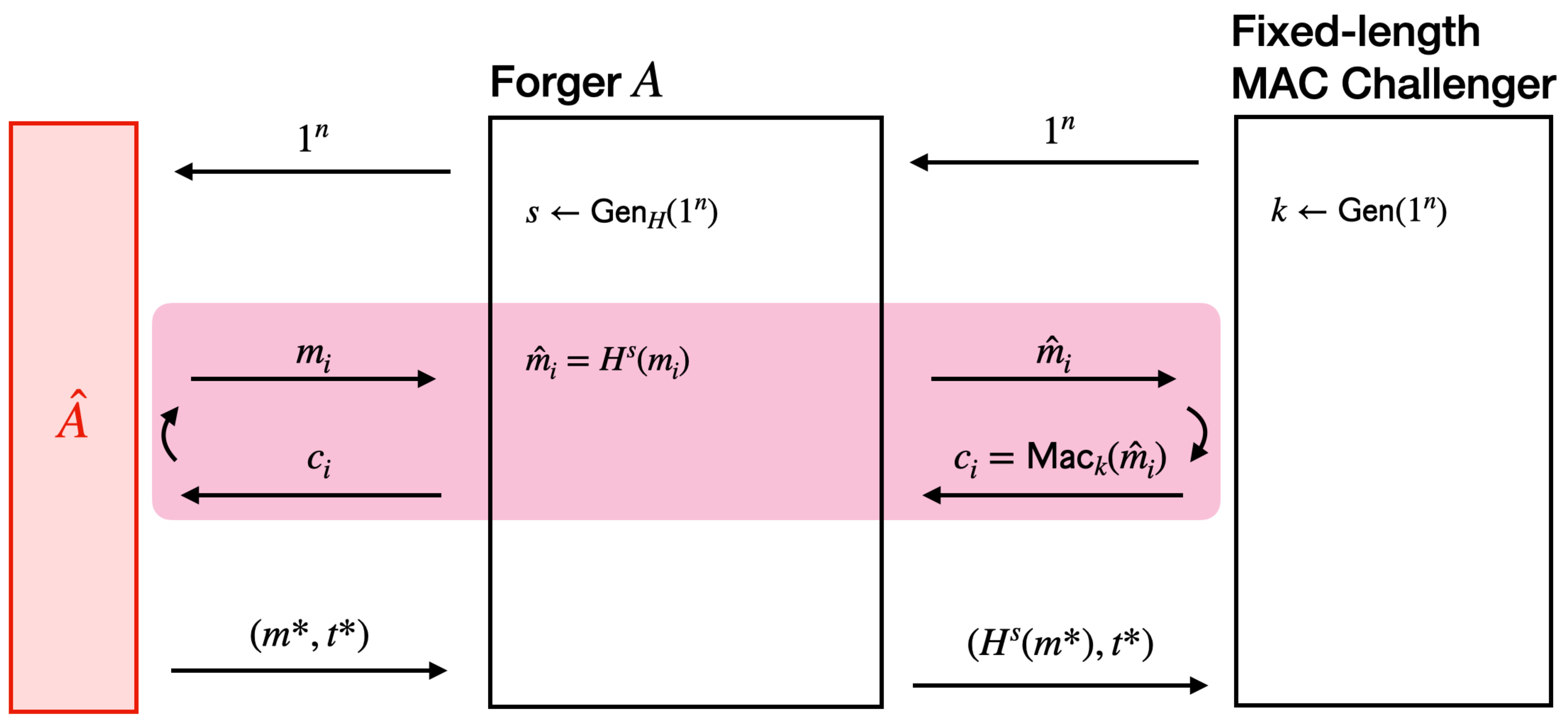
Proof of Claim 2


Forger A
Fixed-length MAC Challenger
$\hat{A}$
1^n
1^n
$s \leftarrow \mathsf{Gen}_H(1^n)$
$k \leftarrow \mathsf{Gen}(1^n)$
m_i
$\hat{m}_i = H^s(m_i)$
$\hat{m}_i$
c_i
$c_i = \mathsf{Mac}_k(\hat{m}_i)$
(m^*, t^*)
$(H^s(m^*), t^*)$

Proof of Claim 2

Assume towards contradiction that there exists an adversary $\hat{A}$ and a polynomial $p(n)$ s.t.
$\Pr[\mathsf{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1 \wedge \neg collision] \geq 1/p(n)$

$\Pr[\mathsf{MacForge}_{\mathscr{A},\Pi}]$

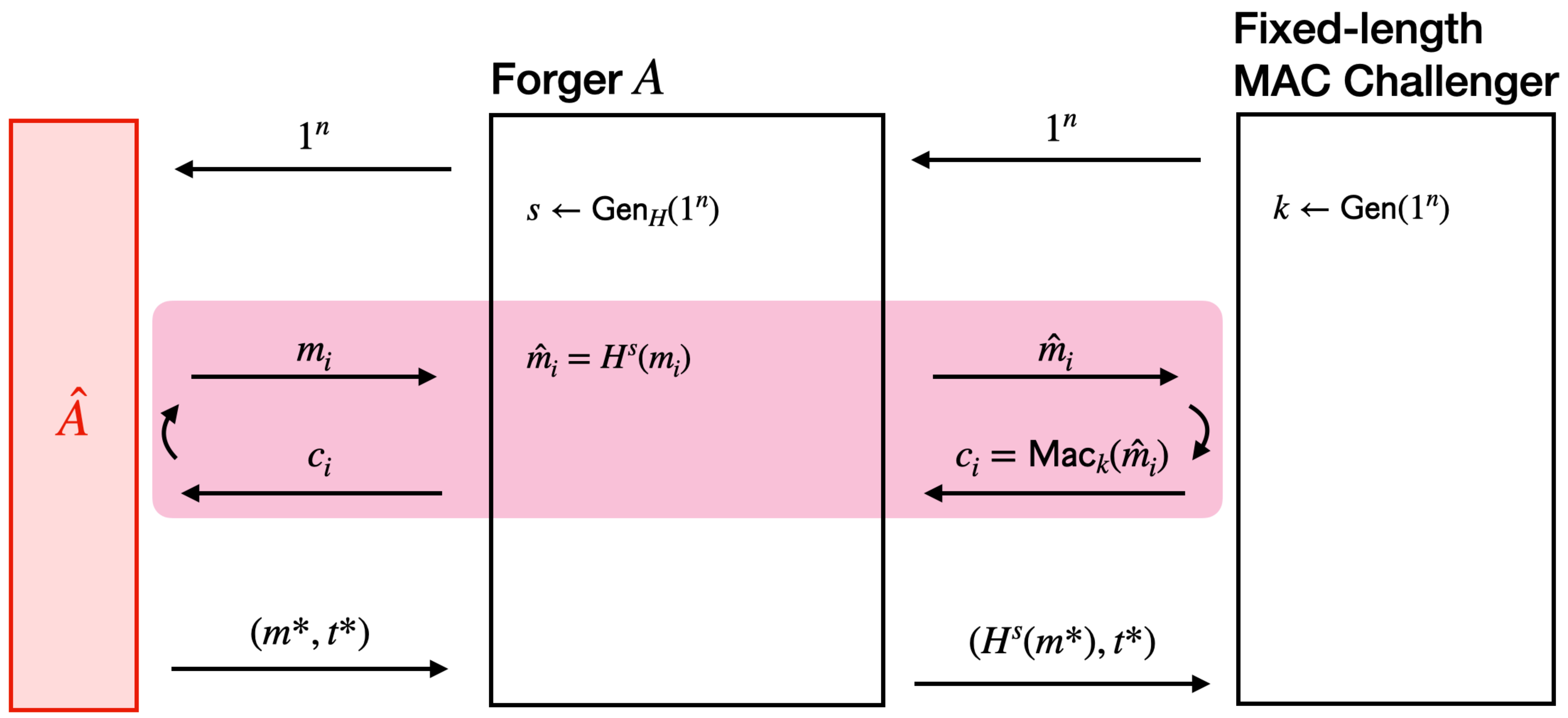

Forger A
Fixed-length MAC Challenger
$\hat{A}$
1^n
1^n
$s \leftarrow \mathsf{Gen}_H(1^n)$
$k \leftarrow \mathsf{Gen}(1^n)$
m_i
$\hat{m}_i = H^s(m_i)$
$\hat{m}_i$
c_i
$c_i = \mathsf{Mac}_k(\hat{m}_i)$
(m^*, t^*)
$(H^s(m^*), t^*)$

Contradiction!

Proof of Claim 2

Assume towards contradiction that there exists an adversary $\hat{A}$ and a polynomial $p(n)$ s.t. $\Pr[\text{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1 \wedge \neg collision] \geq 1/p(n)$

$$\Pr[\text{MacForge}_{\mathscr{A},\Pi}]$$
$$= \Pr[\text{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}} \wedge \neg collision]$$



