CMSC/Math 456: Cryptography (Fall 2022)

Lecture 17

Daniel Gottesman

Administrative

This week's problem set will be out by early next week, maybe earlier, and due two weeks from today (Thursday, Nov. 9).

Next week I will be away, and there will be guest lectures by Gorjan Alagic on quantum and post-quantum cryptography. This material is part of the course content and will potentially be on the final exam, but only qualitative aspects of it.

I will not have an office hour next Tuesday, but Mahathi's office hour will be on as usual unless she announces otherwise.

Message Authenticity

Suppose you receive this e-mail:

From: Daniel Eric Gottesman <dgottesm@umd.edu>

Subject: Important assignment

Please review the material on this website today.

Daniel Gottesman

What do you do?

Message Authenticity

Suppose you receive this e-mail:

From: Daniel Eric Gottesman <dgottesm@umd.edu>

Subject: Important assignment

Please review the material on this website today.

Daniel Gottesman

What do you do?

It seems a bit suspicious. What if you can't reach me to ask if it is real or not?

Message Authenticity

Suppose you receive this e-mail:

> From: Daniel Eric Gottesman <dgottesm@umd.edu>
>
> Subject: Important assignment
>
> Please review the material on this website today.
>
> Daniel Gottesman

What do you do?

It seems a bit suspicious. What if you can't reach me to ask if it is real or not?

And if I sent a lot of e-mails like that, it might not even look suspicious.

Message Authentication

Message authentication is a cryptographic solution to this problem: It lets you verify that the message really came from me.

From: Daniel Eric Gottesman <dgottesm@umd.edu>

Subject: Important assignment

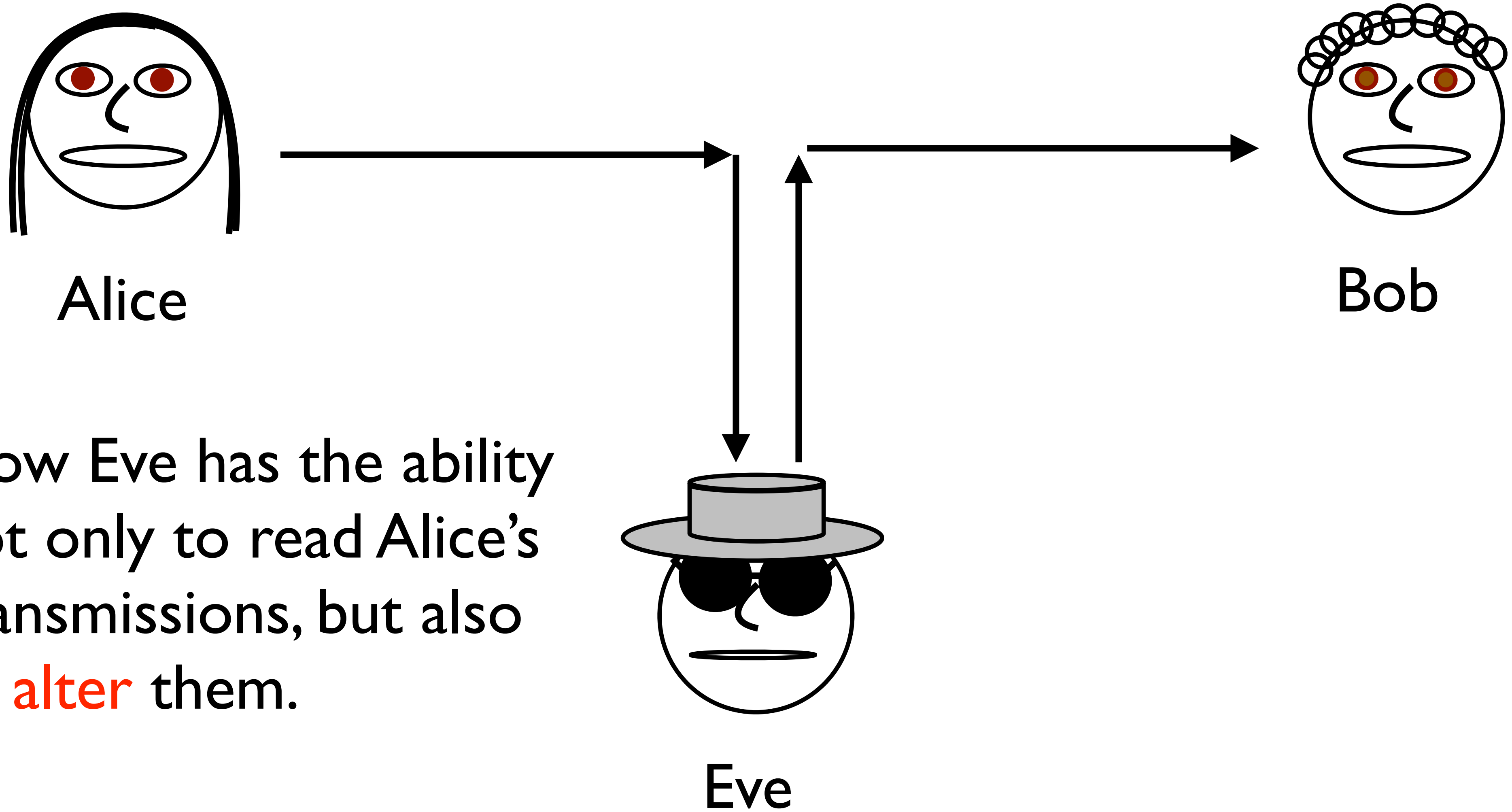
Please review the material on this website today.

Daniel Gottesman

What if the message really did come from me … but it's been altered, perhaps by changing the link? Even a one-character change would point you to a different website which could contain malware. Asking me won't help, but message authentication addresses this too.

New Threat Model

In the threat model for message authentication, Eve is able to change messages sent between Alice and Bob and wants Bob to accept a message that Alice didn't send.



Now Eve has the ability not only to read Alice's transmissions, but also to alter them.

This class is being recorded

Encryption Doesn't Help

Note that encryption by itself doesn't solve the problem. Eve can still change the message even if she can't read it.

One-time pad:

Key: 00101000101110101010

Message: 10111110010011001100

Ciphertext: 10010110111101100110

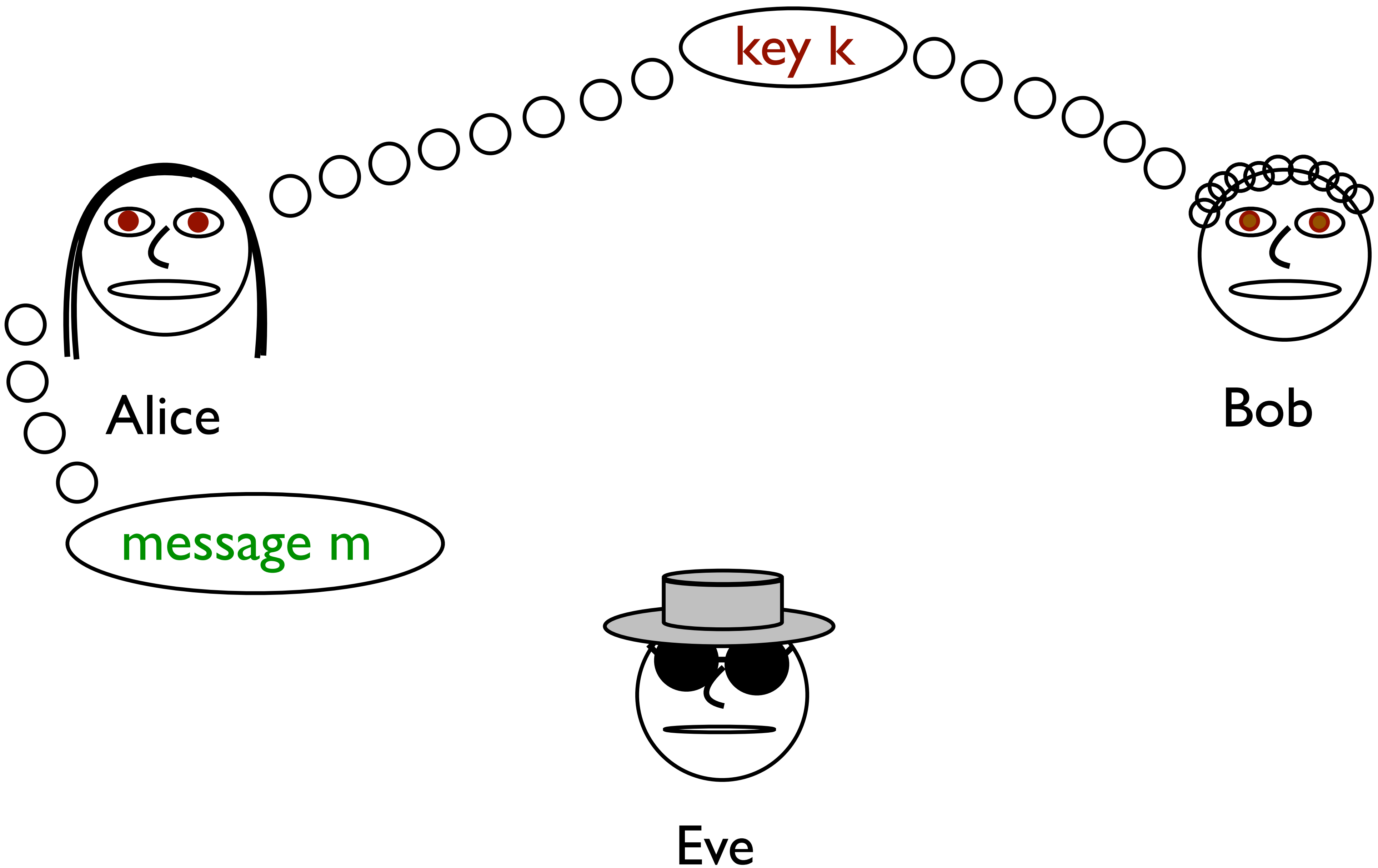
Alteration: 10010110111100100110

Decryption: 10111110010010001100

Changing the ciphertext produces a predictable change in the decrypted plaintext. (The one-time-pad is malleable.)

