COMS BC3262: Introduction to Cryptography

Lecture 6: OWFs and MACs

February 16, 2026

Logistics

Office hours:

- **Eysa**: Mondays 3-5, Milstein 512
- **Mark**: Normally Tuesdays 6:30-8:30, but he is traveling these next two weeks.
    - *No office hours Tuesday, Feb 17 or Tuesday, Feb 24*
    - *Mark's next two office hours are tentatively set for Sunday via Zoom, time TBD*

PS2 is due Thursday

Each late day is 10% off, but we'll cap the late deduction at 30%

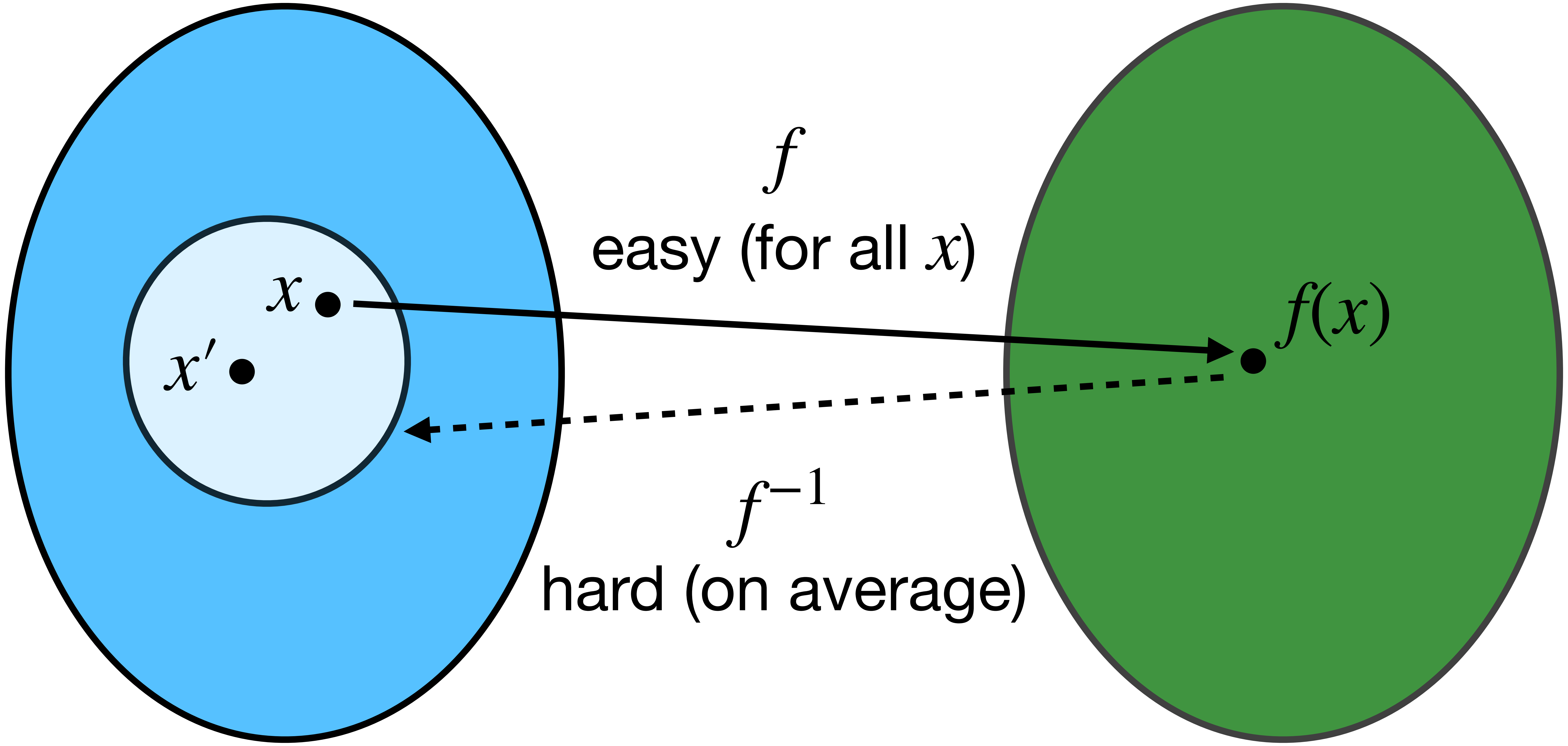
Today's Lecture

- One-Way Functions (OWFs)
- Message Authentication Codes (MACs)

One-Way Functions (OWFs)

One-Way Functions (OWFs)

A function $f: \{0,1\}^* \rightarrow \{0,1\}^*$ that is easy to compute but hard to invert



One-Way Functions

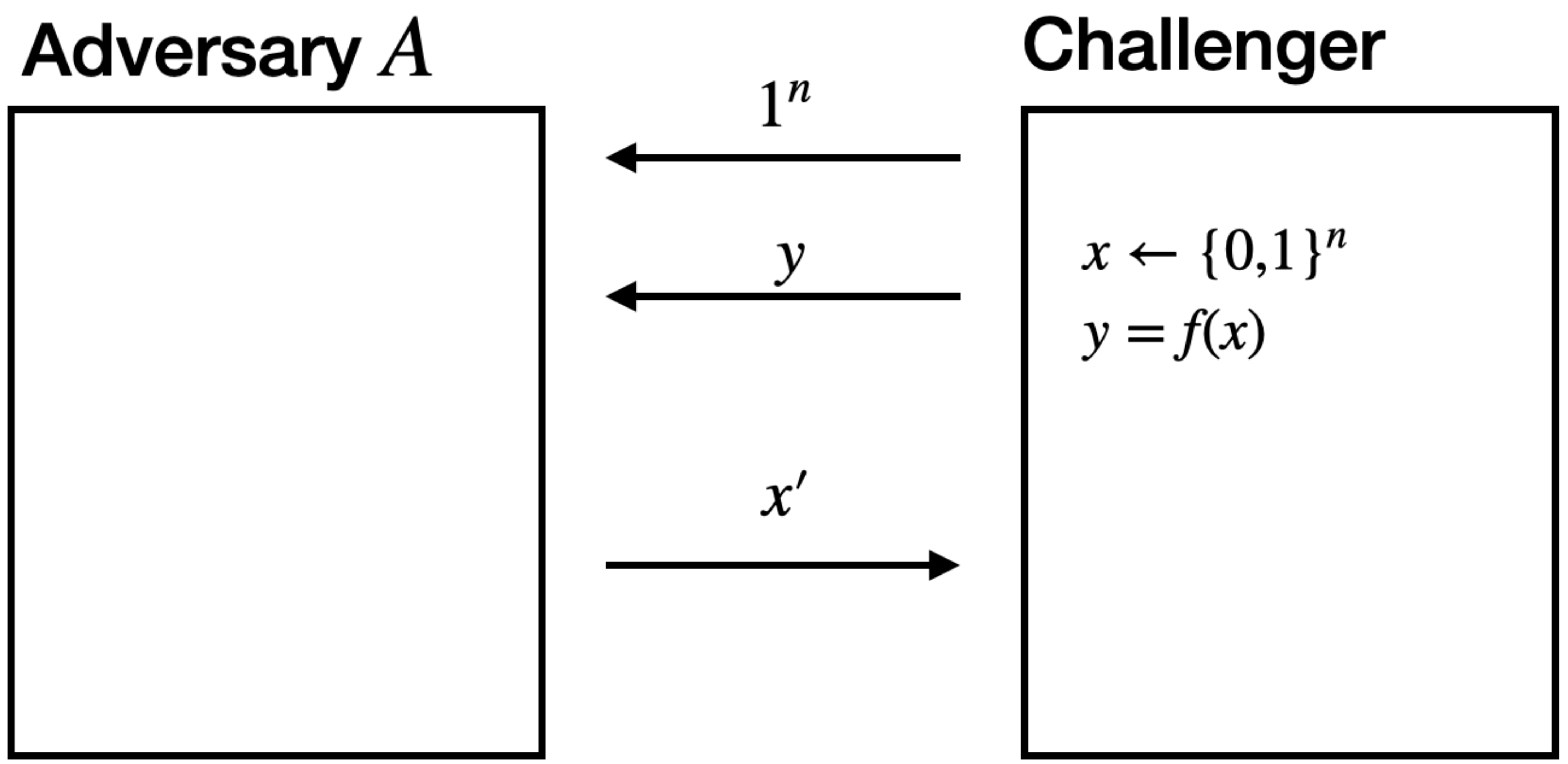
Given $f: \{0,1\}^* \rightarrow \{0,1\}^*$ and an adversary A, consider the experiment $\mathsf{Invert}_{f,A}(n)$:

Definition:

$f: \{0,1\}^* \rightarrow \{0,1\}^*$ is a **one-way function** if f can be computed in polynomial time, and for every PPT adversary A there exists a negligible function $\epsilon(\cdot)$ such that

$$\Pr[\mathsf{Invert}_{f,A}(n) = 1] \leq \epsilon(n)$$

where the probability is taken over the random coins used by A and by the experiment.