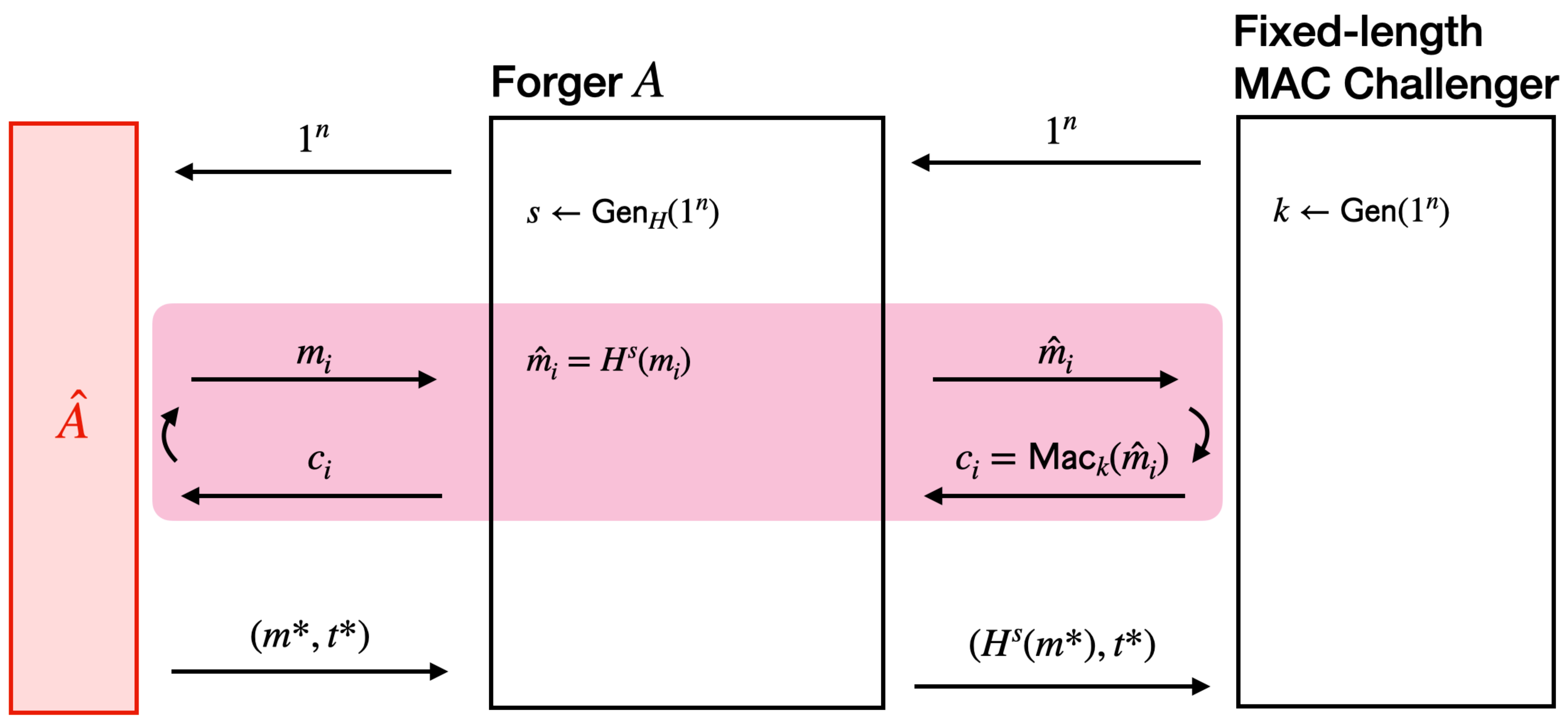
Fixed-length MAC Challenger
Forger A
$\hat{A}$
1^n
1^n
$s \leftarrow \text{Gen}_H(1^n)$
$k \leftarrow \text{Gen}(1^n)$
m_i
$\hat{m}_i = H^s(m_i)$
$\hat{m}_i$
c_i
$c_i = \text{Mac}_k(\hat{m}_i)$
(m^*, t^*)
$(H^s(m^*), t^*)$

Contradiction!

Proof of Claim 2

Assume towards contradiction that there exists an adversary $\hat{A}$ and a polynomial $p(n)$ s.t. $\Pr[\text{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}}(n) = 1 \wedge \neg collision] \geq 1/p(n)$

$$\Pr[\text{MacForge}_{\mathscr{A},\Pi}]$$
$$= \Pr[\text{MacForge}_{\hat{\mathscr{A}},\hat{\Pi}} \wedge \neg collision]$$
$$\geq 1/p(n)$$



Fixed-length MAC Challenger
Forger A
$\hat{A}$
1^n
1^n
$s \leftarrow \text{Gen}_H(1^n)$
$k \leftarrow \text{Gen}(1^n)$
m_i
$\hat{m}_i = H^s(m_i)$
$\hat{m}_i$
c_i
$c_i = \text{Mac}_k(\hat{m}_i)$
(m^*, t^*)
$(H^s(m^*), t^*)$

Contradiction!

Random Oracle Model

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 $H^s(x) = y$

$$\begin{tikzcd}
\boxed{\text{Collision Resistance}} \arrow[r, Rightarrow, shift left] & \boxed{\text{Target-Collision Resistance}} \arrow[l, Rightarrow, shift left, "/" marking] \arrow[r, Rightarrow, shift left] & \boxed{\text{Pre-image Resistance}} \arrow[l, Rightarrow, shift left, "/" marking] \arrow[r, Rightarrow] & \boxed{\text{OWF}}
\end{tikzcd}$$

Strong Notion of Security (informally)

- **Random oracle model / ideal hash:**
    - Pretend that H^s is a truly random function that everyone has access to

The Random Oracle Model

- Many constructions in practice that use a hash function we only know how to prove if the hash function H is a random function
  - We don't have a security proof under collision-resistance or pre-image resistance, but we also don't have an attack
- Can we assume H behaves like a random function?
  - No! There is no secret key and the code of H is known to all!
- Instead we consider a new idealized model where entities have oracle access to a truly random function
  - What is the difference to a PRF?

The Random Oracle Model

1. Start with a protocol Π that uses a hash function

The standard model

The random oracle model

The Random Oracle Model

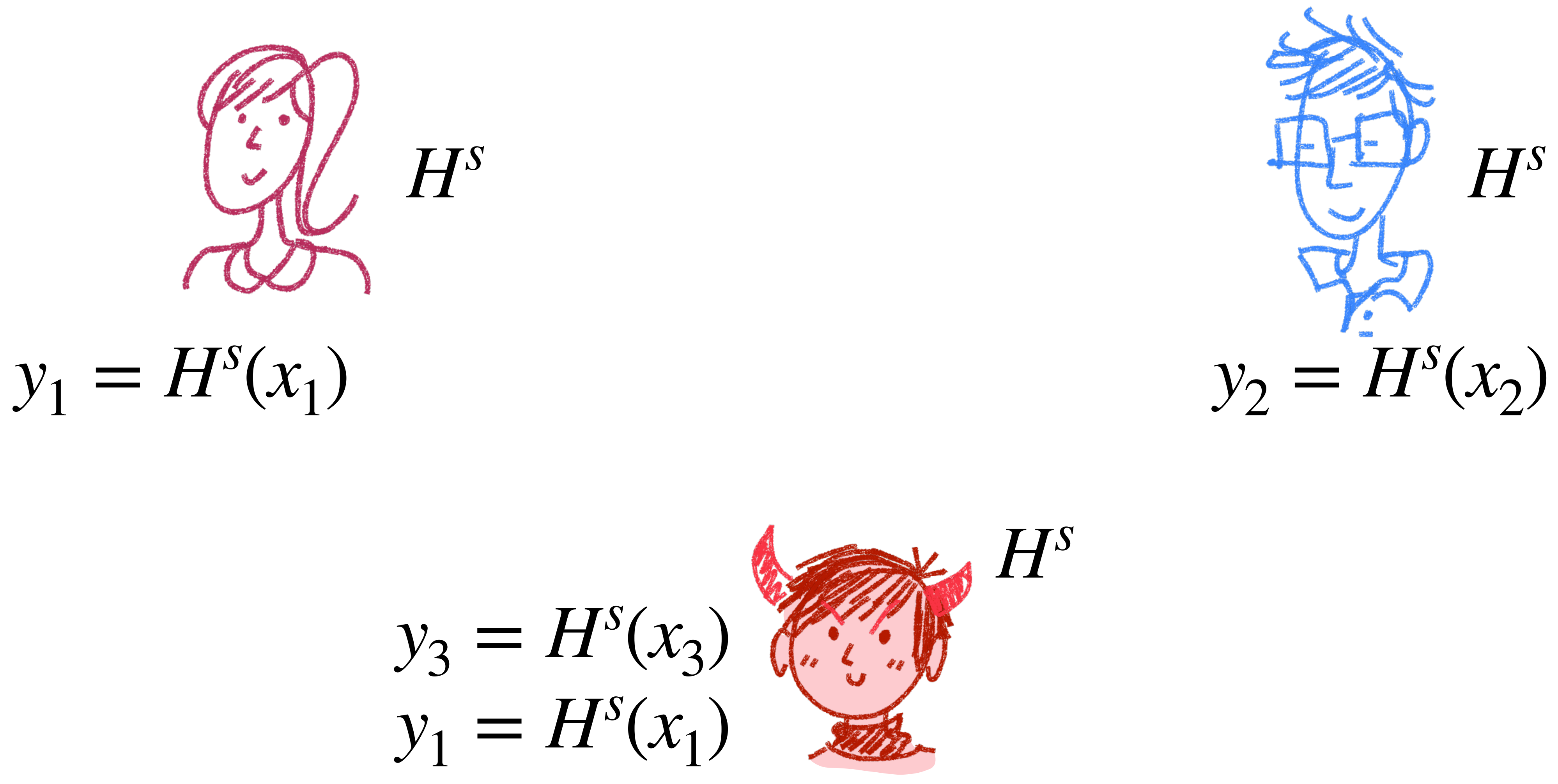
1. Start with a protocol Π that uses a hash function
   - Every party can locally compute H