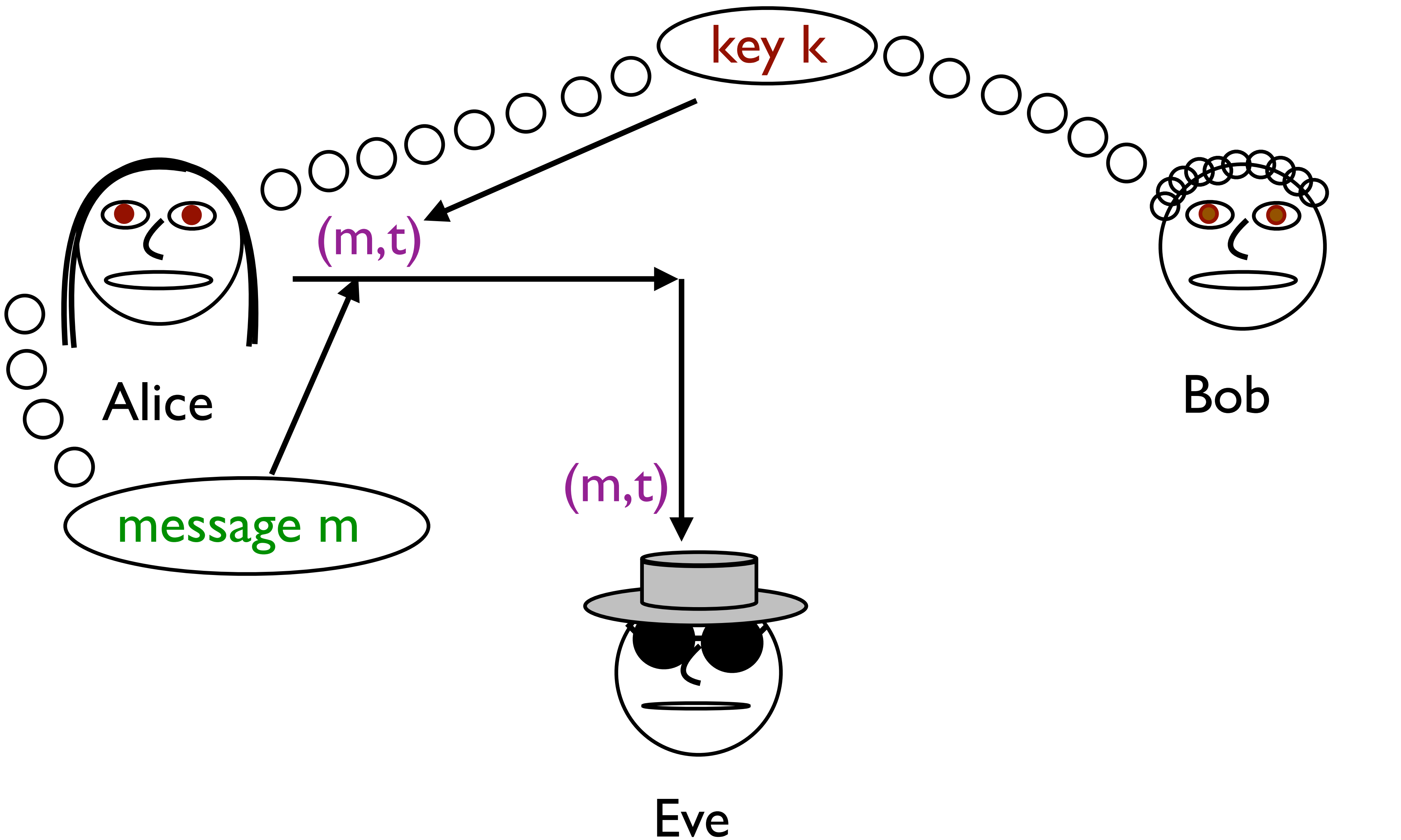
Message Authentication Code

Encryption is also not necessary. Instead, Alice appends a tag to her message to prove its authenticity. Alice and Bob share a secret key to give them an advantage over Eve.



Message Authentication Code

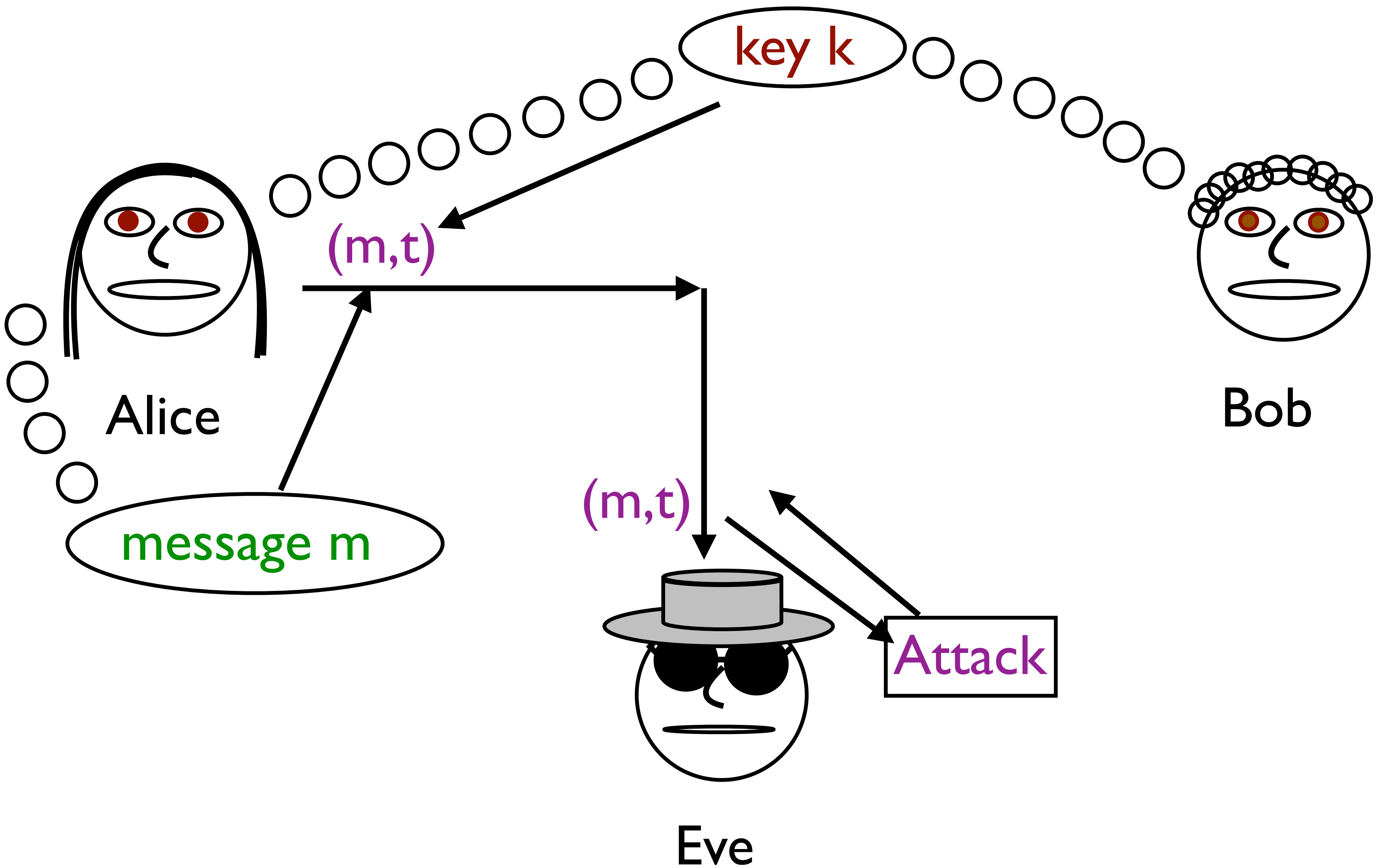
Encryption is also not necessary. Instead, Alice appends a tag to her message to prove its authenticity. Alice and Bob share a secret key to give them an advantage over Eve.