$\mathsf{Invert}_{f,A}(n) = 1$ if $f(x') = y$
and 0 otherwise

One-Way Functions

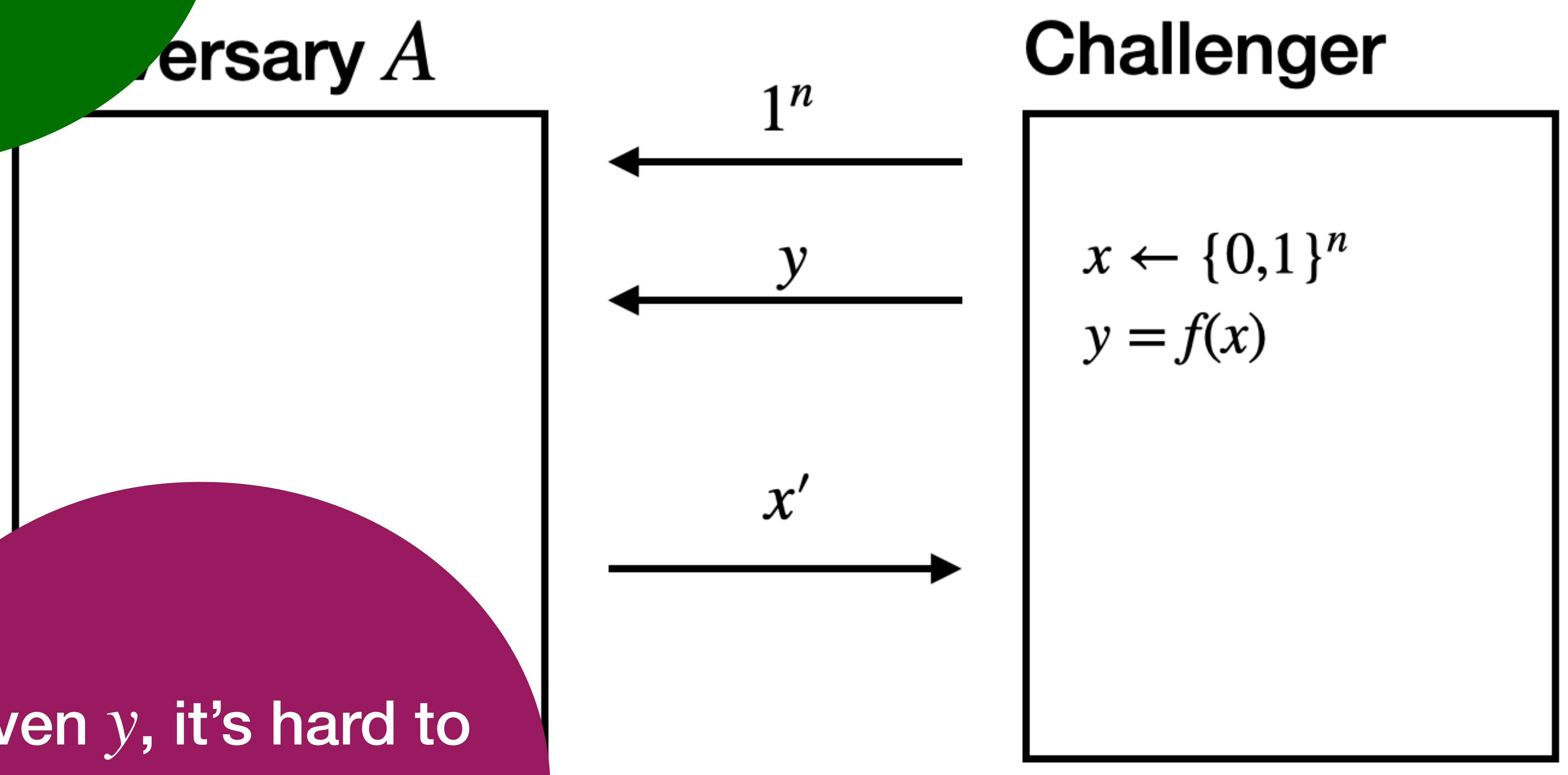
Given $f: \{0,1\}^* \to \{0,1\}^*$ and an ad… der the experiment $\mathsf{Invert}_{f,A}(n)$:

Definition:

$f: \{0,1\}^* \to \{0,1\}^*$ is a **one-way fun… ion** if f can be computed in polynomial time, and for every PPT adversary A there exists a negligible function $\epsilon(\cdot)$ such that

$$\Pr[\mathsf{Invert}_{f,A}(n) = 1] \leq \epsilon(n)$$

where the probability is taken over the random coins used by A and by the experiment.



$\mathsf{Invert}_{f,A}(n) = 1$ if $f(x') = y$ and 0 otherwise

Some facts about OWFs

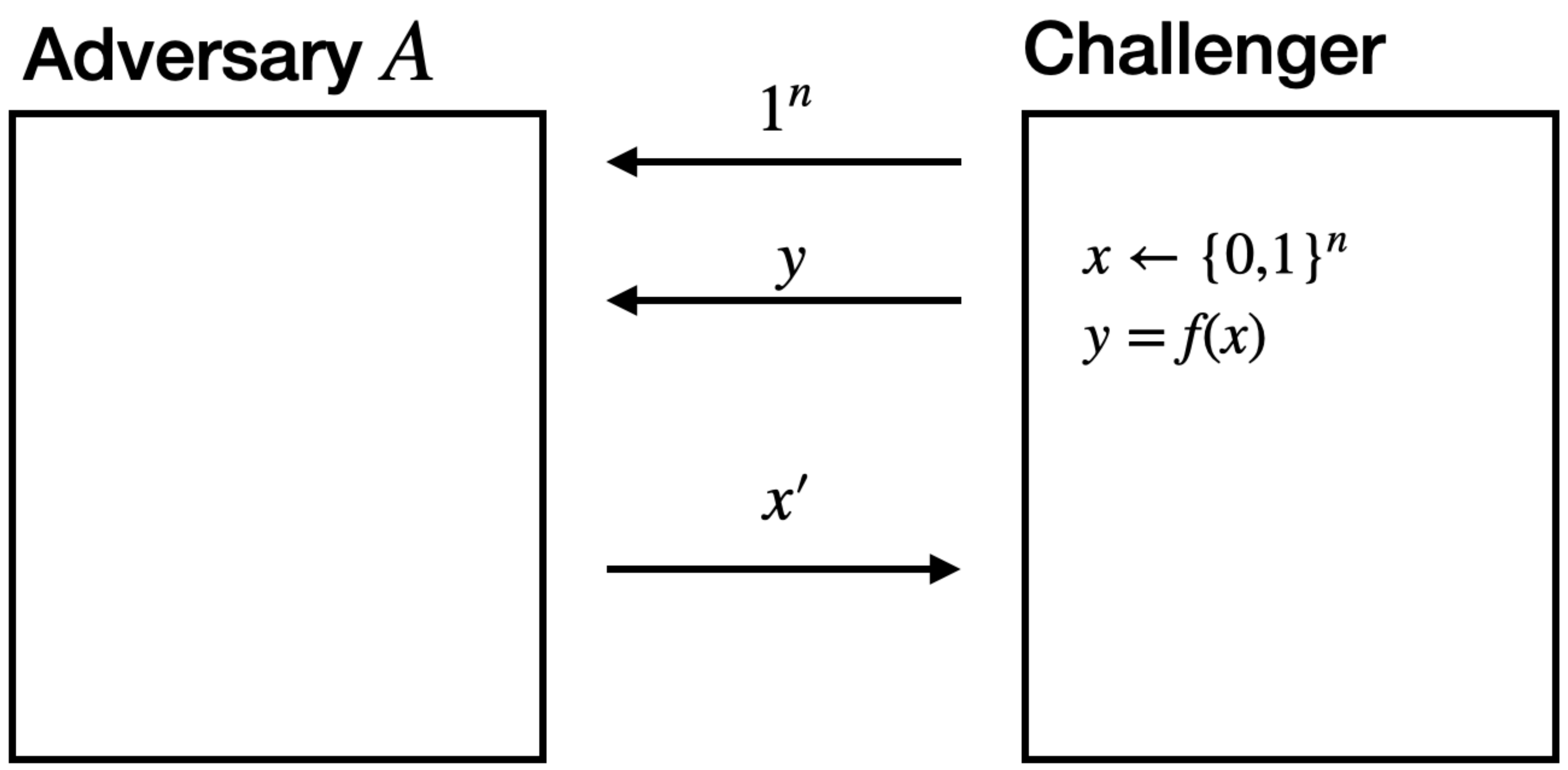
- A OWF does not need to hide all of its input!