The standard model

The random oracle model

The Random Oracle Model

1. Start with a protocol Π that uses a hash function
   - Every party can locally compute H

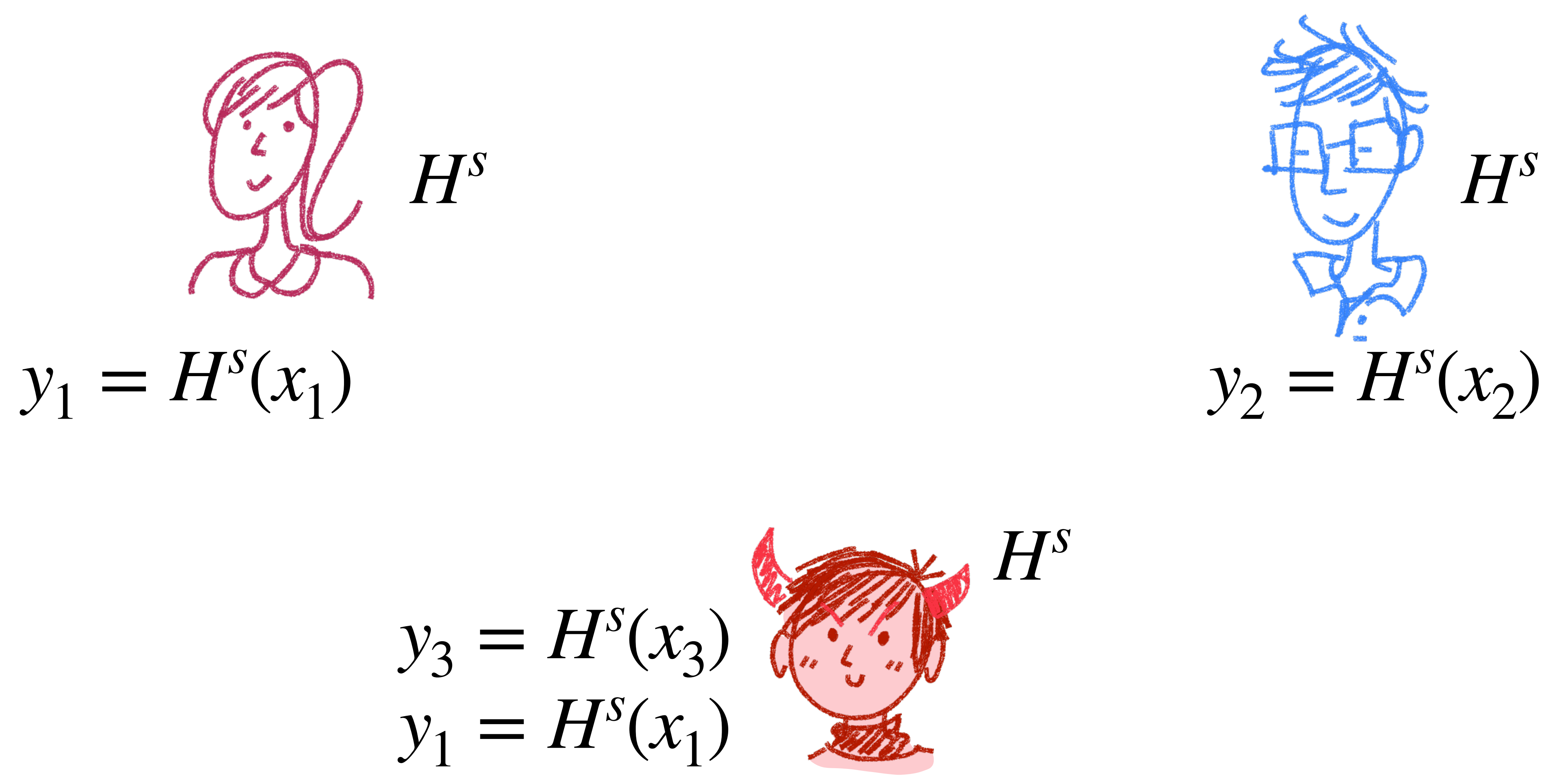


The standard model

The random oracle model

The Random Oracle Model

1. Start with a protocol Π that uses a hash function
   - Every party can locally compute H
2. Prove security of Π' in the random oracle model

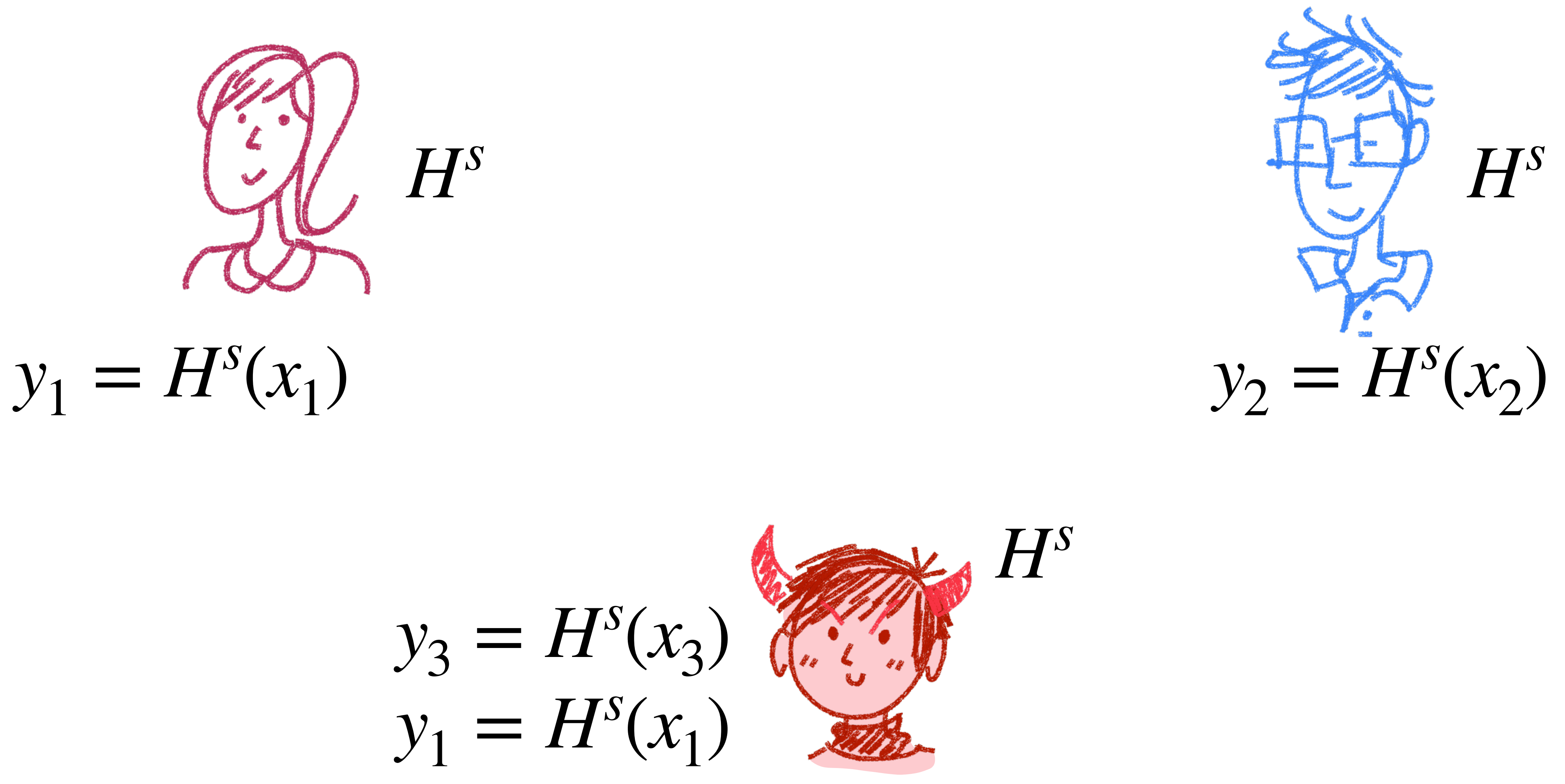


The standard model

The random oracle model

The Random Oracle Model

1. Start with a protocol Π that uses a hash function
   - Every party can locally compute H
2. Prove security of Π' in the random oracle model
   - Use an oracle to a random function instead of H