This class is being recorded

Message Authentication Code

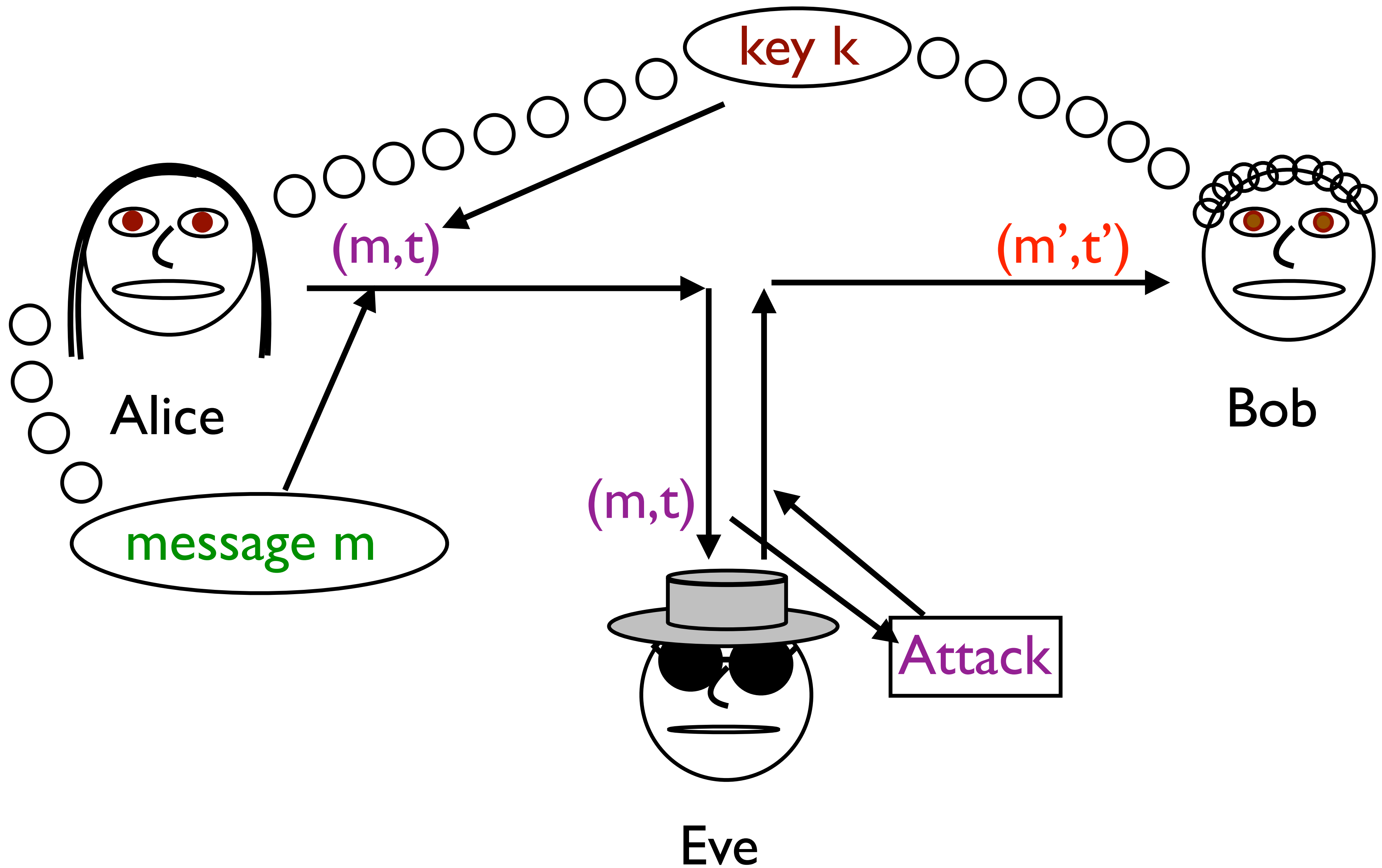
Encryption is also not necessary. Instead, Alice appends a tag to her message to prove its authenticity. Alice and Bob share a secret key to give them an advantage over Eve.



This class is being recorded

Message Authentication Code

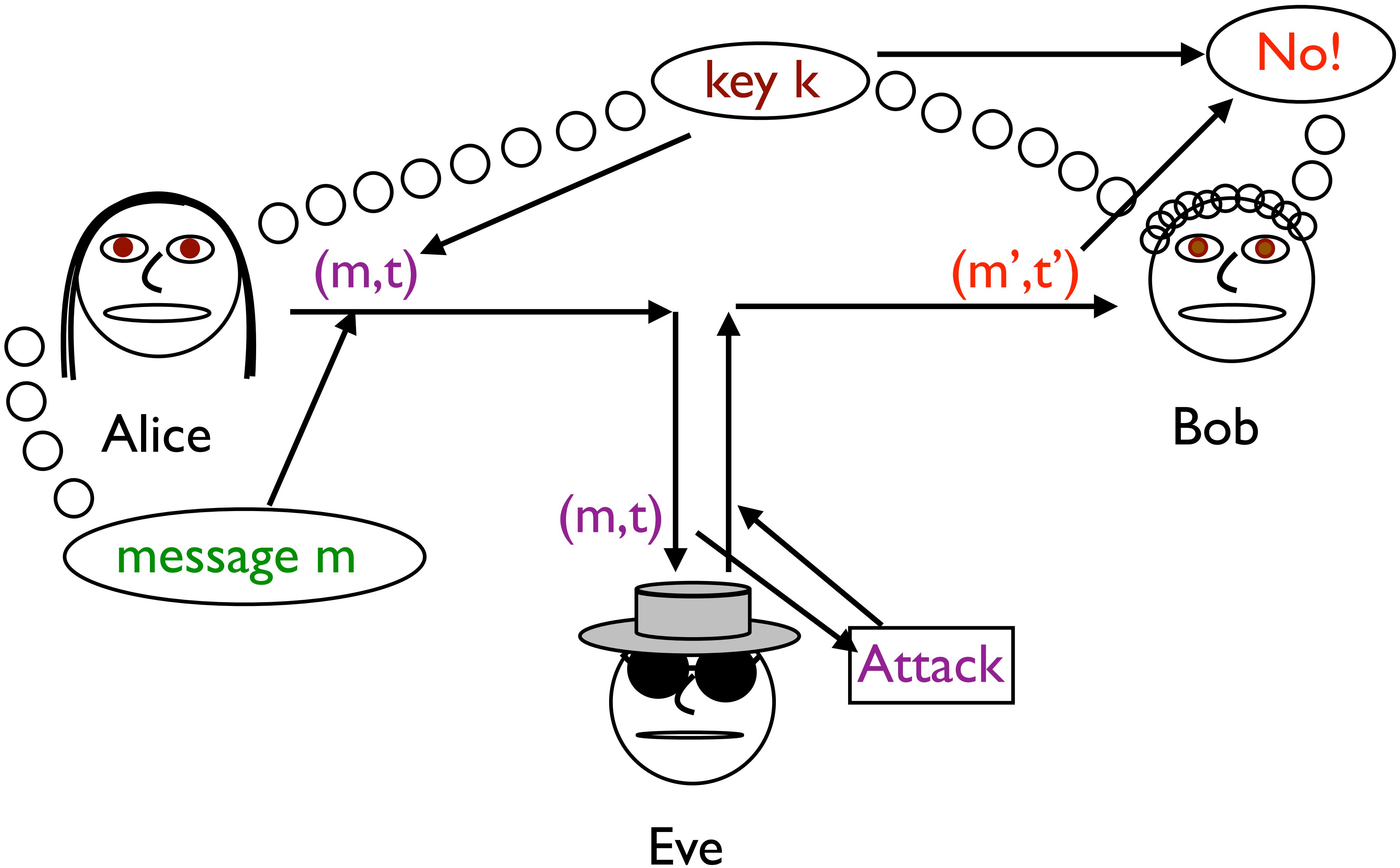
Encryption is also not necessary. Instead, Alice appends a tag to her message to prove its authenticity. Alice and Bob share a secret key to give them an advantage over Eve.



This class is being recorded

Message Authentication Code

Encryption is also not necessary. Instead, Alice appends a tag to her message to prove its authenticity. Alice and Bob share a secret key to give them an advantage over Eve.



This class is being recorded

MAC Definition

Definition: A message authentication code (MAC) is a set of three probabilistic polynomial-time algorithms (Gen, Mac, Vrfy):

Gen is the key generation algorithm. It takes as input s, the security parameter, and outputs a private key $k \in \{0,1\}^*$ of length poly(s).

Mac is the tag-generation algorithm. It takes as input k and a message $m \in \{0,1\}^*$ and outputs a tag $t \in \{0,1\}^*$.

Vrfy is the verification algorithm. It takes as input k and (m,t) and outputs "valid" or "invalid."

MAC Definition

Definition: A message authentication code (MAC) is a set of three probabilistic polynomial-time algorithms (Gen, Mac, Vrfy):

Gen is the key generation algorithm. It takes as input s, the security parameter, and outputs a private key $k \in \{0,1\}^*$ of length poly(s).