- A OWF does not guarantee hardness of inverting *every* input

- If OWF exist, then $P \neq NP$

- OWF implies PRG

- If f is a OWF and a bijection, then f is a one-way permutation (OWP)



$\text{Invert}_{f,A}(n) = 1$ if $f(x') = y$
and 0 otherwise

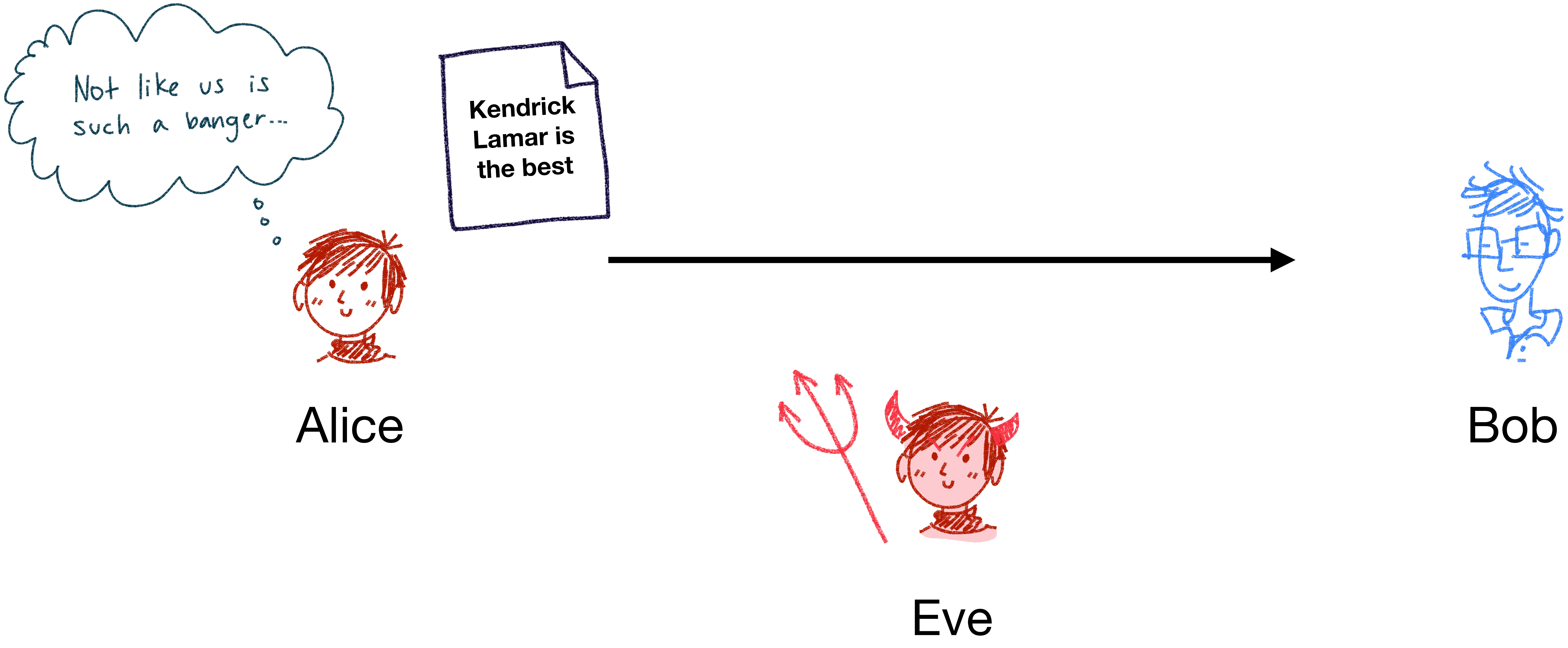
Message Authenticity

Message Authentication

So far we've focused on secrecy, but what about authenticity?

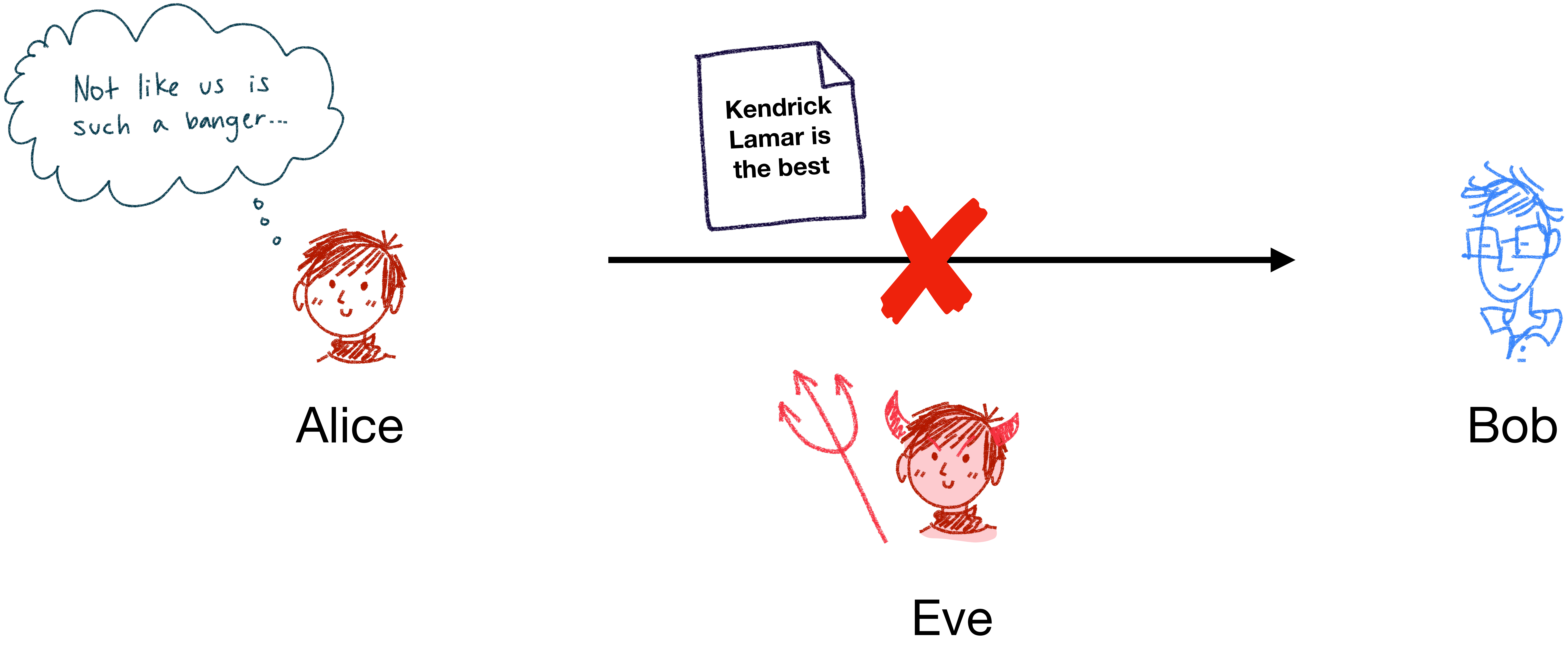
Message Authentication

So far we've focused on secrecy, but what about authenticity?