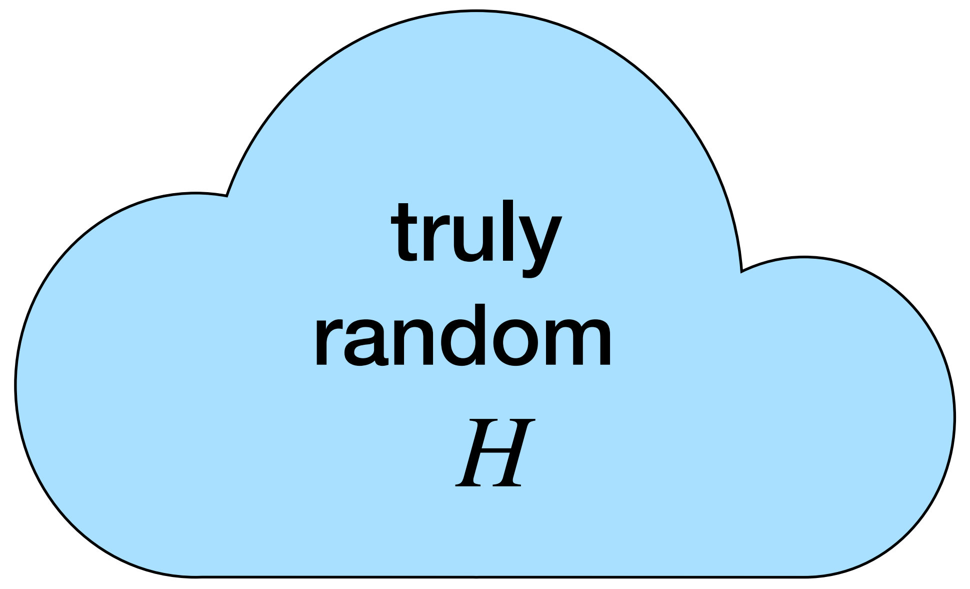


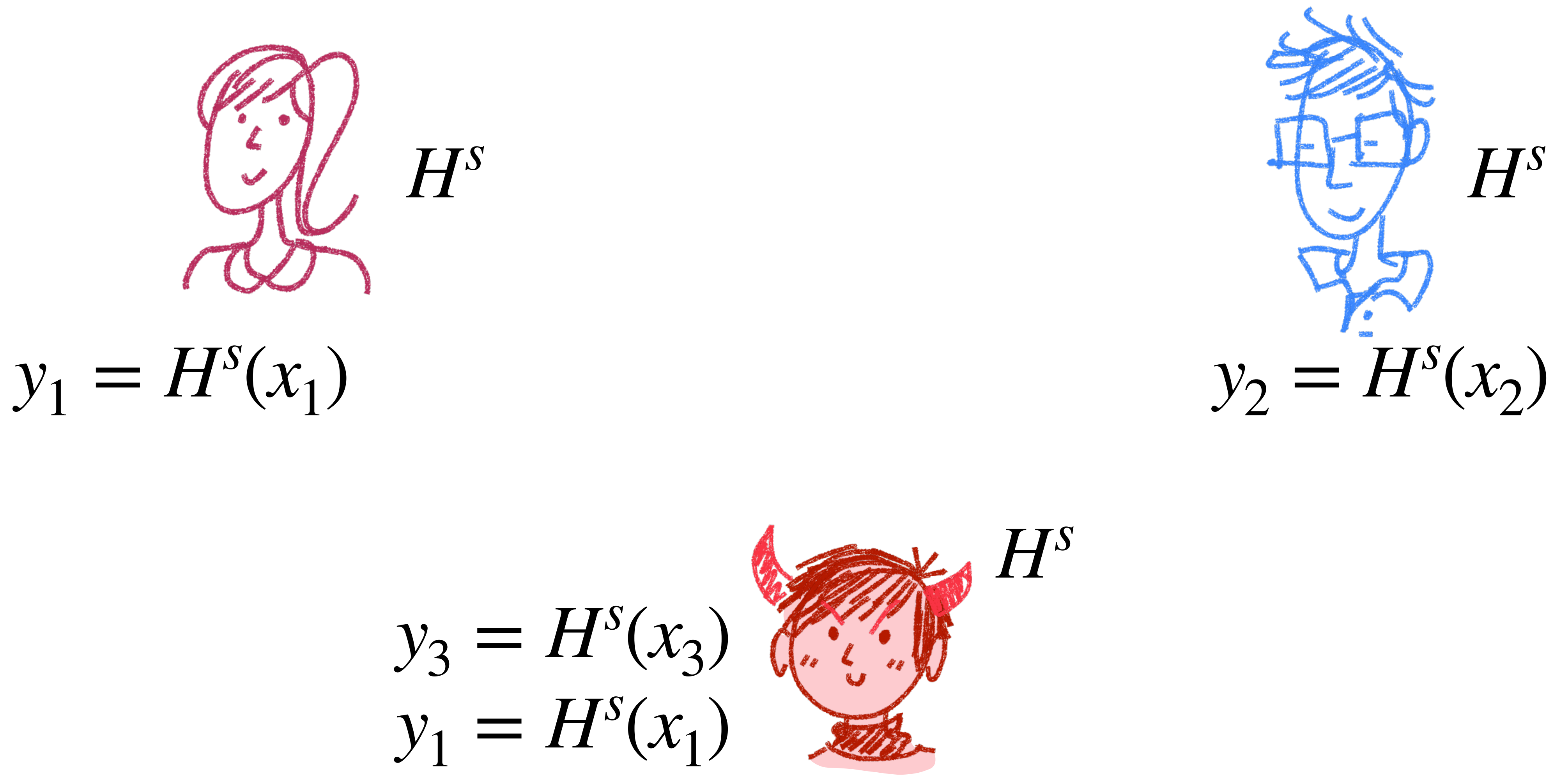
The standard model

The random oracle model

The Random Oracle Model

1. Start with a protocol Π that uses a hash function
   - Every party can locally compute H
2. Prove security of Π' in the random oracle model
   - Use an oracle to a random function instead of H



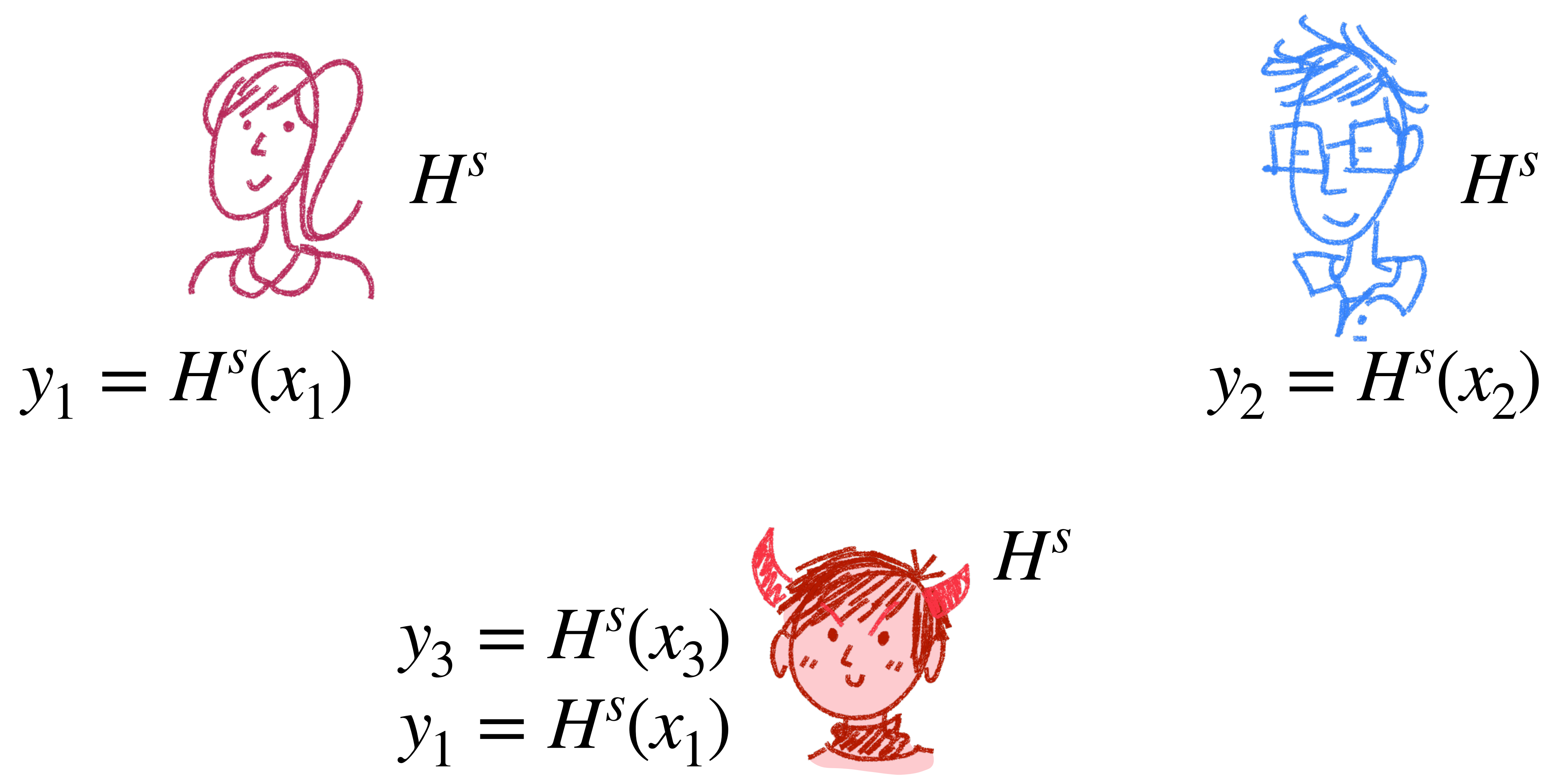


The standard model

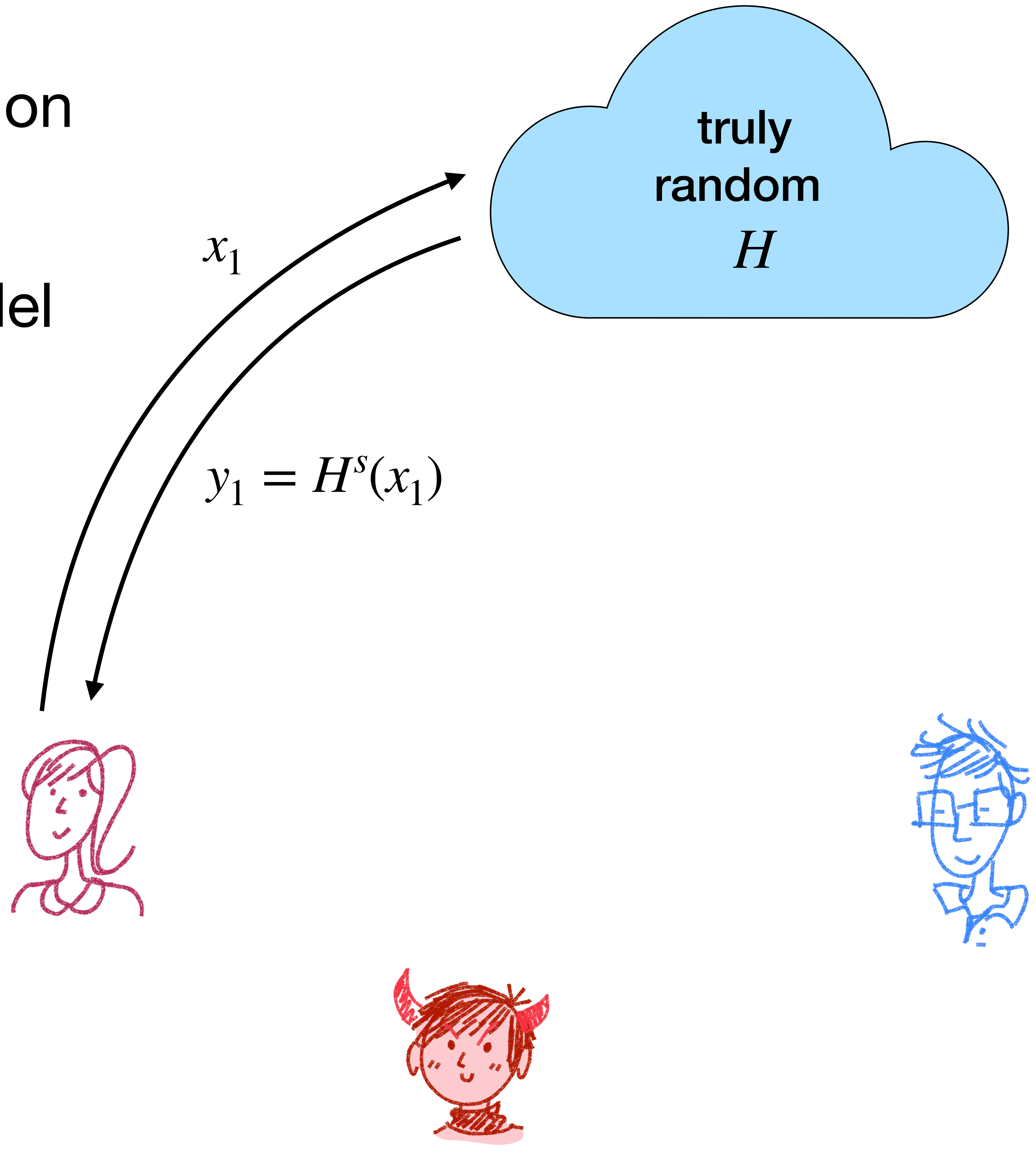
The random oracle model

The Random Oracle Model

1. Start with a protocol Π that uses a hash function
   - Every party can locally compute H
2. Prove security of Π' in the random oracle model
   - Use an oracle to a random function instead of H