Mac is the tag-generation algorithm. It takes as input k and a message $m \in \{0,1\}^*$ and outputs a tag $t \in \{0,1\}^*$.

Vrfy is the verification algorithm. It takes as input k and (m,t) and outputs "valid" or "invalid."

The MAC is correct if

$$\mathrm{Vrfy}(k, m, \mathrm{Mac}(k, m)) = \mathrm{valid}$$

MAC Definition

Definition: A message authentication code (MAC) is a set of three probabilistic polynomial-time algorithms (Gen, Mac, Vrfy):

Gen is the key generation algorithm. It takes as input s, the security parameter, and outputs a private key $k \in \{0,1\}^*$ of length poly(s).

Mac is the tag-generation algorithm. It takes as input k and a message $m \in \{0,1\}^*$ and outputs a tag $t \in \{0,1\}^*$.

Vrfy is the verification algorithm. It takes as input k and (m,t) and outputs "valid" or "invalid."

The MAC is correct if

$$\mathrm{Vrfy}(k, m, \mathrm{Mac}(k, m)) = \mathrm{valid}$$

Often Vrfy just runs Mac(k,m) to get a tag t' and outputs "valid" if t=t'.

Private Key vs. Public Key

A MAC is a *private key* protocol. That means that Alice and Bob must share the private key and can only use it to authenticate messages to each other.

If you want a *public key* authentication protocol, use a digital signature, which is basically a public key version of a MAC. However, they have some useful additional properties by virtue of being public key.

We will discuss digital signatures later in the course.

First Try

Suppose we let t be the parity of the bits in the message m (i.e., the XOR of all bits, 0 if m has an even number of 1s).

m = 001101011 ⇨ t=1

Vote: Does this work? (Yes/no/unknown)

First Try

Suppose we let t be the parity of the bits in the message m (i.e., the XOR of all bits, 0 if m has an even number of 1s).

m = 001101011 ⇒ t=1

Vote: Does this work? (Yes/no/unknown)

Answer: No. It has a number of problems.

First Try

Suppose we let t be the parity of the bits in the message m (i.e., the XOR of all bits, 0 if m has an even number of 1s).

m = 001101011 → t=1

Vote: Does this work? (Yes/no/unknown)

Answer: No. It has a number of problems.

- Eve knows the procedure and can easily make her own tags.

E.g.: m = 000000000, t=0

We need to involve a key somehow.

Second Try

What if we instead have a key k and the tag is $t = m \cdot k$?

k = 100101100

m = 001101011

→ t=0

Vote: Does this work? (Yes/no/unknown)

Second Try

What if we instead have a key k and the tag is $t = m \cdot k$?

k = 100101100

m = 001101011

t=0

Vote: Does this work? (Yes/no/unknown)

Answer: Still no.

Second Try

What if we instead have a key k and the tag is $t = m \cdot k$?

k = 100101100
m = 001101011 → t=0

Vote: Does this work? (Yes/no/unknown)

Answer: Still no.

- The tag space is too small. Eve can just guess the tag with 50% chance of success.
- The all 0's message always has the same tag, so Eve can forge that:

  m = 000000000, t=0

- With multiple messages, Eve can quickly determine k and then she can forge messages easily.

Pseudorandom Functions

What we would really want is that the tag t = Mac(k,m) is a random string unrelated to t' = Mac(k,m'). That way, seeing some tags won't help in forging different messages.

That means we want a random function for Mac.

But random functions need a long key and can't be computed efficiently. So we will use the next-best thing: a pseudorandom function.

$\mathrm{Mac}(k, m) = F_k(m)$ for pseudorandom $F_k(m)$.

E.g.: $F_k(m)$ could be AES.

Note that we don't need a pseudorandom *permutation* here, only a pseudorandom *function*. Mac does not need to be invertible.

Security of MACs

What, precisely, does it mean for a MAC to be secure?

Security of MACs

What, precisely, does it mean for a MAC to be secure?

- Eve should be unable, except with negligible probability, to produce a valid pair (m,t).

Security of MACs

What, precisely, does it mean for a MAC to be secure?

- Eve should be unable, except with negligible probability, to produce a valid pair (m,t).
- Even if she has seen a valid message pair (m_0, t_0).

Security of MACs

What, precisely, does it mean for a MAC to be secure?

- Eve should be unable, except with negligible probability, to produce a valid pair (m,t).
- Even if she has seen a valid message pair (m_0, t_0).
- Even if she has seen *many* valid message pairs (m_i, t_i).

Security of MACs

What, precisely, does it mean for a MAC to be secure?

- Eve should be unable, except with negligible probability, to produce a valid pair (m,t).
- Even if she has seen a valid message pair (m_0, t_0).
- Even if she has seen *many* valid message pairs (m_i, t_i).
- Actually, we should probably let Eve choose which messages m_i she's seen authenticated.

Security of MACs

What, precisely, does it mean for a MAC to be secure?

- Eve should be unable, except with negligible probability, to produce a valid pair (m,t).
- Even if she has seen a valid message pair (m_0, t_0).
- Even if she has seen *many* valid message pairs (m_i, t_i).
- Actually, we should probably let Eve choose which messages m_i she's seen authenticated.
- Except we can't let her have a valid tag for the message m.

Security of MACs

What, precisely, does it mean for a MAC to be secure?

- Eve should be unable, except with negligible probability, to produce a valid pair (m,t).
- Even if she has seen a valid message pair (m_0, t_0).
- Even if she has seen *many* valid message pairs (m_i, t_i).
- Actually, we should probably let Eve choose which messages m_i she's seen authenticated.
- Except we can't let her have a valid tag for the message m.
- But we let her pick the message m after seeing tags for the other messages.

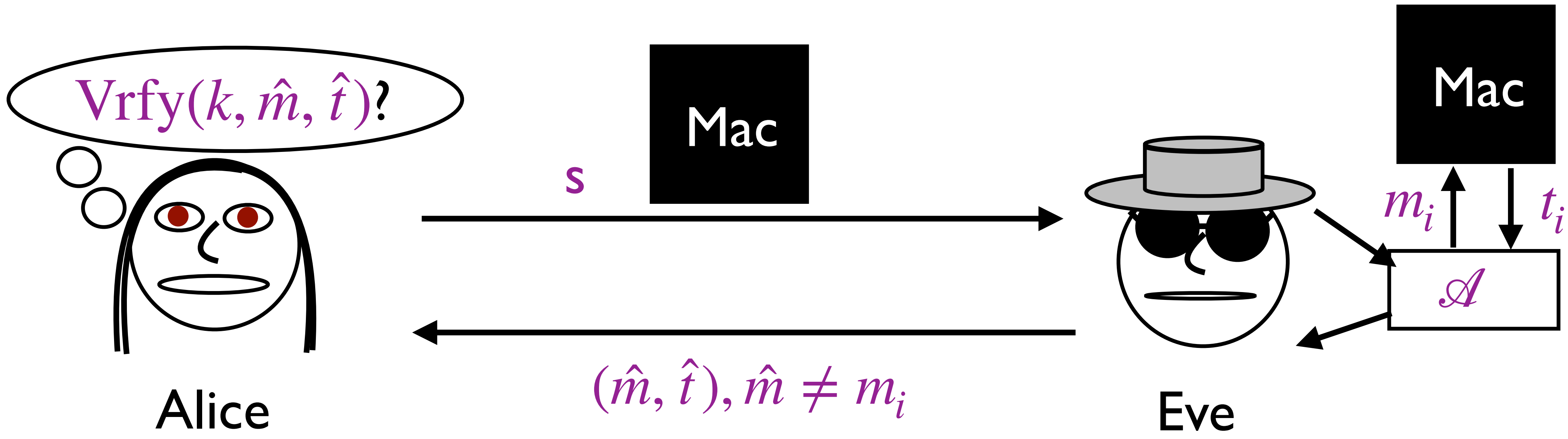
Security of MACs

What, precisely, does it mean for a MAC to be secure?

- Eve should be unable, except with negligible probability, to produce a valid pair (m,t).
- Even if she has seen a valid message pair (m_0, t_0).
- Even if she has seen *many* valid message pairs (m_i, t_i).
- Actually, we should probably let Eve choose which messages m_i she's seen authenticated.
- Except we can't let her have a valid tag for the message m.
- But we let her pick the message m after seeing tags for the other messages.

Time for another game!