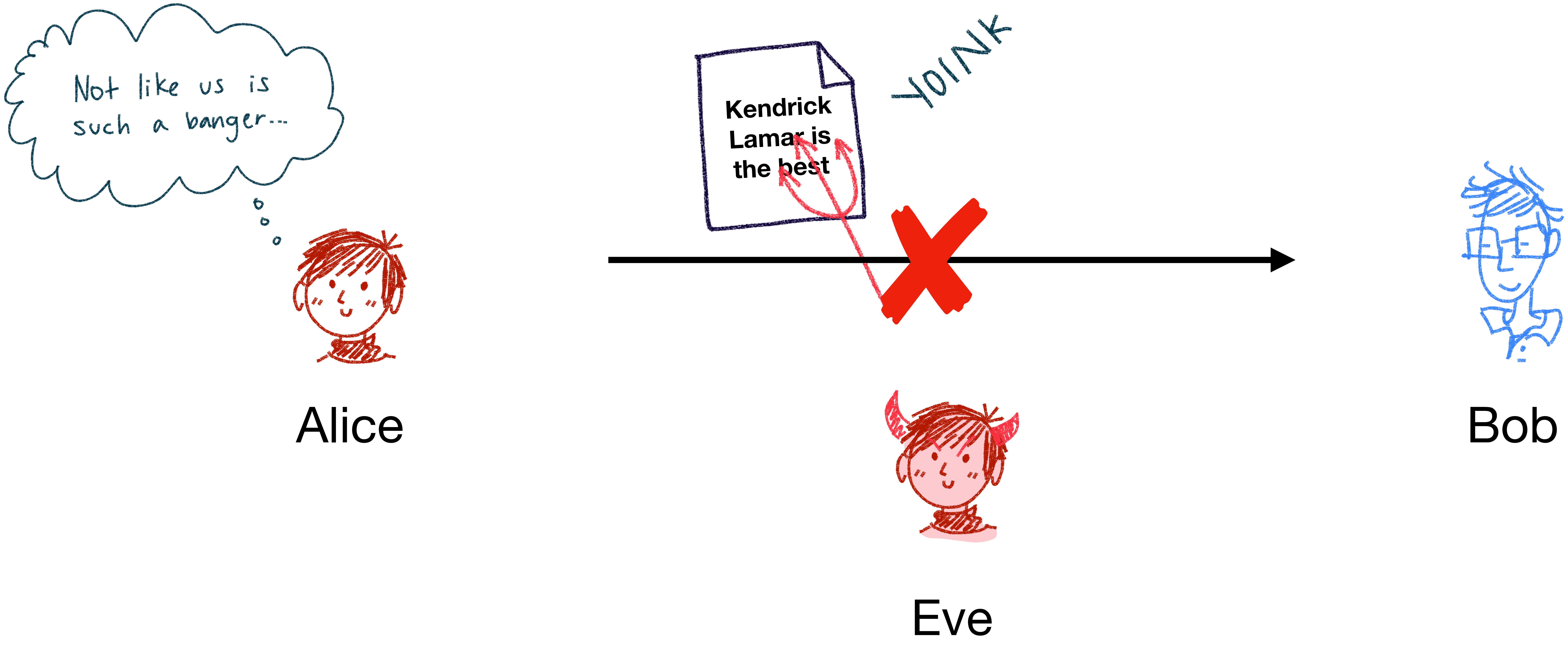
Message Authentication

So far we've focused on secrecy, but what about authenticity?



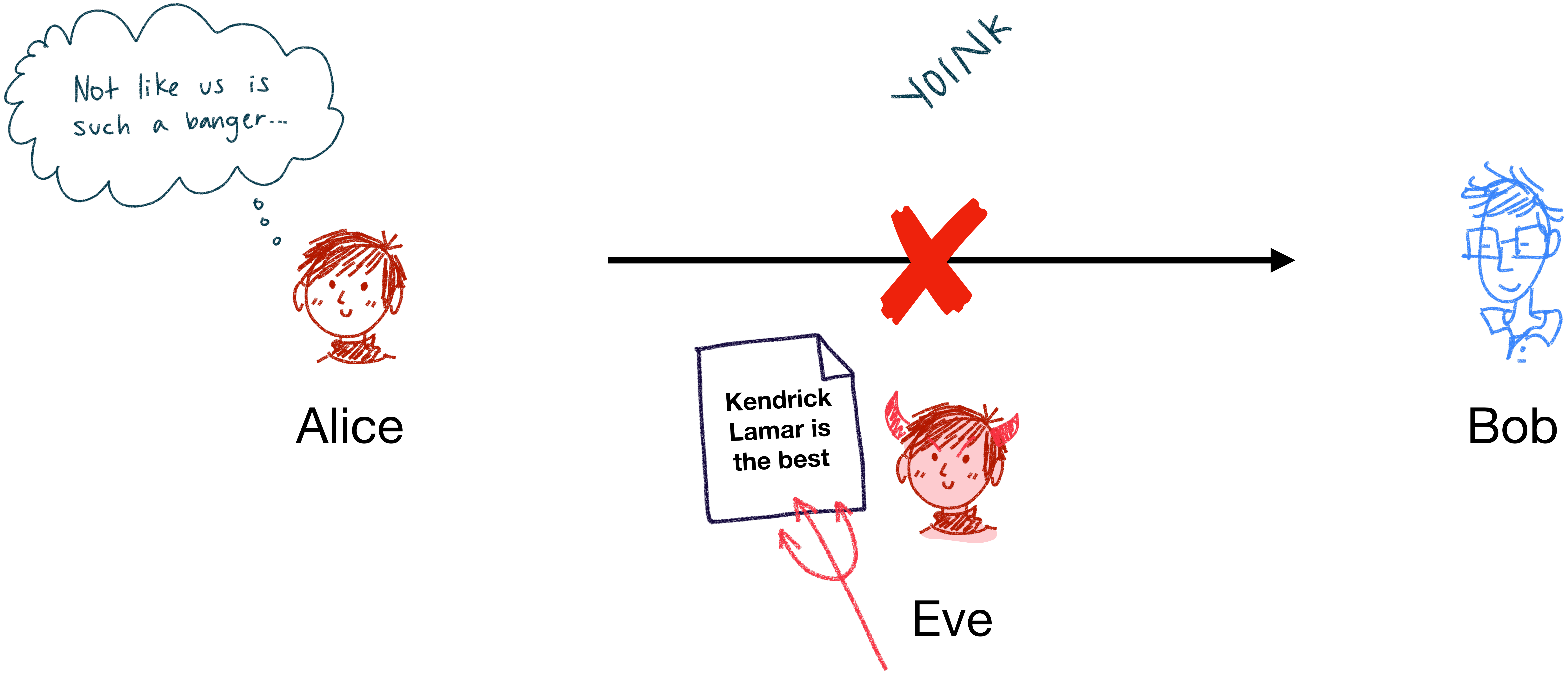
Message Authentication

So far we've focused on secrecy, but what about authenticity?