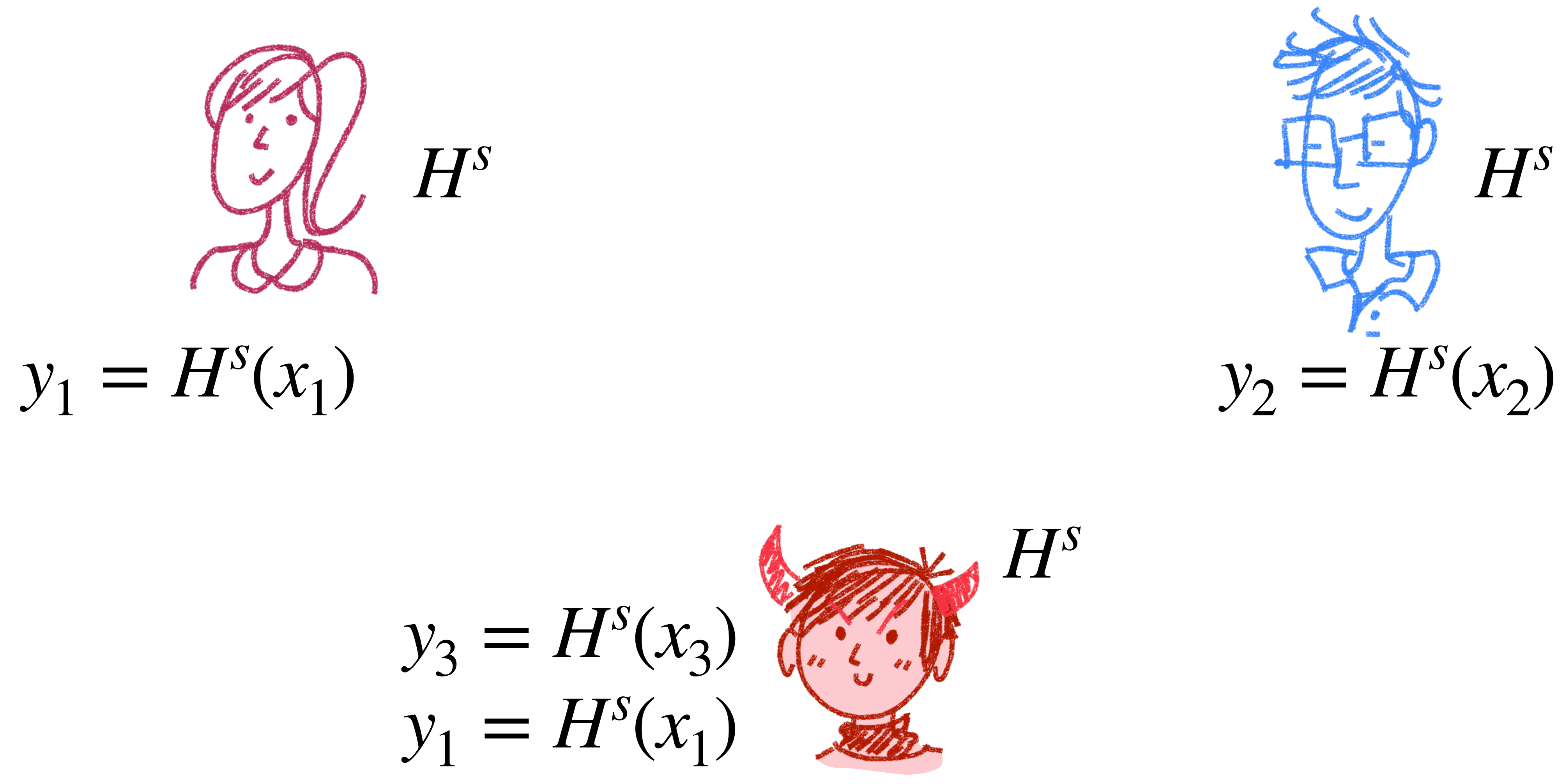
The standard model



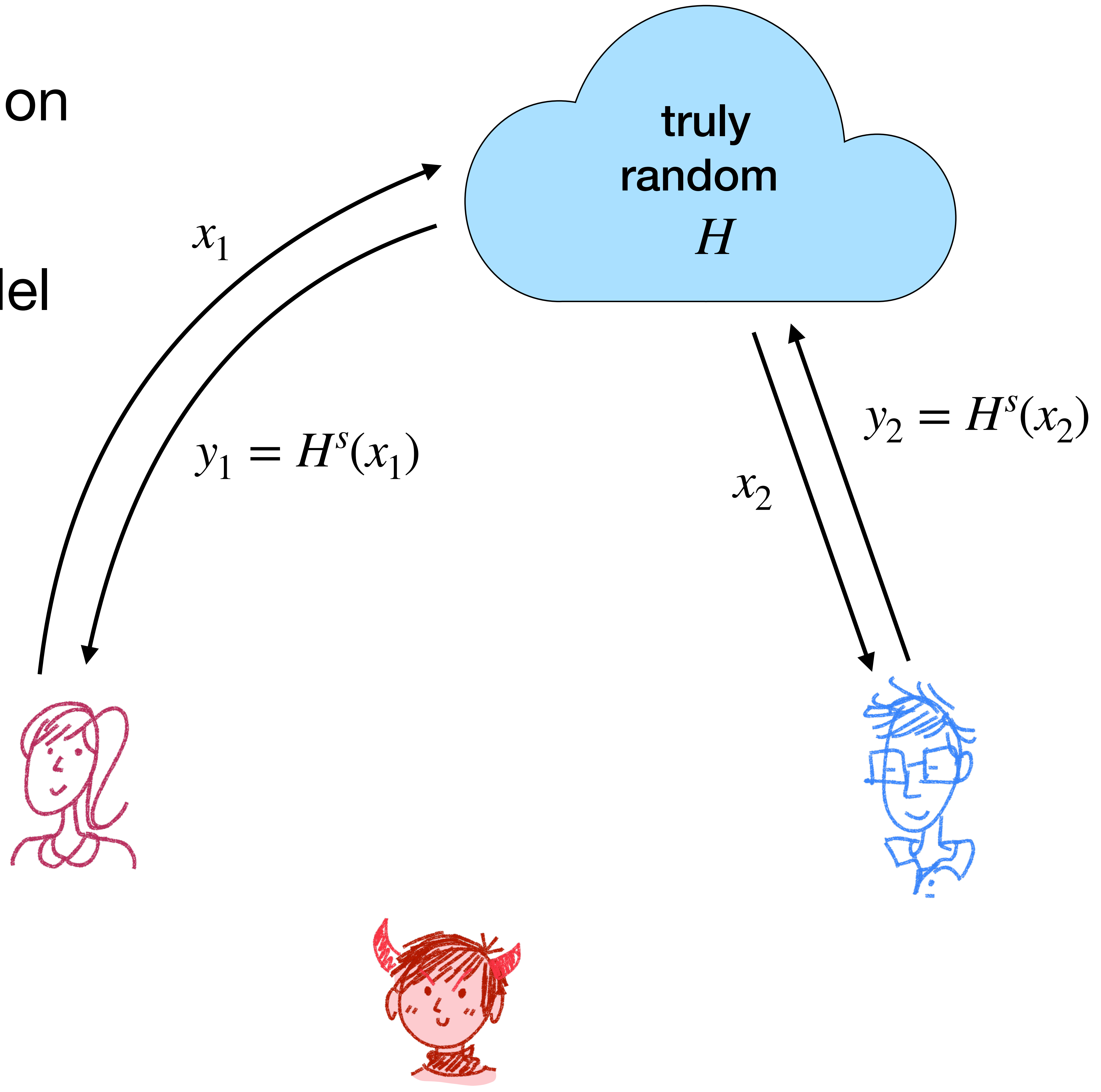
The random oracle model

The Random Oracle Model

1. Start with a protocol Π that uses a hash function
   - Every party can locally compute H
2. Prove security of Π' in the random oracle model
   - Use an oracle to a random function instead of H