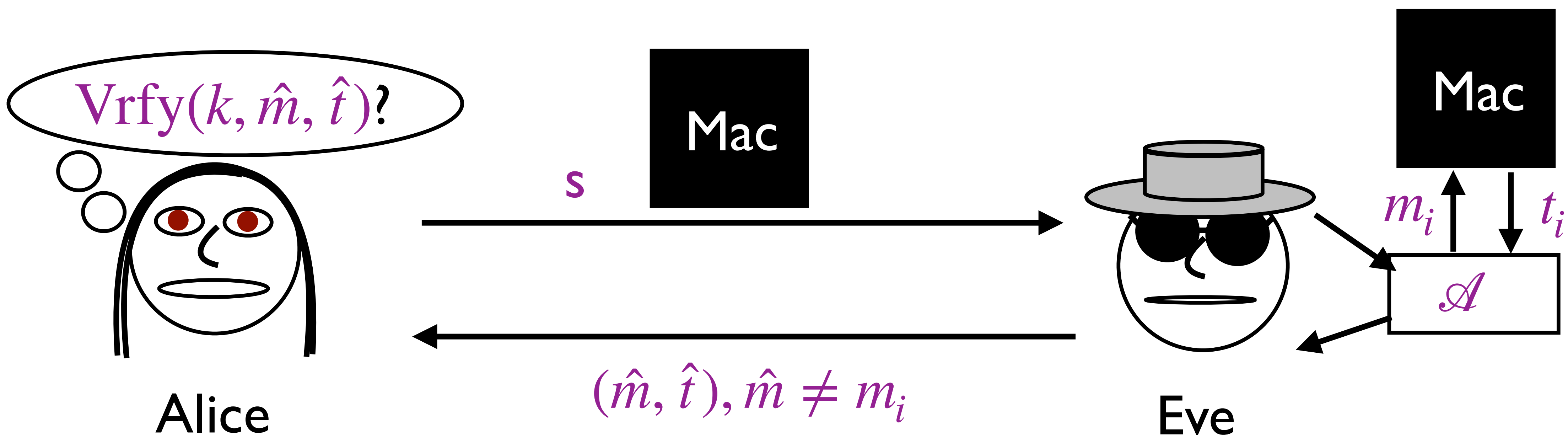
MAC Security Game



In this game:

This class is being recorded

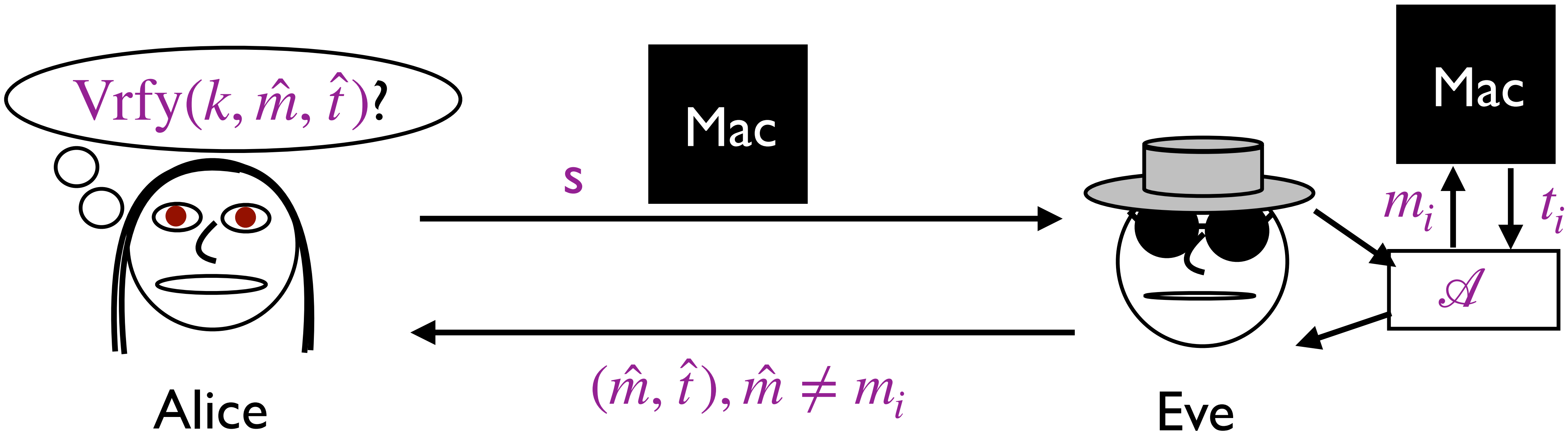
MAC Security Game



In this game:

1. Alice gives Eve access to an oracle that Eve can query with possible messages m_i and receive valid tags $t_i = \mathrm{Mac}(k, m_i)$ for m_i using the key k.

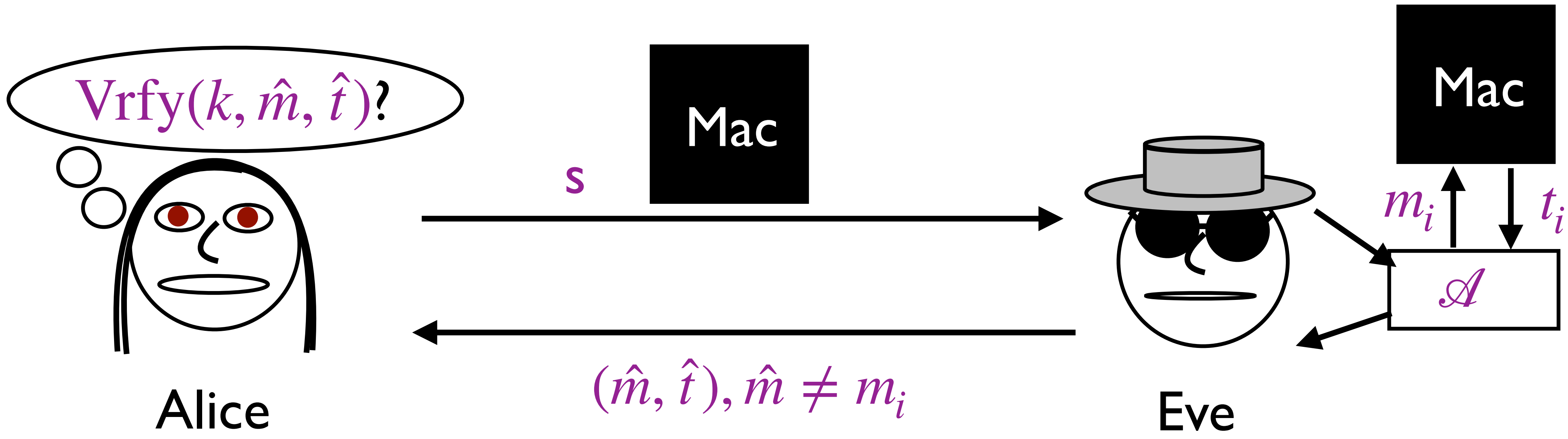
MAC Security Game



In this game:

1. Alice gives Eve access to an oracle that Eve can query with possible messages m_i and receive valid tags $t_i = \mathrm{Mac}(k, m_i)$ for m_i using the key k.
2. Eve runs a polynomial-size attack using the oracle access and generates a (message, tag) pair $(\hat{m}, \hat{t})$.

MAC Security Game



In this game:

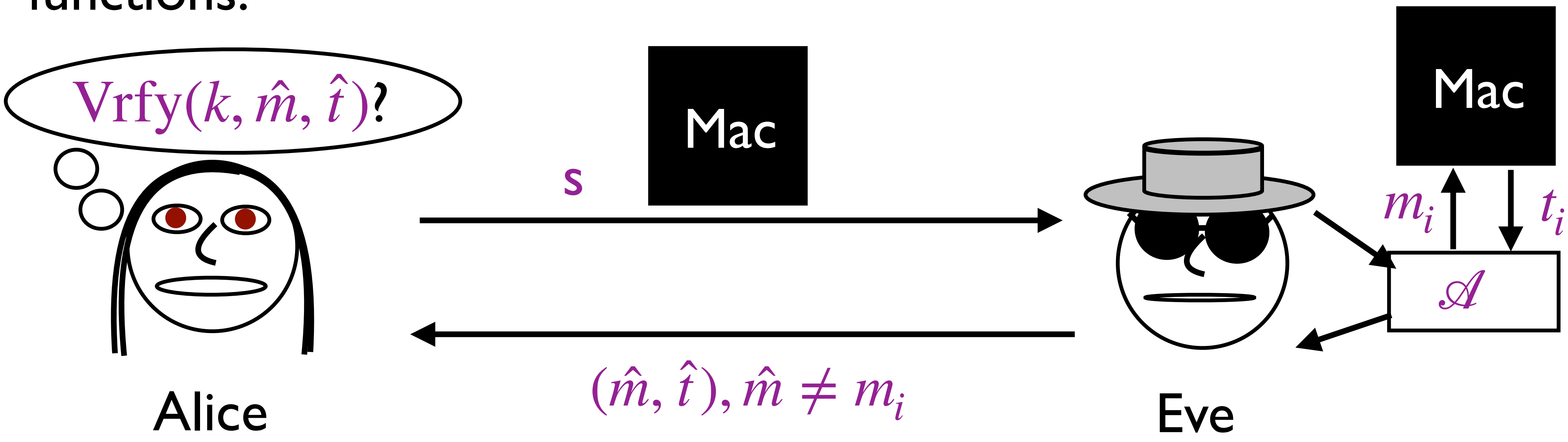
1. Alice gives Eve access to an oracle that Eve can query with possible messages m_i and receive valid tags $t_i = \mathrm{Mac}(k, m_i)$ for m_i using the key k.
2. Eve runs a polynomial-size attack using the oracle access and generates a (message, tag) pair $(\hat{m}, \hat{t})$.
3. Eve wins if $\mathrm{Vrfy}(k, \hat{m}, \hat{t}) =$ valid **and** $\hat{m} \neq m_i$ for any m_i which she used to query the oracle.