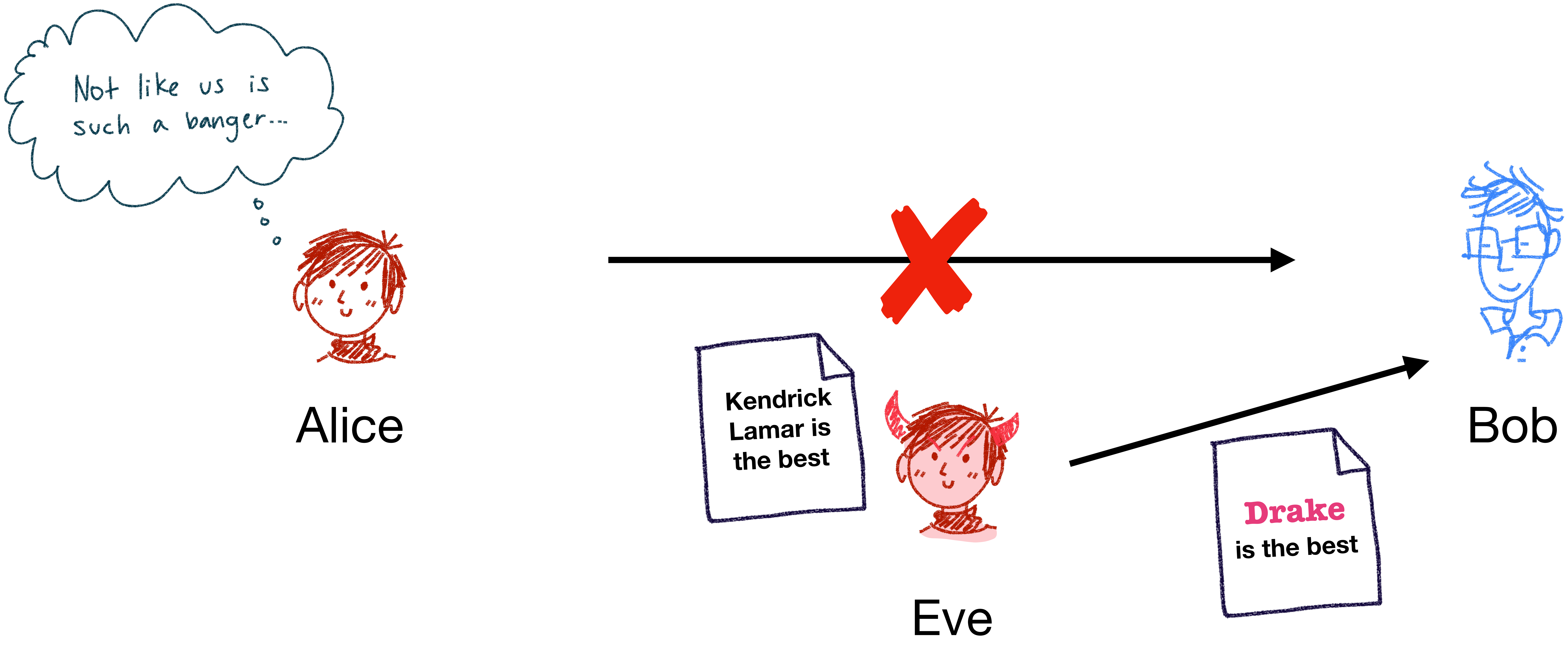
Message Authentication

So far we've focused on secrecy, but what about authenticity?



Message Authentication

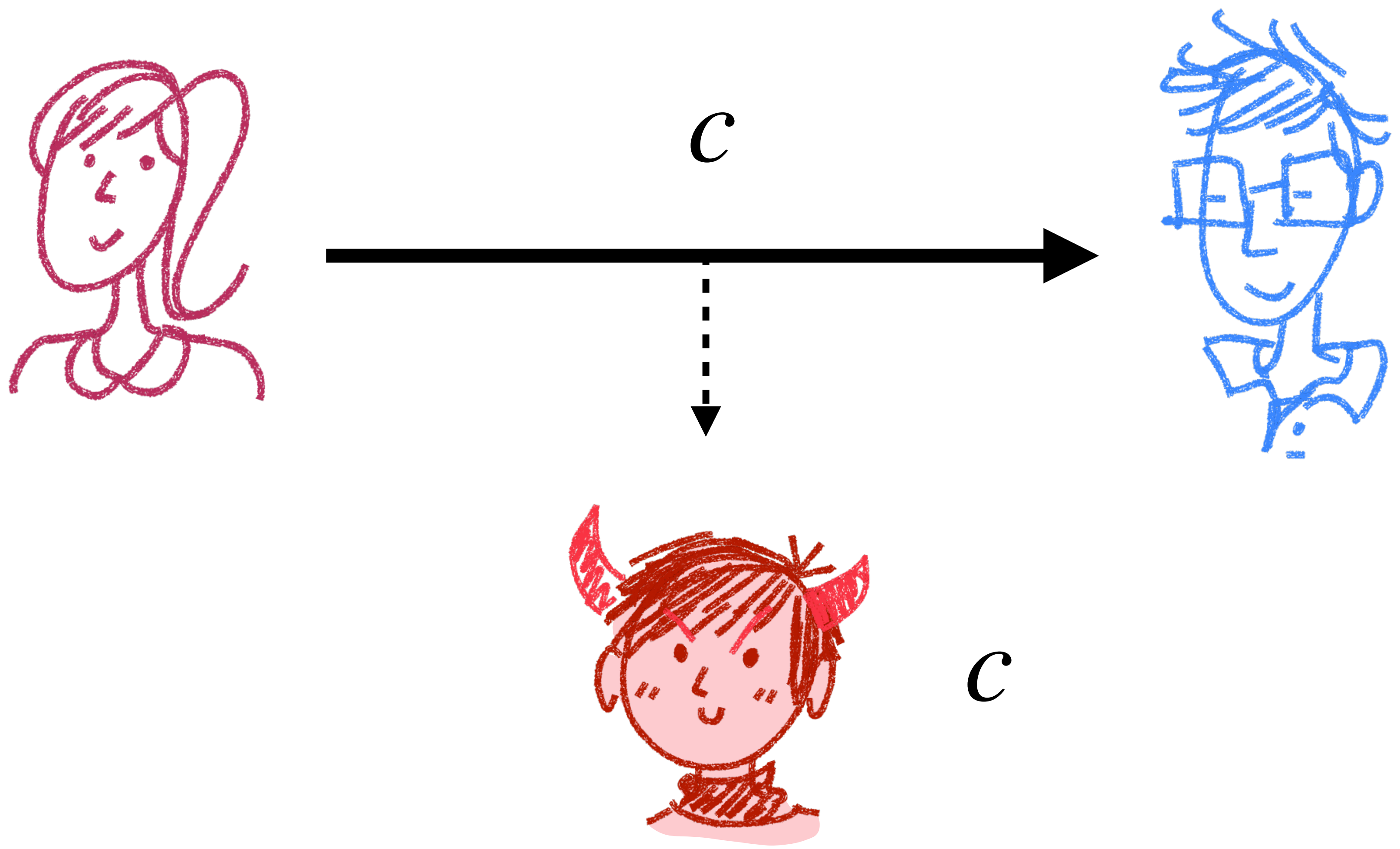
So far we've focused on secrecy, but what about authenticity?



Encryption vs Authentication

Orthogonal aspects; in general, one does not guarantee the other

- **Encryption** focuses on data **secrecy**
  - Hiding the contents of the message from an adversary



- **Authentication** focuses on data **integrity**
  - Assuring a receiver that a message has not been modified

Secrecy vs Authentication

Consider the CPA-secure encryption scheme we saw in Lecture 4:

- Gen(1^n): Sample $k \leftarrow \{0,1\}^n$
- Enc(k, m): On input $m \in \{0,1\}^{\ell_{\text{in}}}$, sample $r \leftarrow \{0,1\}^{\ell_{\text{in}}}$ and output

$$c = (r, F_k(r) \oplus m)$$

- Dec(k, c): On input $c = (c_1, c_2)$, output $F_k(c_1) \oplus c_2$

How might an adversary generate ciphertexts that look like this but do not originate from Alice?

Secrecy vs Authentication

Enc(k, m): On input $m \in \{0,1\}^{\ell_{\text{in}}}$, sample $r \leftarrow \{0,1\}^{\ell_{\text{in}}}$ and output

$$c = (r, F_k(r) \oplus m)$$

Dec(k, c): On input $c = (c_1, c_2)$, output $F_k(c_1) \oplus c_2$

- Adversary could send arbitrary (c_1, c_2)
  - It may not necessarily correspond to a meaningful m', but it's well-formed and will decrypt

Secrecy vs Authentication

Enc(k, m): On input $m \in \{0,1\}^{\ell_{\text{in}}}$, sample $r \leftarrow \{0,1\}^{\ell_{\text{in}}}$ and output

$$c = (r, F_k(r) \oplus m)$$

Dec(k, c): On input $c = (c_1, c_2)$, output $F_k(c_1) \oplus c_2$

- Adversary could send arbitrary (c_1, c_2)
  - It may not necessarily correspond to a meaningful m', but it's well-formed and will decrypt
- If the adversary sees a (c_1, c_2) corresponding to a *known* m, the adversary can modify it to be an encryption of *any* m' of its choice

Secrecy vs Authentication

Enc(k, m): On input $m \in \{0,1\}^{\ell_{\text{in}}}$, sample $r \leftarrow \{0,1\}^{\ell_{\text{in}}}$ and output

$$c = (r, F_k(r) \oplus m)$$

Dec(k, c): On input $c = (c_1, c_2)$, output $F_k(c_1) \oplus c_2$

- Adversary could send arbitrary (c_1, c_2)
  - It may not necessarily correspond to a meaningful m', but it's well-formed and will decrypt
- If the adversary sees a (c_1, c_2) corresponding to a *known* m, the adversary can modify it to be an encryption of *any* m' of its choice
- Replay attack: send the same ciphertext
  - Unavoidable for any scheme that can copy and resend, usually fixed on an application level with timestamps etc

Message Authentication Codes (MACs)