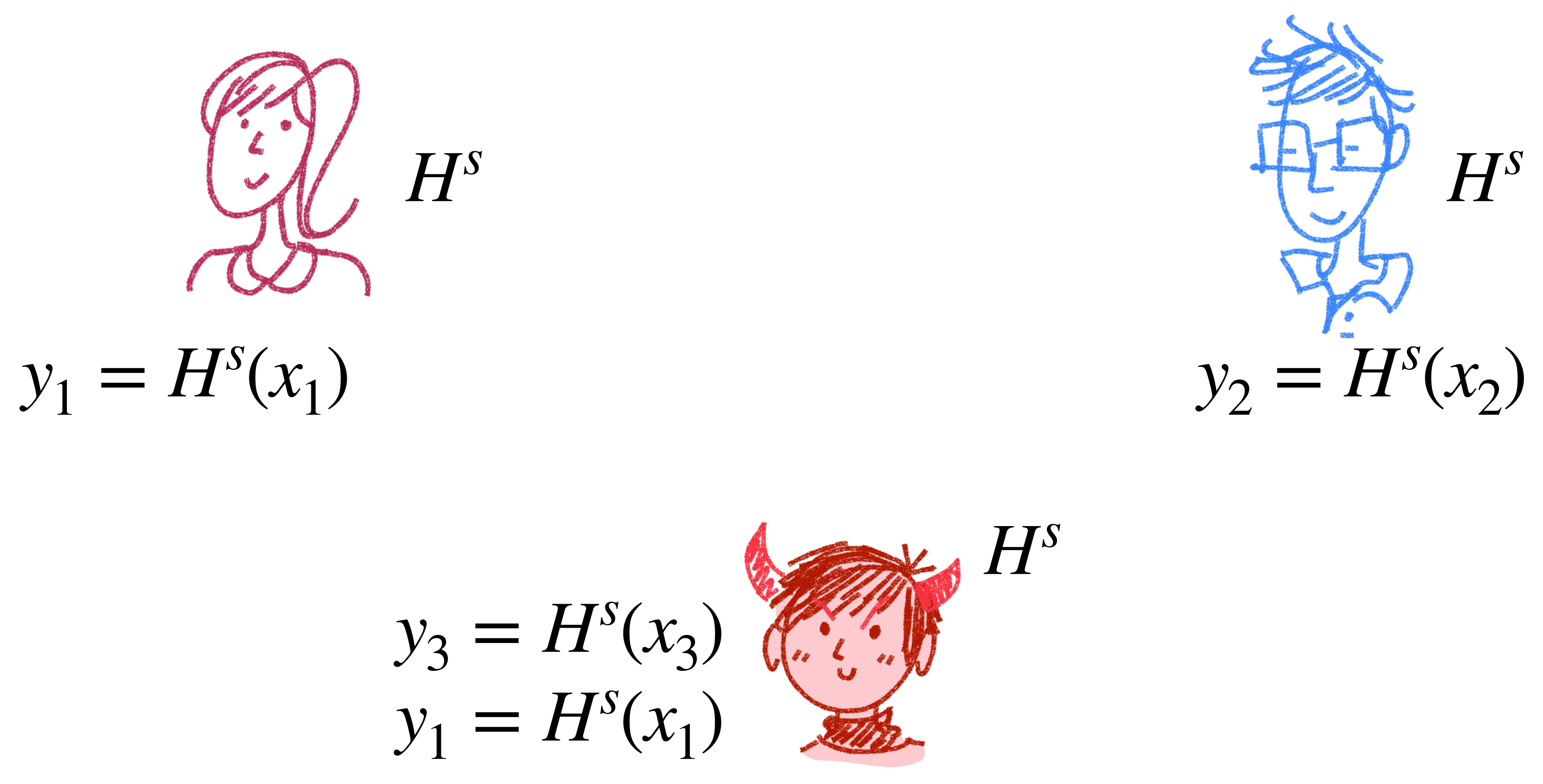
The standard model



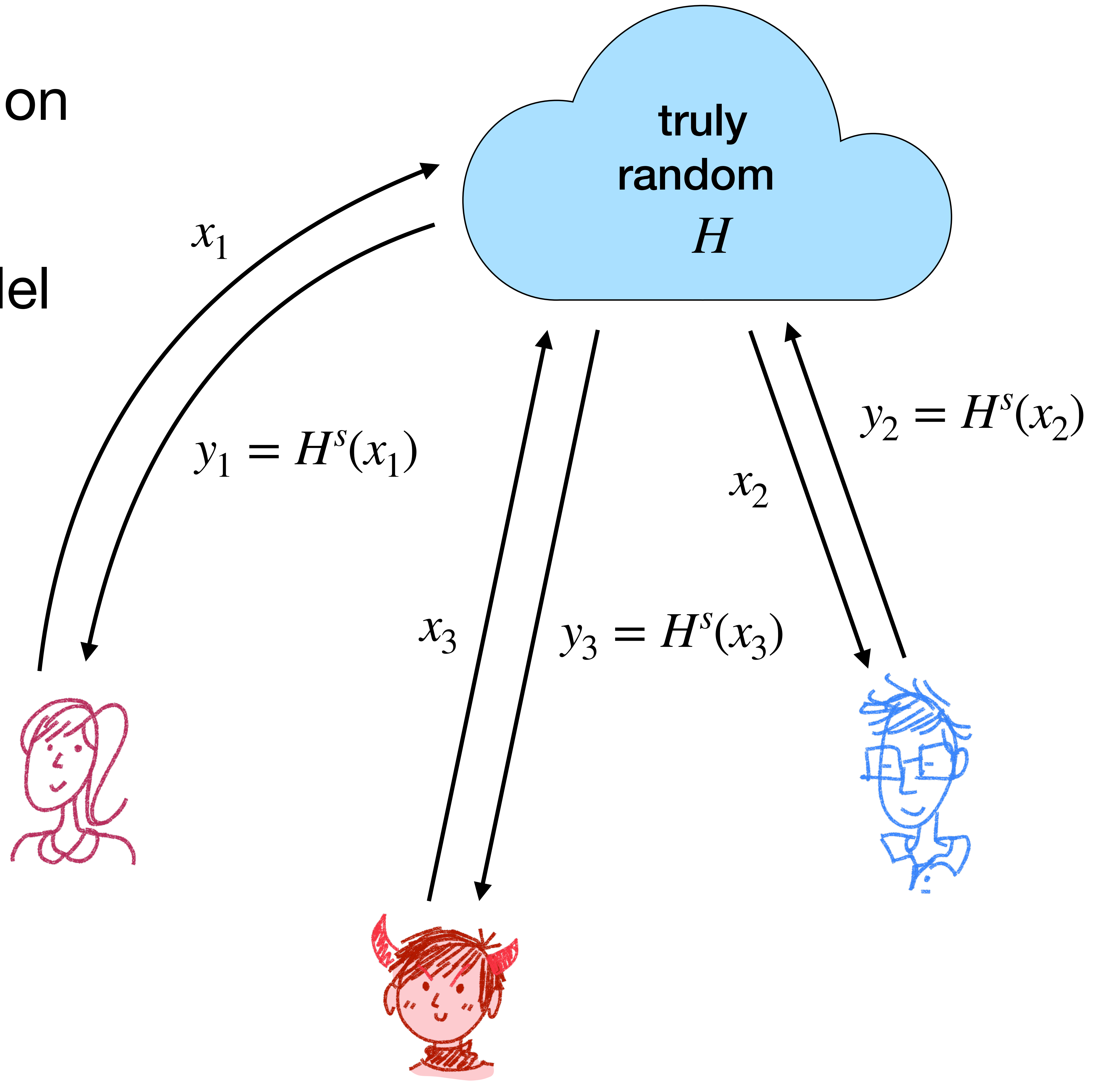
The random oracle model

The Random Oracle Model

1. Start with a protocol Π that uses a hash function
   - Every party can locally compute H
2. Prove security of Π' in the random oracle model
   - Use an oracle to a random function instead of H