MAC Security Definition

Definition: A MAC (Gen, Mac, Vrfy) with security parameter s is secure (against an adaptive chosen-message attack) if, for any polynomial-time attack $\mathscr{A}$ with oracle access to $M_k(m) = \mathrm{Mac}(k, m)$, where $\mathscr{A}$ outputs $(\hat{m}, \hat{t})$ such that $\mathscr{A}$ never queried the oracle for $m = \hat{m}$,

$$\Pr(\mathrm{Vrfy}(k, \hat{m}, \hat{t}) = \mathrm{valid}) \leq \epsilon(s)$$

where $\epsilon(s)$ is a negligible function and the probability is averaged over k generated by Gen and the randomness used in any of the functions.



This class is being recorded

Security of MAC w/ PRF

Theorem: If $F_k(m)$ is a pseudorandom function then the following MAC is secure:

Gen chooses a uniform random bit string k.

$\mathrm{Mac}(k, m) = F_k(m)$.

Vrfy(k,m,t) outputs "valid" if $\mathrm{Mac}(k, m) = t$.

Security of MAC w/ PRF

Theorem: If $F_k(m)$ is a pseudorandom function then the following MAC is secure:

Gen chooses a uniform random bit string k.

$\mathrm{Mac}(k, m) = F_k(m)$.

Vrfy(k,m,t) outputs "valid" if $\mathrm{Mac}(k, m) = t$.

Proof Idea:

First show that the protocol is secure if we use a random function f instead of the pseudorandom function.

Security of MAC w/ PRF

Theorem: If $F_k(m)$ is a pseudorandom function then the following MAC is secure:

Gen chooses a uniform random bit string k.

$\mathrm{Mac}(k, m) = F_k(m)$.

Vrfy(k,m,t) outputs "valid" if $\mathrm{Mac}(k, m) = t$.

Proof Idea:

First show that the protocol is secure if we use a random function f instead of the pseudorandom function.

Then show, via reduction, that if we have an attack against the protocol with a pseudorandom function, then we can distinguish $F_k(m)$ from a random function.

Security of MAC w/ PRF

Theorem: If $F_k(m)$ is a pseudorandom function then the following MAC is secure:

Gen chooses a uniform random bit string k.

$\mathrm{Mac}(k, m) = F_k(m)$.

Vrfy(k,m,t) outputs "valid" if $\mathrm{Mac}(k, m) = t$.

Proof Idea:

First show that the protocol is secure if we use a random function f instead of the pseudorandom function.

Then show, via reduction, that if we have an attack against the protocol with a pseudorandom function, then we can distinguish $F_k(m)$ from a random function.

Which, by the definition of pseudorandom function, implies the protocol is secure.

MAC with Random Function

Lemma: The following MAC is secure:

Gen chooses a random function f as the key.

$\mathrm{Mac}(f, m) = f(m)$.

Vrfy(f,m,t) outputs "valid" if $\mathrm{Mac}(f, m) = t$.

Proof:

MAC with Random Function

Lemma: The following MAC is secure:

Gen chooses a random function f as the key.

$\mathrm{Mac}(f, m) = f(m)$.

Vrfy(f,m,t) outputs "valid" if $\mathrm{Mac}(f, m) = t$.

Proof:

Eve gets an oracle for f(m). She queries it on inputs m_i.

MAC with Random Function

Lemma: The following MAC is secure:

Gen chooses a random function f as the key.

$\mathrm{Mac}(f, m) = f(m)$.

Vrfy(f,m,t) outputs "valid" if $\mathrm{Mac}(f, m) = t$.

Proof:

Eve gets an oracle for f(m). She queries it on inputs m_i.

She chooses an $\hat{m} \neq m_i$. But f(m) is uncorrelated with f(m') for $m \neq m'$. In particular, $f(\hat{m})$ is a random string uncorrelated with all $f(m_i)$ and all of Eve's other data.

MAC with Random Function

Lemma: The following MAC is secure:

Gen chooses a random function f as the key.

$\mathrm{Mac}(f, m) = f(m)$.

Vrfy(f,m,t) outputs "valid" if $\mathrm{Mac}(f, m) = t$.

Proof:

Eve gets an oracle for f(m). She queries it on inputs m_i.

She chooses an $\hat{m} \neq m_i$. But f(m) is uncorrelated with f(m') for $m \neq m'$. In particular, $f(\hat{m})$ is a random string uncorrelated with all $f(m_i)$ and all of Eve's other data. Therefore, the probability that her attack outputs $\hat{t} = f(\hat{m})$ is

$$\Pr(\mathscr{A}^f = (\hat{m}, \hat{t} = f(\hat{m}))) = 2^{-s}$$

where f has outputs of length s bits.

Reduction Strategy

We want a reduction from security of the MAC protocol to security of the pseudorandom function.

Reduction Strategy

We want a reduction from security of the MAC protocol to security of the pseudorandom function.

In other words, we want the following: Given a successful attack on the MAC, turn it into an attack on the pseudorandom function.

Reduction Strategy

We want a reduction from security of the MAC protocol to security of the pseudorandom function.

> In other words, we want the following: Given a successful attack on the MAC, turn it into an attack on the pseudorandom function.

The pseudorandom function game is to distinguish $F_k(m)$ from a random function.

Reduction Strategy

We want a reduction from security of the MAC protocol to security of the pseudorandom function.

In other words, we want the following: Given a successful attack on the MAC, turn it into an attack on the pseudorandom function.

The pseudorandom function game is to distinguish $F_k(m)$ from a random function.

So:

Reduction Strategy

We want a reduction from security of the MAC protocol to security of the pseudorandom function.

In other words, we want the following: Given a successful attack on the MAC, turn it into an attack on the pseudorandom function.

The pseudorandom function game is to distinguish $F_k(m)$ from a random function.

So:

- Given function f, we run a virtual MAC security game.

Reduction Strategy

We want a reduction from security of the MAC protocol to security of the pseudorandom function.

In other words, we want the following: Given a successful attack on the MAC, turn it into an attack on the pseudorandom function.

The pseudorandom function game is to distinguish $F_k(m)$ from a random function.

So:

- Given function f, we run a virtual MAC security game.
- If f is random, the MAC is secure by the lemma, so Virtual Eve is likely to lose. Thus, if Eve loses: guess random.

Reduction Strategy

We want a reduction from security of the MAC protocol to security of the pseudorandom function.

In other words, we want the following: Given a successful attack on the MAC, turn it into an attack on the pseudorandom function.

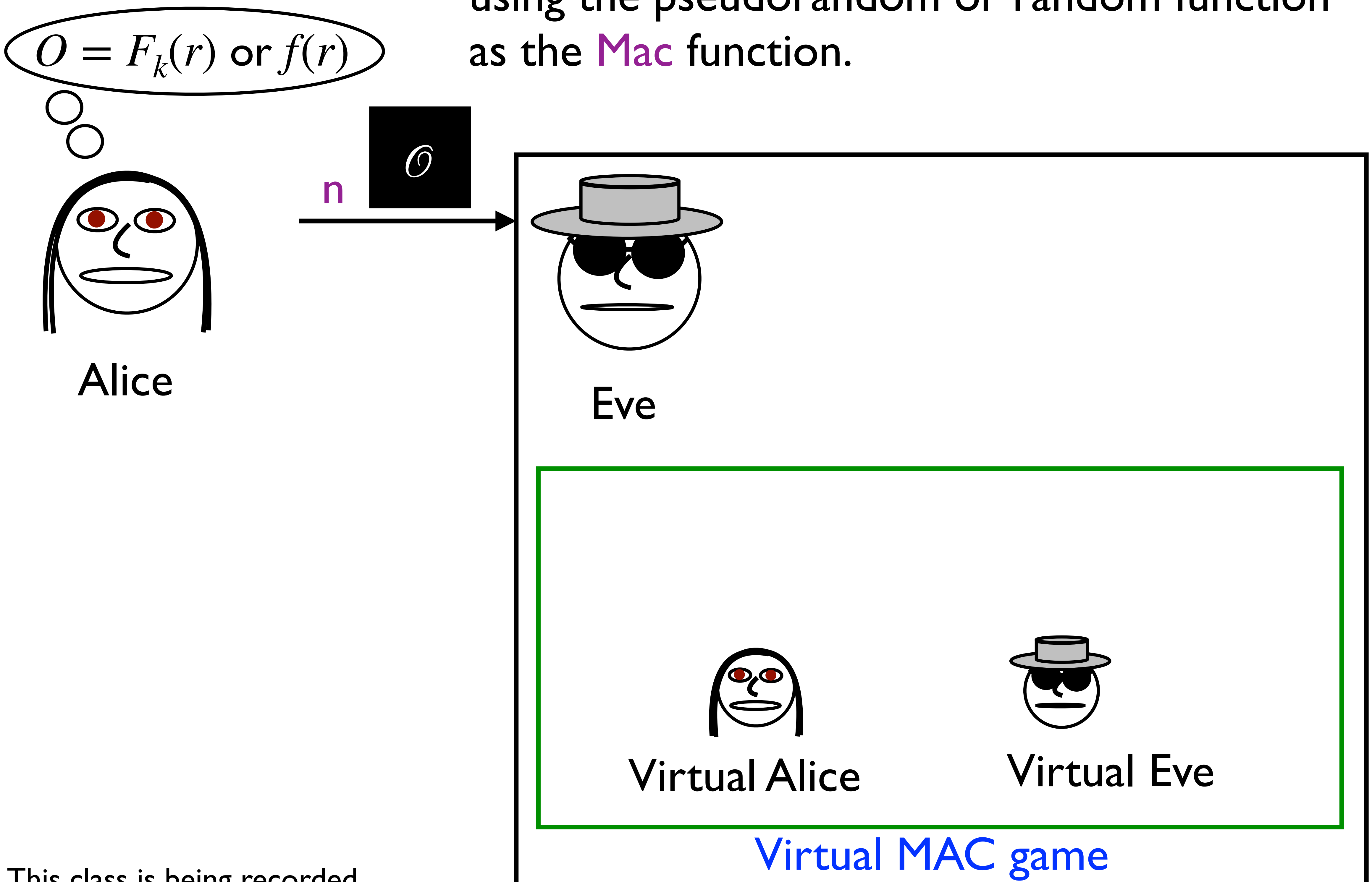
The pseudorandom function game is to distinguish $F_k(m)$ from a random function.