Syntax: Three algorithms $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$

- **Key generation** algorithm Gen takes input 1^n and outputs a key k
- **Tag generation** algorithm Mac takes a key k and a message $m \in \{0,1\}^*$ and outputs a tag $t \in \{0,1\}^*$
- **Verification** algorithm Verify takes a key k, a message m, a tag t, and outputs a bit b

m

$t \leftarrow \mathsf{Mac}_k(m)$

$$\xrightarrow{\quad (m, t) \quad}$$

$b = \mathsf{Verify}_k(m, t)$

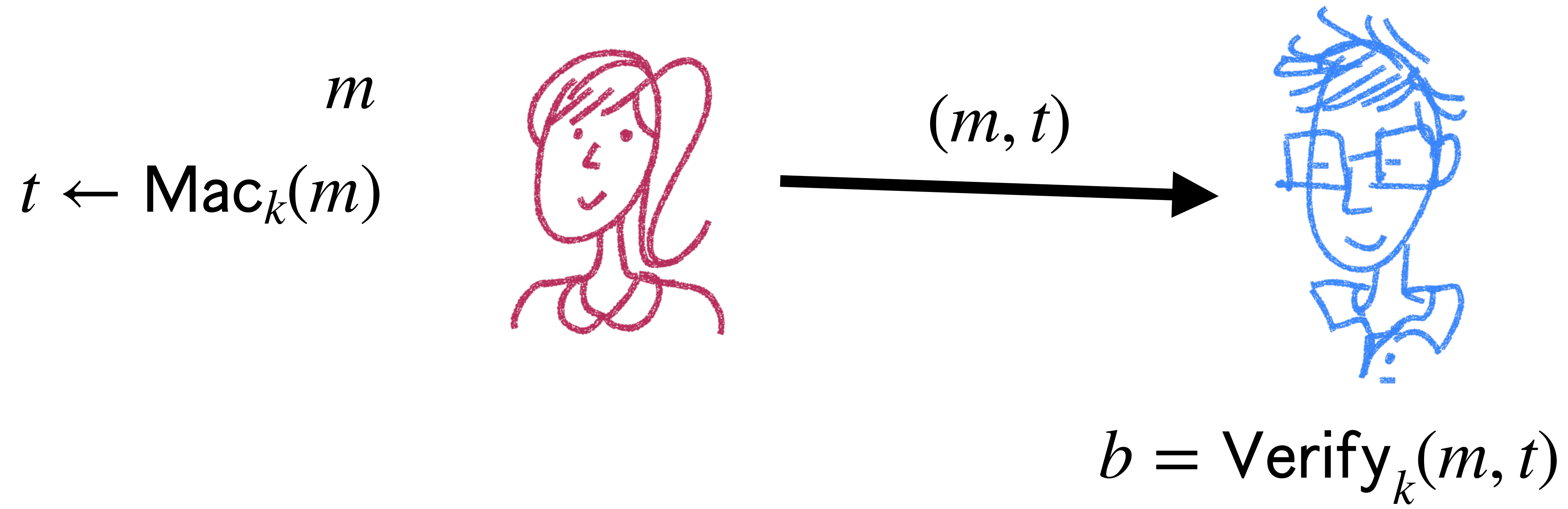
Message Authentication Codes (MACs)

Syntax: Three algorithms (Gen, Mac, Verify)

- **Key generation** algorithm Gen takes input 1^n and outputs a key k
- **Tag generation** algorithm Mac takes a key k and a message $m \in \{0,1\}^*$ and outputs a tag $t \in \{0,1\}^*$
- **Verification** algorithm Verify takes a key k, a message m, a tag t, and outputs a bit b

Correctness: $\forall n, \forall k$ output by Gen(1^n), $\forall m \in \{0,1\}^*$, $\forall t$ output by Mac$_k(m)$,

$$\text{Verify}_k(m, t) = 1$$



Message Authentication Codes (MACs)

Syntax: Three algorithms (Gen, Mac, Verify)

- **Key generation** algorithm Gen takes input 1^n and outputs a key k
- **Tag generation** algorithm Mac takes a key k and a message $m \in \{0,1\}^*$ and outputs a tag $t \in \{0,1\}^*$
- **Verification** algorithm Verify takes a key k, a message m, a tag t, and outputs a bit b

Correctness: $\forall n, \forall k$ output by Gen(1^n), $\forall m \in \{0,1\}^*$, $\forall t$ output by Mac$_k(m)$,

Verify$_k(m, t) = 1$

Canonical Verification:
If Mac algorithm is deterministic, then Verify just checks if Mac$_k(m) = t$

$b =$ Verify$_k(m, t)$

How should we define security?

m

$t \leftarrow \mathsf{Mac}_k(m)$

Alice

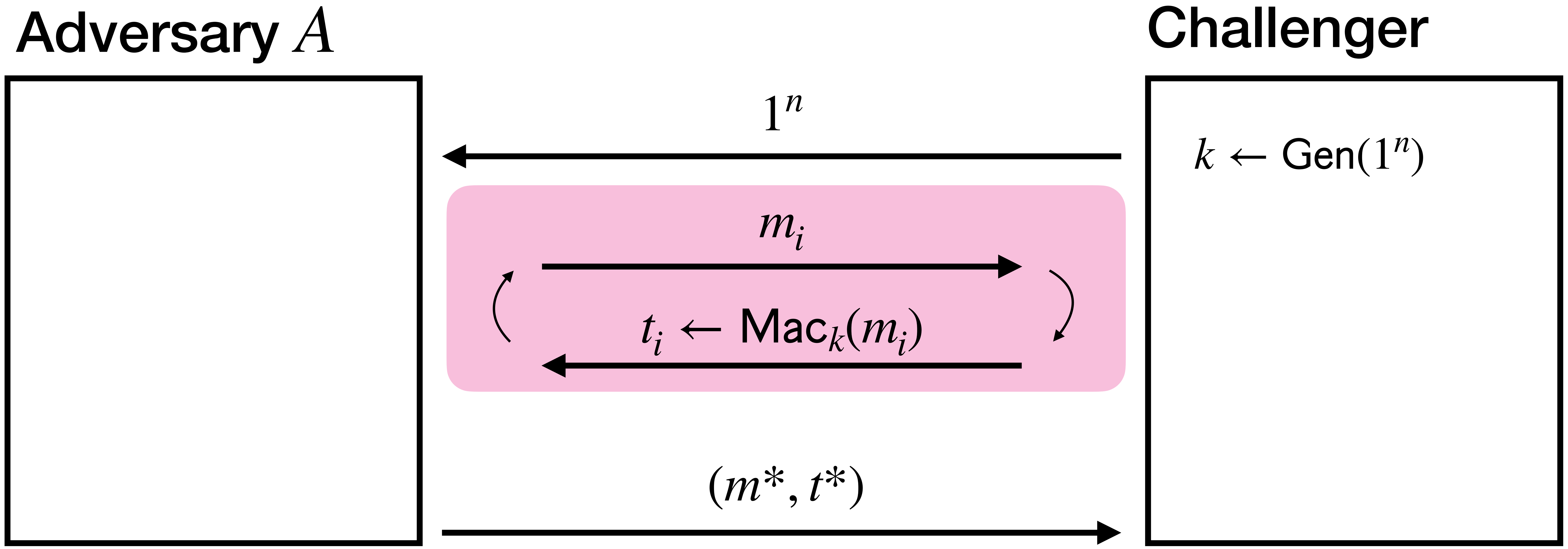
$b = \mathsf{Verify}_k(m, t)$

Bob

Eve

MAC Security

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

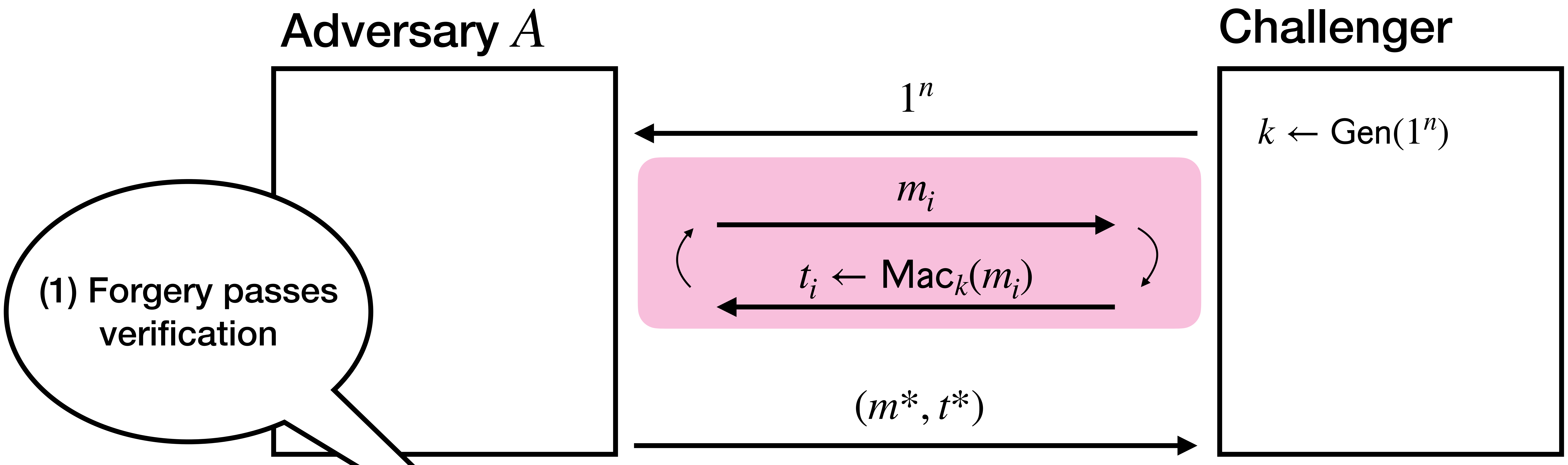


We say the adversary succeeds ($\mathsf{MacForge}_{\mathscr{A},\Pi}(n) = 1$) if:

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

MAC Security

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



We say the adversary succeeds (MacFor

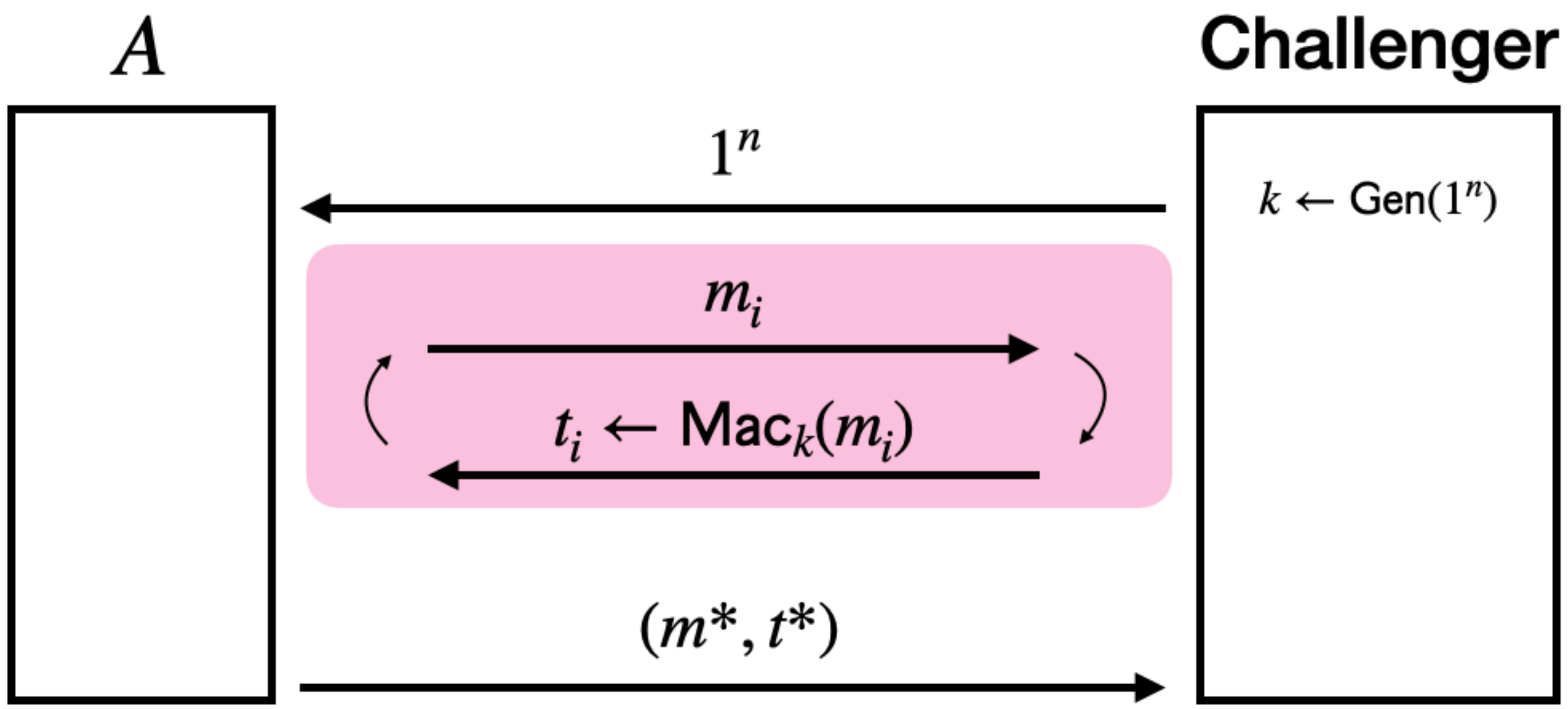
1. $\mathsf{Verify}_k(m^*, t^*) = 1$
2. $m^* \neq m_i$ for all queried m_i

(2) Forgery is on a new message

MAC Security

Definition: A MAC $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ is existentially unforgeable under an adaptive chosen-message attack if for every PPT adversary A there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr[\mathsf{MacForge}_{\mathscr{A},\Pi}(n) = 1] \leq \mathsf{negl}(n)$$



$\mathsf{MacForge}_{\mathscr{A},\Pi}(n) = 1$ if:

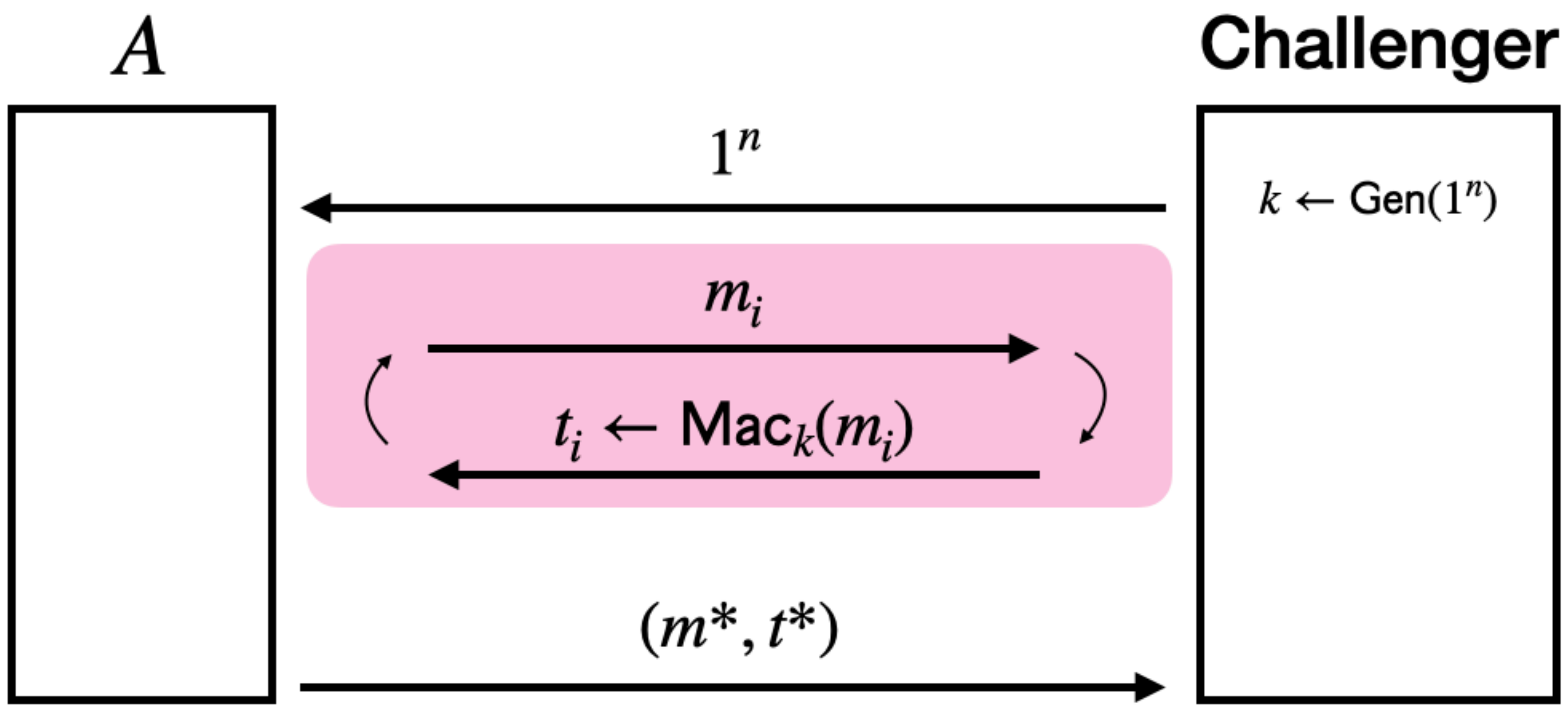
1. $\mathsf{Verify}_k(m^*, t^*) = 1$
2. $m^* \neq m_i$ for all queried m_i