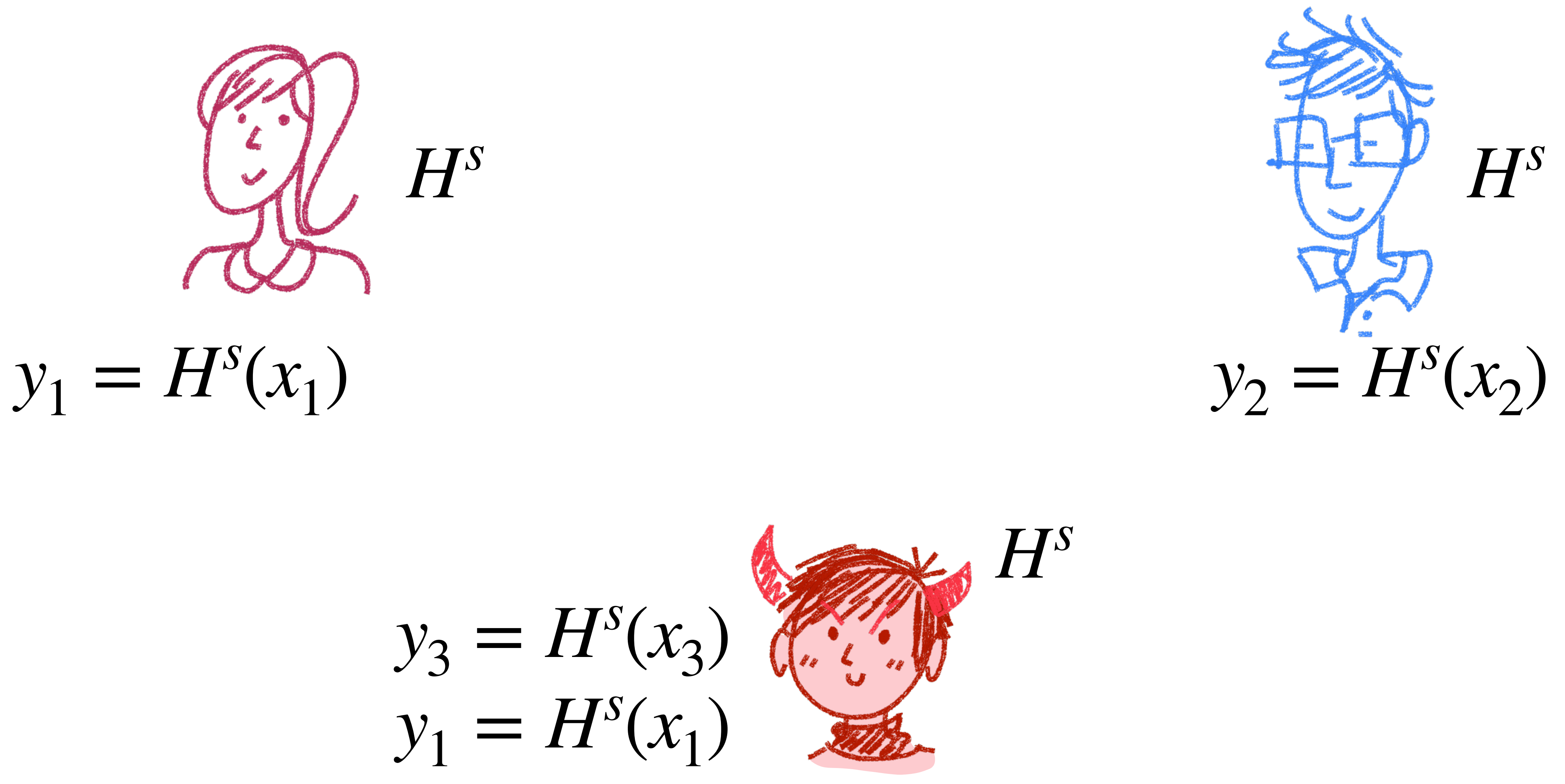
The standard model



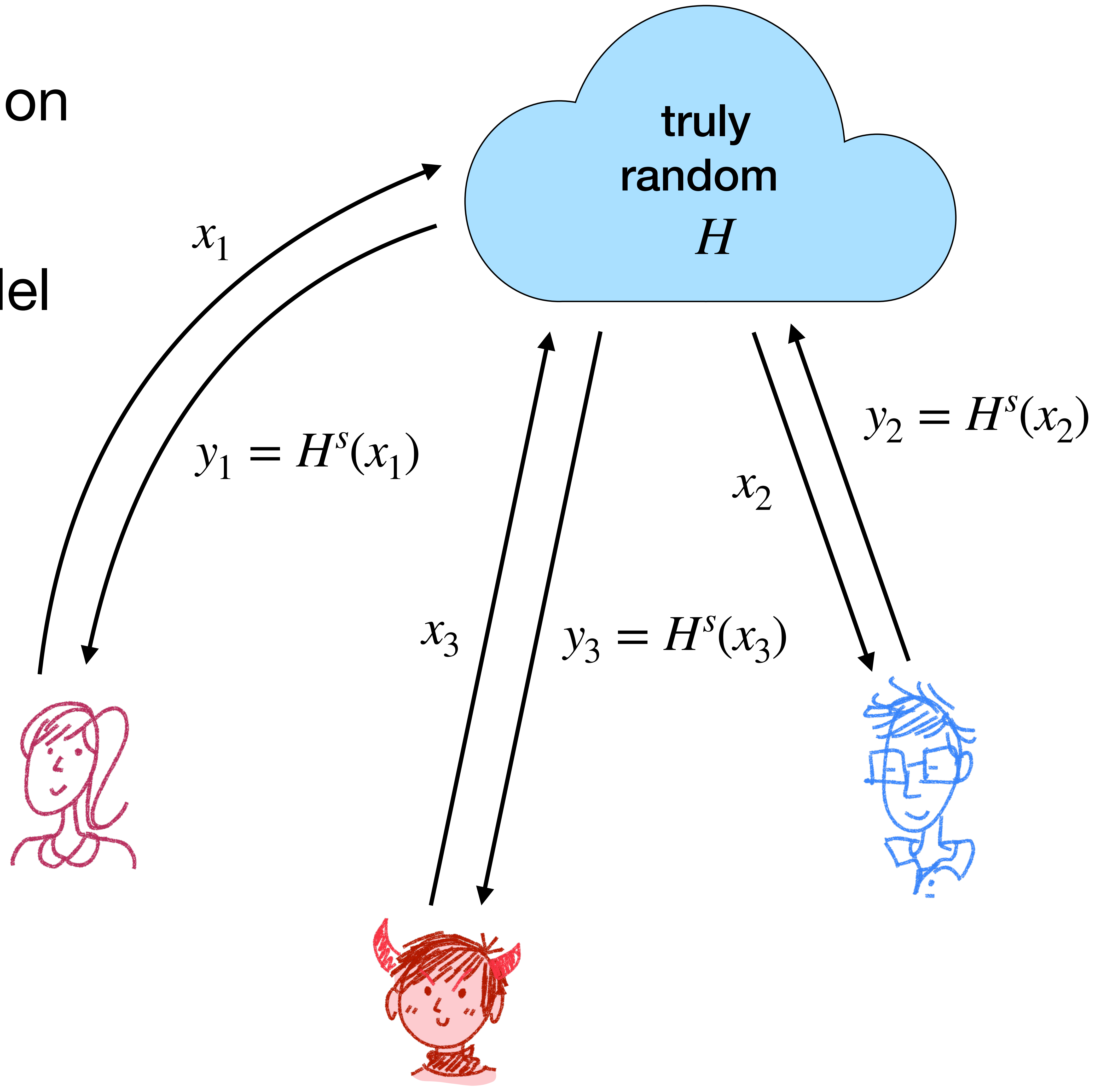
The random oracle model

The Random Oracle Model

1. Start with a protocol Π that uses a hash function
   - Every party can locally compute H
2. Prove security of Π' in the random oracle model
   - Use an oracle to a random function instead of H
3. Hope for the best…



The standard model



The random oracle model

The Random Oracle Model

- The random oracle satisfies two properties:
  - If x has not been queried to H, then $H(x)$ is uniformly distributed
  - If x has been queried before, then the result is consistent
- Some examples in the ROM with $H : \{0,1\}^{\ell_{in}} \to \{0,1\}^{\ell_{out}}$:
  - If $\ell_{in} < \ell_{out}$, then H can be used as a PRG
  - If $\ell_{in} > \ell_{out}$, then H can be used as a collision-resistant hash function
  - If $\ell_{in} = 2n$ and $\ell_{out} = n$, then $F_k(x) = H(k||x)$ is a PRF

The Random Oracle Model

- Are we saying random oracles actually exist?
  - No! This is more of a middle ground between full rigorous proofs and no proof at all
    - Schemes in the random oracle model tend to be efficient
  - You can think of the random oracle model as a methodology for designing and validating cryptographic schemes
    - If you can prove security using this idealized model, the hope is that instantiating it with a hash function is good enough

Pros and Cons of ROM

Cons

- Security in the ROM does not imply security in the standard model!
  - There are contrived examples of schemes that are secure in the ROM but are insecure when instantiated with **any** hash function
- What does it even mean for SHA-3 to “act like a random function”??

Pros

- All “natural” schemes proven secure in the ROM are not broken
- Many primitives that were first known in the ROM were later constructed in the standard model
- An attack on the scheme must use a weakness in the hash it was instantiated with
- A proof in the ROM is better than no proof at all

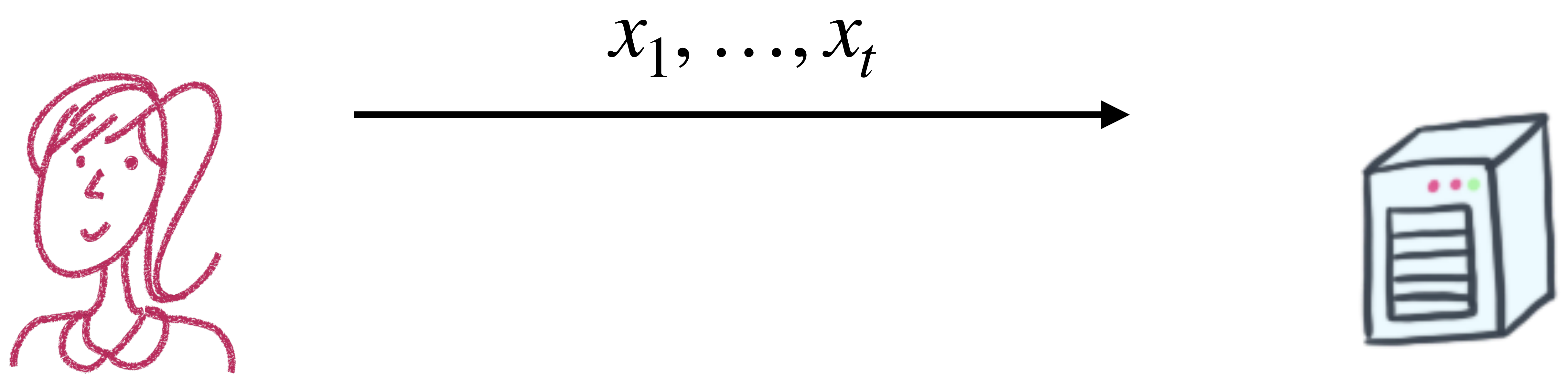
More on Applications of Hashes

Fingerprinting / equality checking

- If H is a collision resistant hash function, we can use $H(x)$ as a unique identifier for x and compare the short digest (hash output)
  - Deduplication
  - Peer-to-peer file sharing

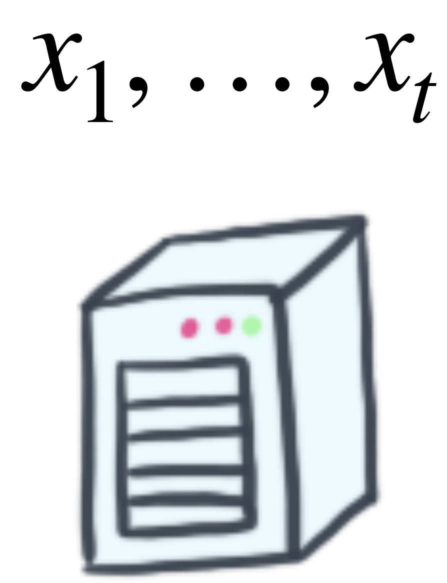
Merkle Trees

Suppose a user stores many files $x_1, \ldots, x_t$ on a server.



Merkle Trees

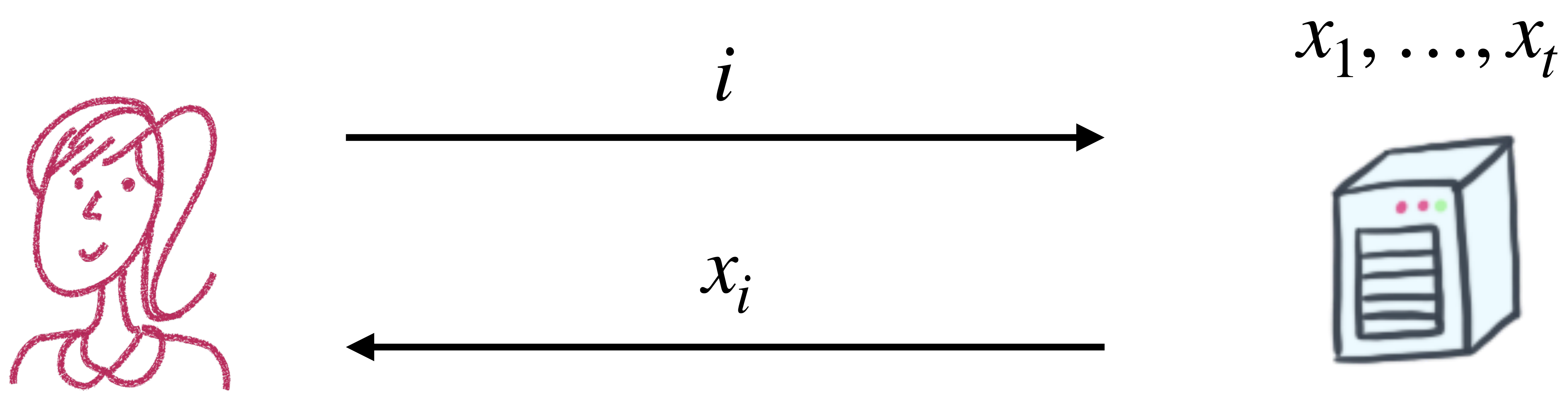
Suppose a user stores many files $x_1, \ldots, x_t$ on a server.