So:

- Given function f, we run a virtual MAC security game.
- If f is random, the MAC is secure by the lemma, so Virtual Eve is likely to lose. Thus, if Eve loses: guess random.
- If $f = F_k$, the attack on the MAC succeeds, so Eve is likely to win. Thus, if Eve wins: guess pseudorandom.

Reduction to Pseudorandom Function

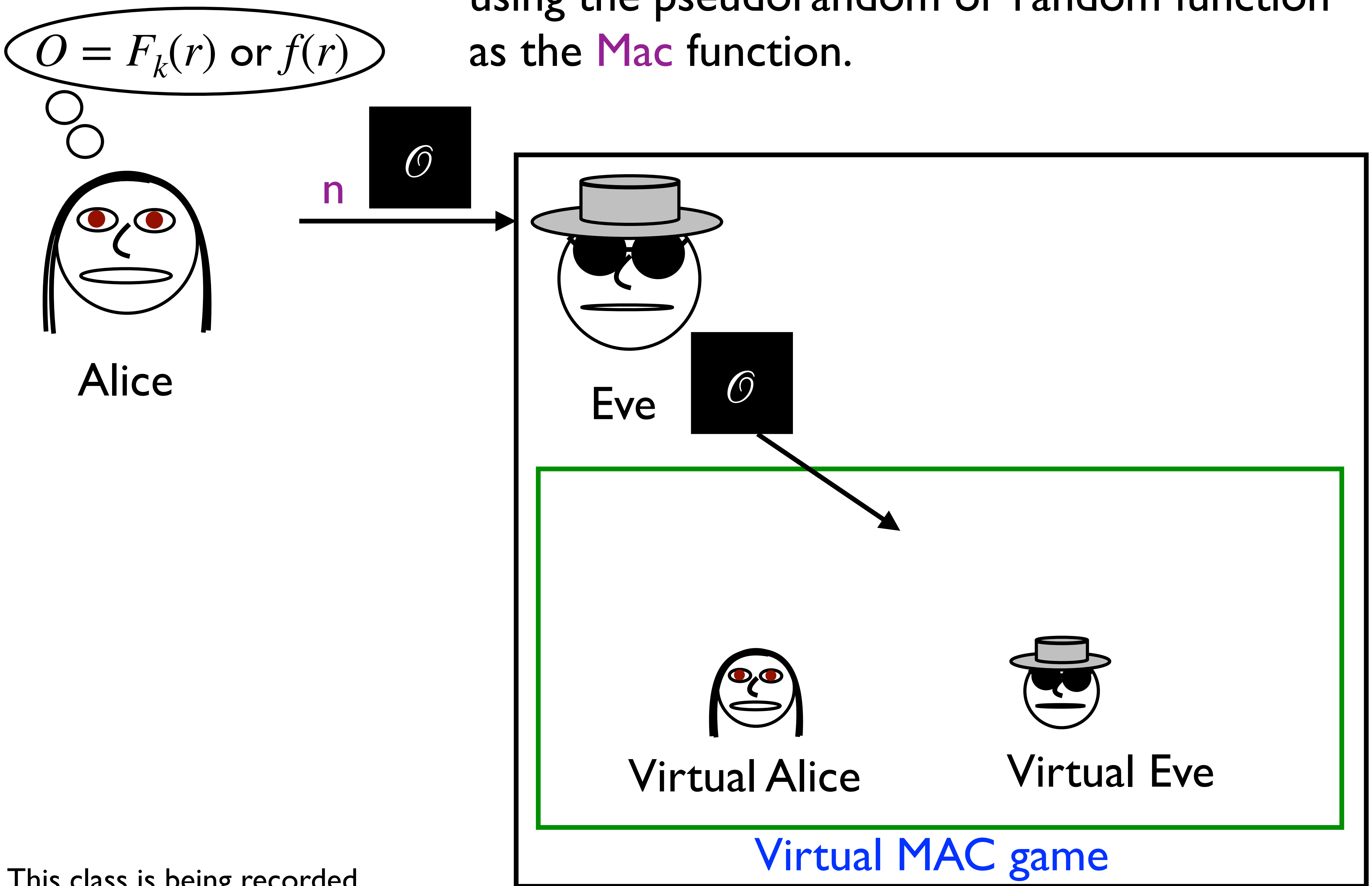
The reduction runs a virtual MAC protocol using the pseudorandom or random function as the Mac function.



This class is being recorded

Reduction to Pseudorandom Function

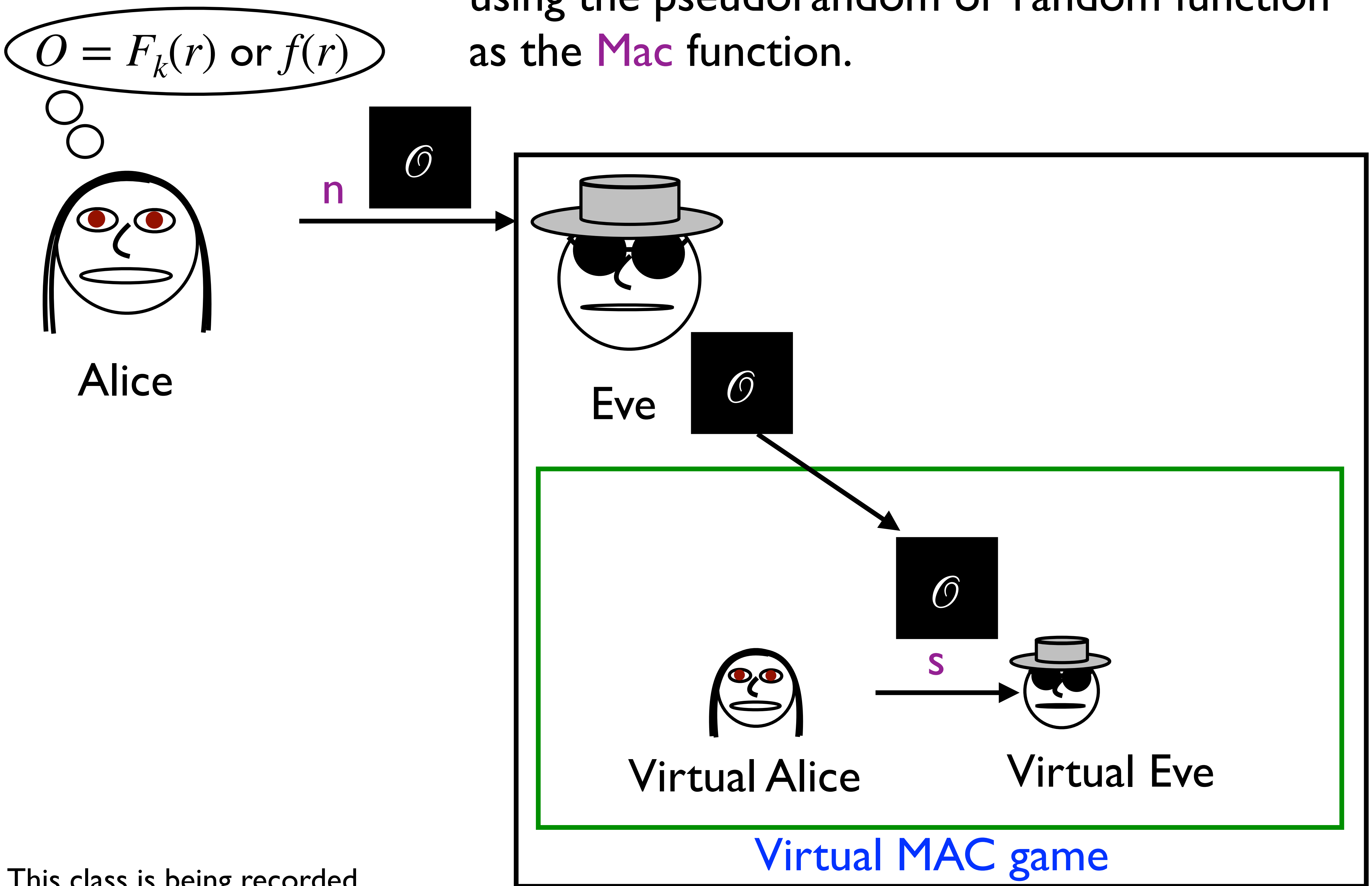
The reduction runs a virtual MAC protocol using the pseudorandom or random function as the Mac function.