And is 0 otherwise

Strong MAC Security

Definition: A MAC $\Pi = (\text{Gen}, \text{Mac}, \text{Verify})$ is strongly secure if for every PPT adversary A there exists a negligible function $\text{negl}(\cdot)$ such that

$$\Pr[\text{MacSForge}_{\mathcal{A},\Pi}(n) = 1] \leq \text{negl}(n)$$



$\text{MacSForge}_{\mathcal{A},\Pi}(n) = 1$ if:

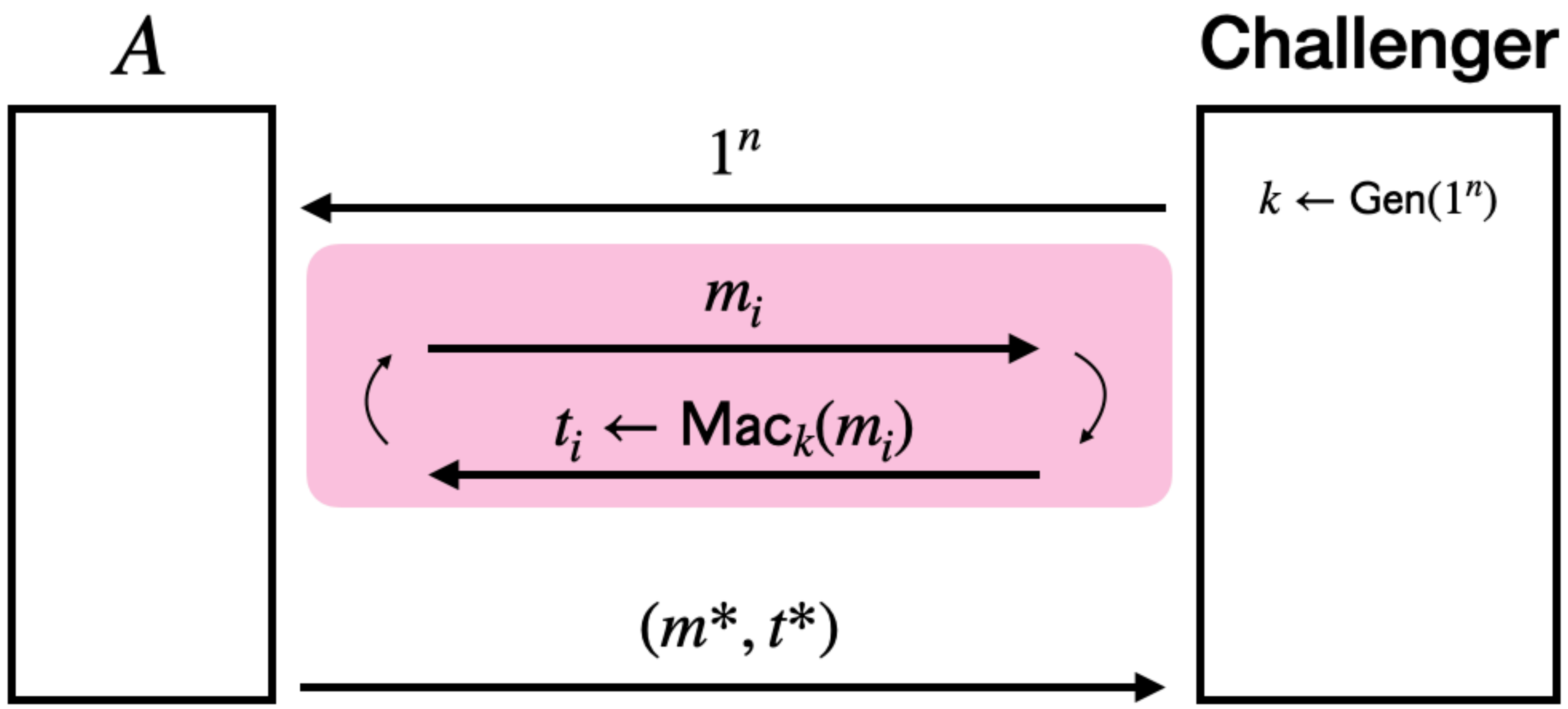
1. $\text{Verify}_k(m^*, t^*) = 1$
2. $(m^*, t^*) \neq (m_i, t_i)$ for all queried m_i and response t_i

And is 0 otherwise

Strong MAC Security

Definition: A MAC $\Pi = (\text{Gen}, \text{Mac}, \text{Verify})$ is strongly secure if for every PPT adversary A there exists a negligible function $\text{negl}(\cdot)$ such that

$$\Pr[\text{MacSForge}_{\mathscr{A},\Pi}(n) = 1] \leq \text{negl}(n)$$



$\text{MacSForge}_{\mathscr{A},\Pi}(n) = 1$ if:

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

And is 0 otherwise

Theorem: A secure MAC that uses canonical verification is a strong MAC

Fixed-Length MAC

A Fixed-Length MAC

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a PRF

We can construct a MAC as follows:

- **Key generation:** On input 1^n, output a randomly sampled $k \leftarrow \{0,1\}^n$
- **Tag generation:** On input $k \in \{0,1\}^n$ and $m \in \{0,1\}^n$, output $t = F_k(m)$
- **Verification:** On input $k \in \{0,1\}^n$, $m \in \{0,1\}^n$, and $t \in \{0,1\}^n$, output 1 if $t = F_k(m)$ and 0 otherwise

A Fixed-Length MAC

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a PRF

We can construct a MAC as follows:

- **Key generation:** On input 1^n, output a randomly sampled $k \leftarrow \{0,1\}^n$
- **Tag generation:** On input $k \in \{0,1\}^n$ and $m \in \{0,1\}^n$, output $t = F_k(m)$
- **Verification:** On input $k \in \{0,1\}^n$, $m \in \{0,1\}^n$, and $t \in \{0,1\}^n$, output 1 if $t = F_k(m)$ and 0 otherwise

Theorem: If F is a PRF, then the above MAC scheme is secure for messages of length n

A Fixed-Length MAC

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a PRF

- **Key generation:** On input 1^n, output a randomly sampled $k \leftarrow \{0,1\}^n$
- **Tag generation:** On input $k \in \{0,1\}^n$ and $m \in \{0,1\}^n$, output $t = F_k(m)$
- **Verification:** On input $k \in \{0,1\}^n$, $m \in \{0,1\}^n$, and $t \in \{0,1\}^n$, output 1 if $t = F_k(m)$ and 0 otherwise

Theorem: If F is a PRF, then the above MAC scheme is secure for messages of length n

Proof idea:

- Given a forger for the MAC, construct a distinguisher for the PRF.

- The distinguisher has oracle access to a function which is either a PRF or a truly random function.

- Distinguisher guesses "PRF" if the forger is able to produce a valid forgery. Otherwise, distinguisher guesses "random"

Arbitrary-Length MACs

Authenticating Arbitrary-Length Messages

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

Suppose we had a $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ for fixed-length messages.

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

Authenticating Arbitrary-Length Messages

$m \quad =$

m_1	m_2		$\dots$		$\dots$		m_d

Suppose we had a $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ for fixed-length messages.

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

Idea 1: Authenticate each block on its own

$$\hat{\mathsf{Mac}}_k(m_1 || m_2 || \dots || m_d) = \mathsf{Mac}_k(m_1) || \dots || \mathsf{Mac}_k(m_d)$$

Authenticating Arbitrary-Length Messages

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

Suppose we had a $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Verify})$ for fixed-length messages.

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

Idea 2: Authenticate each block on its own with an index?

$$\hat{\mathsf{Mac}}_k(m_1 || m_2 || \ldots || m_d) = \mathsf{Mac}_k(1 || m_1) || \ldots || \mathsf{Mac}_k(d || m_d)$$

Next Time

- Today:
  - OWFs review
  - MACs
- Wednesday
  - Arbitrary-Length MACs
  - CCA-Security