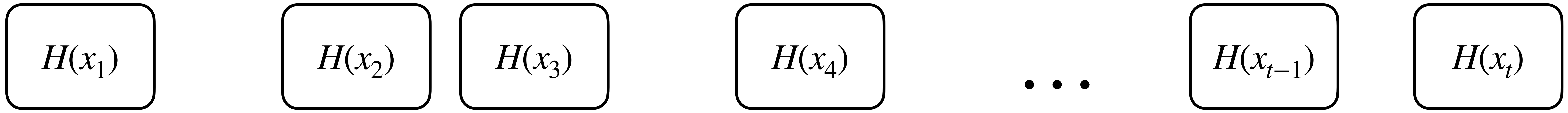
Merkle Trees

Suppose a user stores many files $x_1, \ldots, x_t$ on a server.

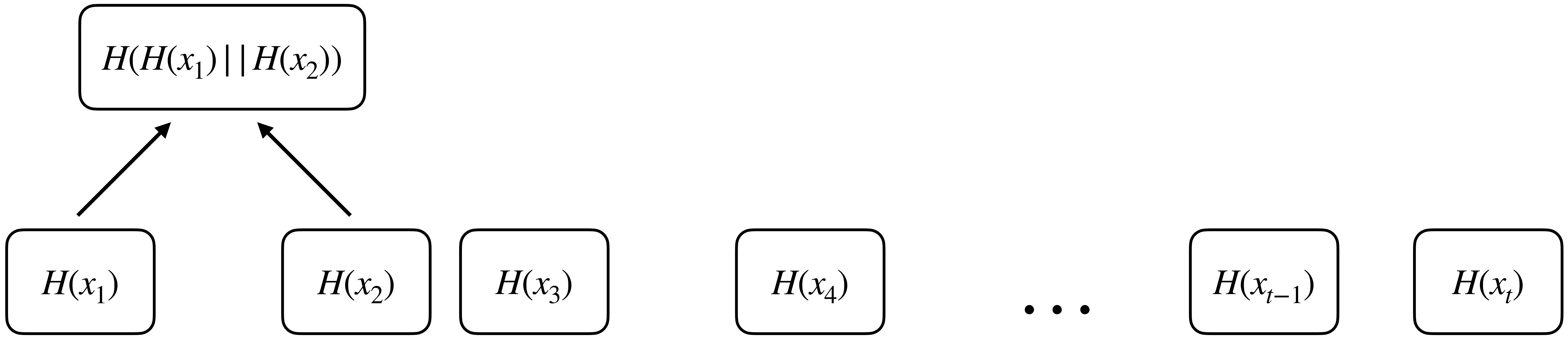
When the user later wants to retrieve one of the files, how can they check that the file was not modified?



Merkle Trees



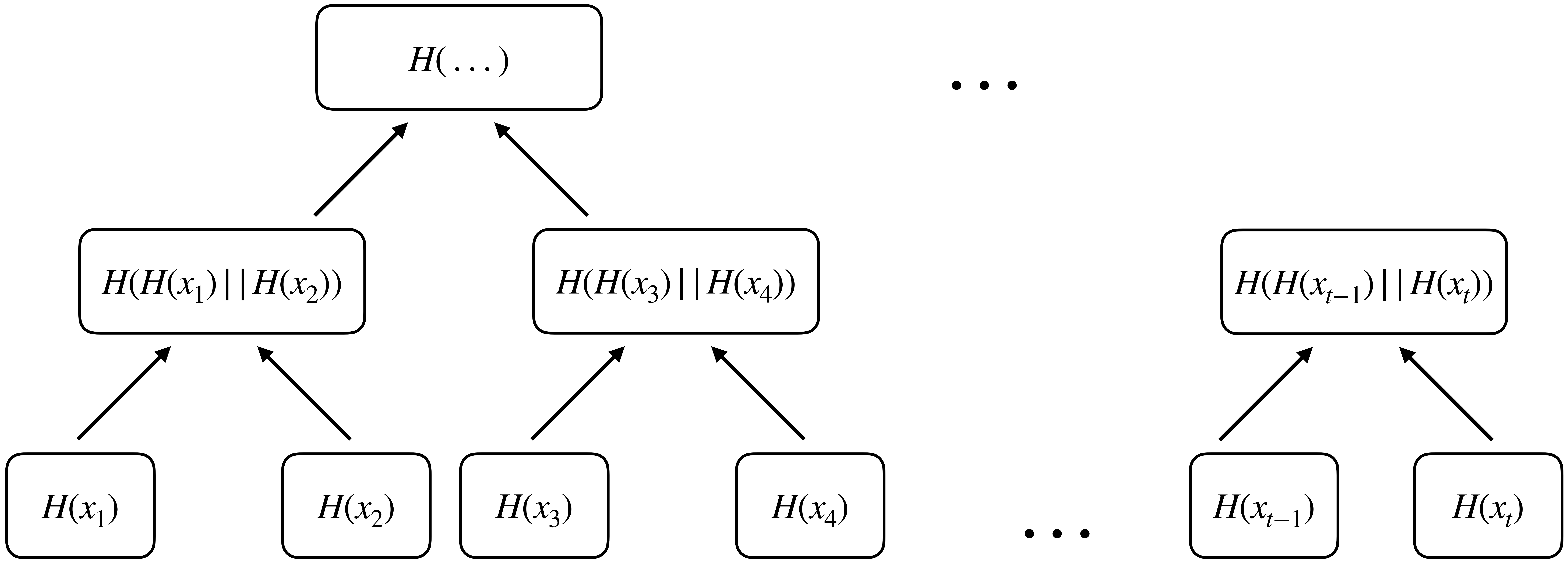
Merkle Trees



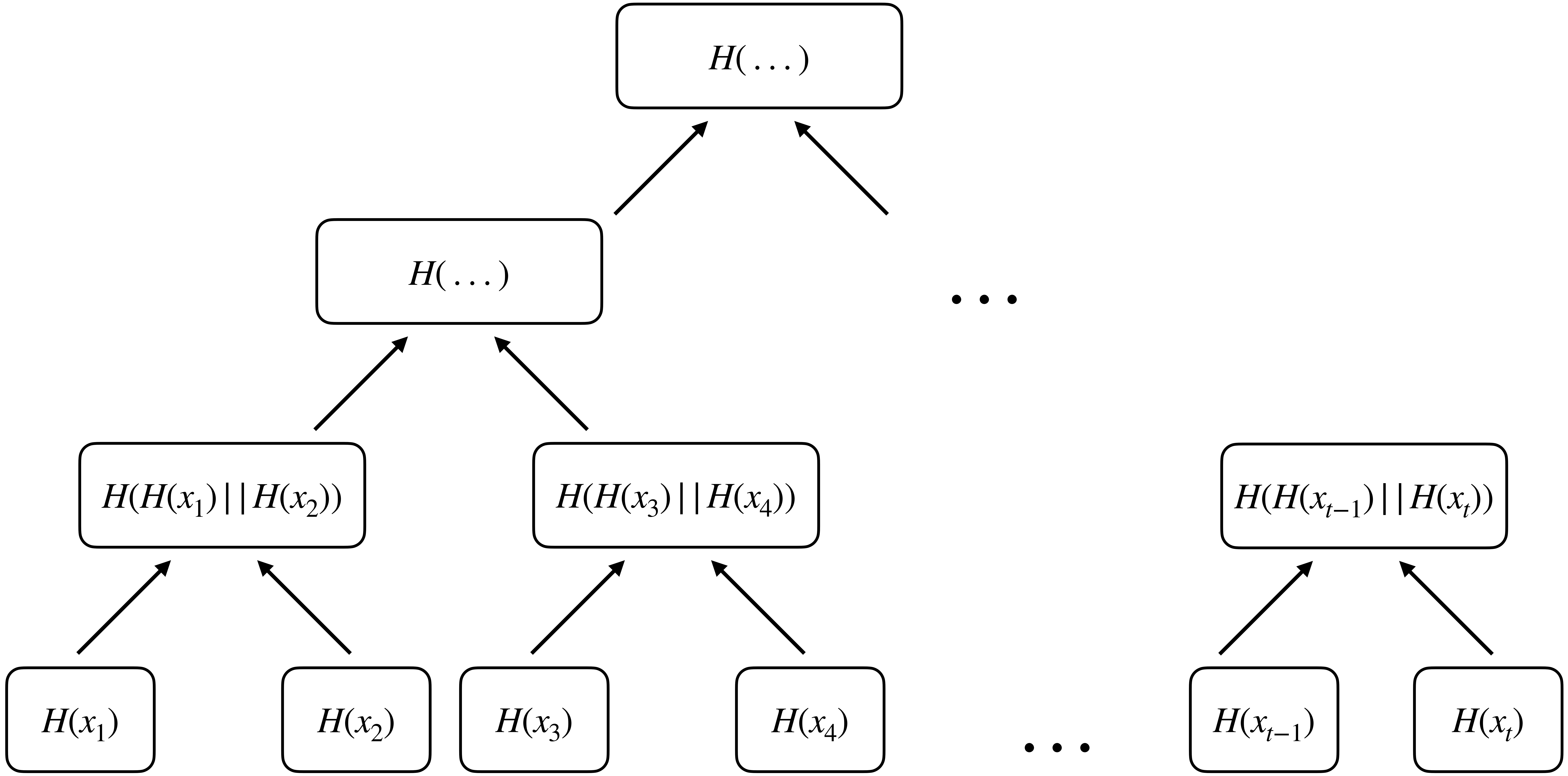
Merkle Trees

$$\begin{tikzcd}
& \boxed{H(H(x_1)\,||\,H(x_2))} & & & \boxed{H(H(x_3)\,||\,H(x_4))} & & & & \boxed{H(H(x_{t-1})\,||\,H(x_t))} & \\
\boxed{H(x_1)} \arrow[ur] & & \boxed{H(x_2)} \arrow[ul] & \boxed{H(x_3)} \arrow[ur] & & \boxed{H(x_4)} \arrow[ul] & \cdots & \boxed{H(x_{t-1})} \arrow[ur] & & \boxed{H(x_t)} \arrow[ul]
\end{tikzcd}$$

Merkle Trees



Merkle Trees



Next Time

- Today
    - More hash functions
- Monday
    - Number Theory