This class is being recorded

Reduction to Pseudorandom Function

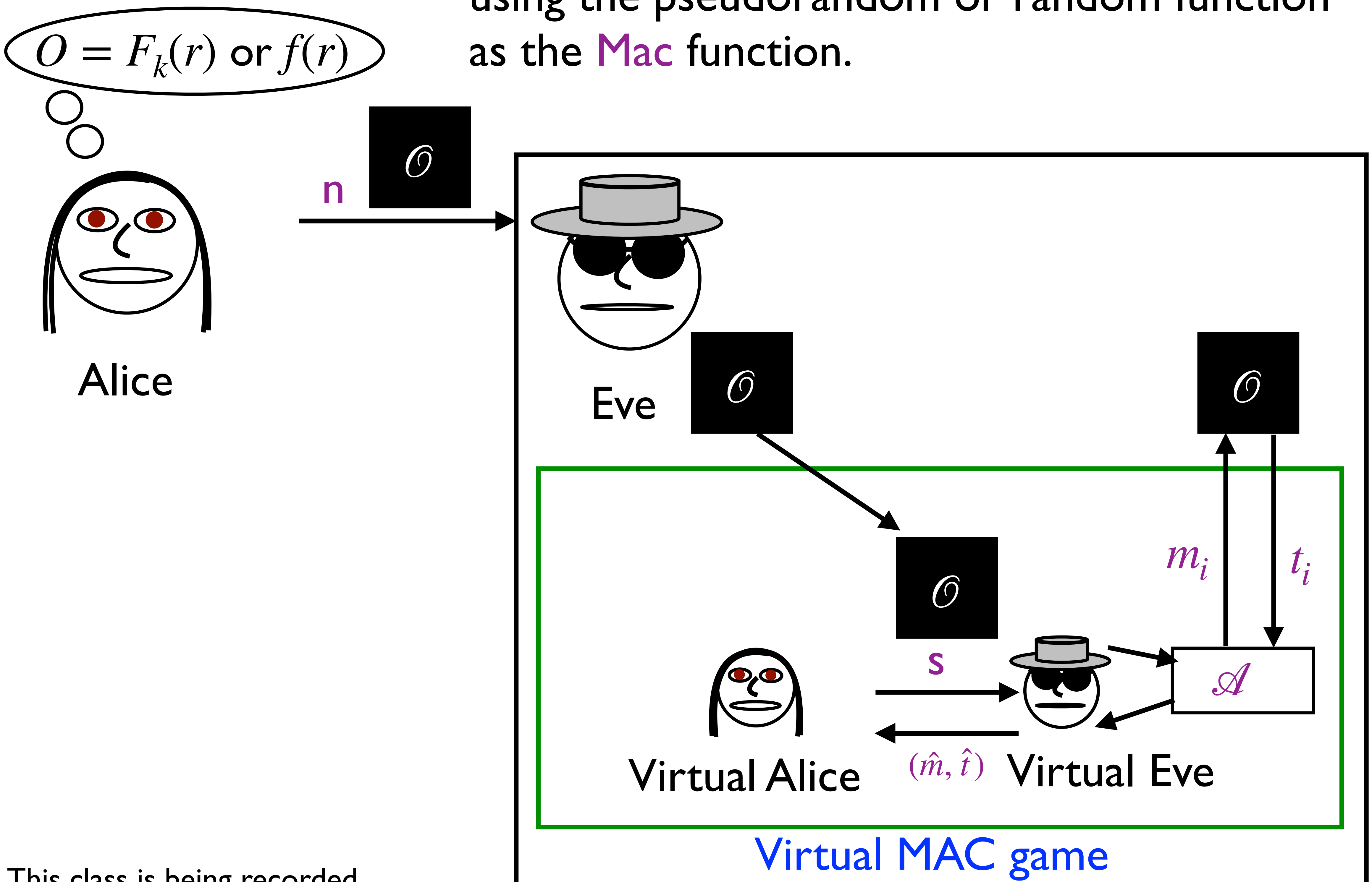
The reduction runs a virtual MAC protocol using the pseudorandom or random function as the Mac function.



This class is being recorded

Reduction to Pseudorandom Function

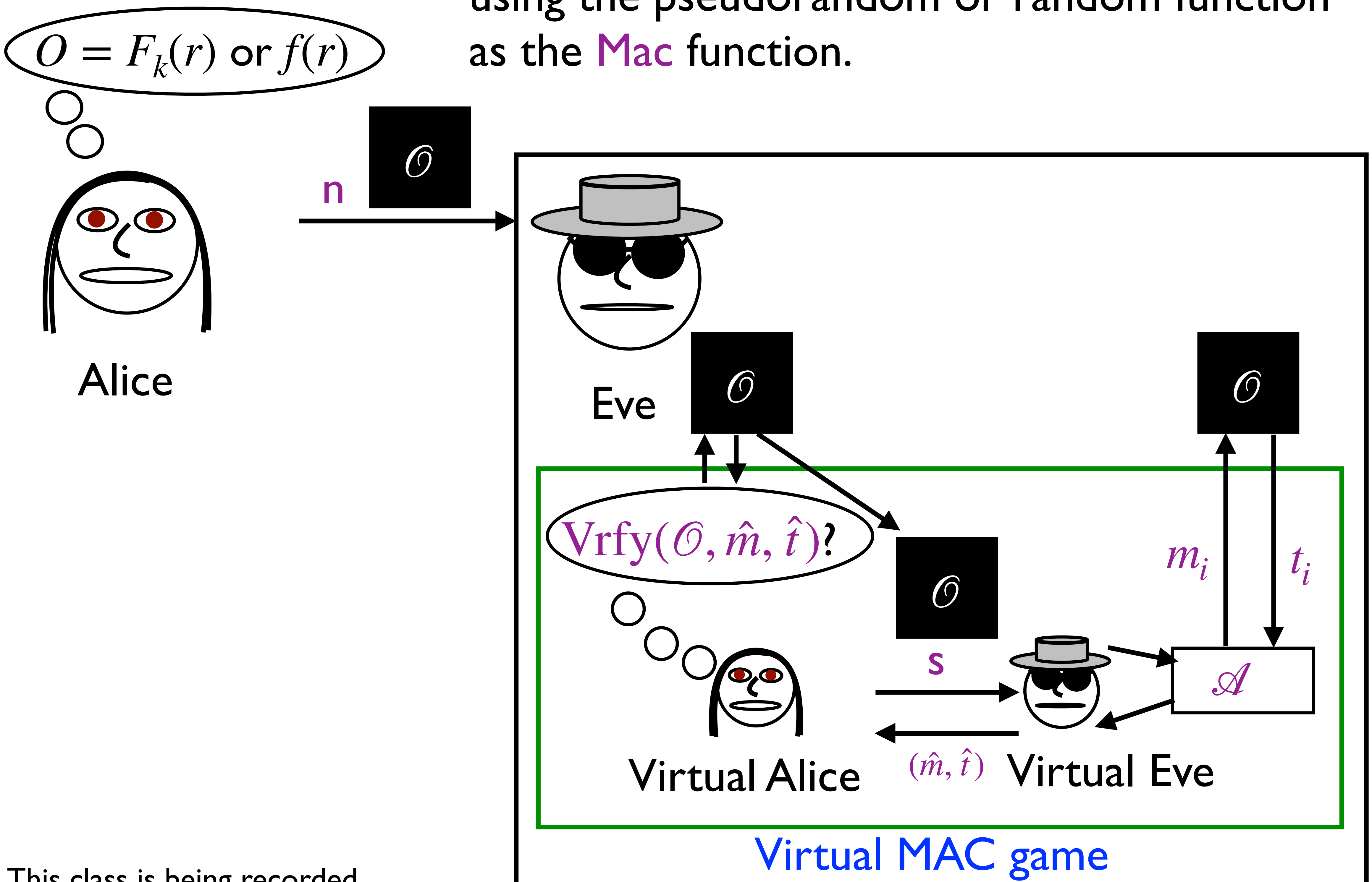
The reduction runs a virtual MAC protocol using the pseudorandom or random function as the Mac function.



This class is being recorded

Reduction to Pseudorandom Function

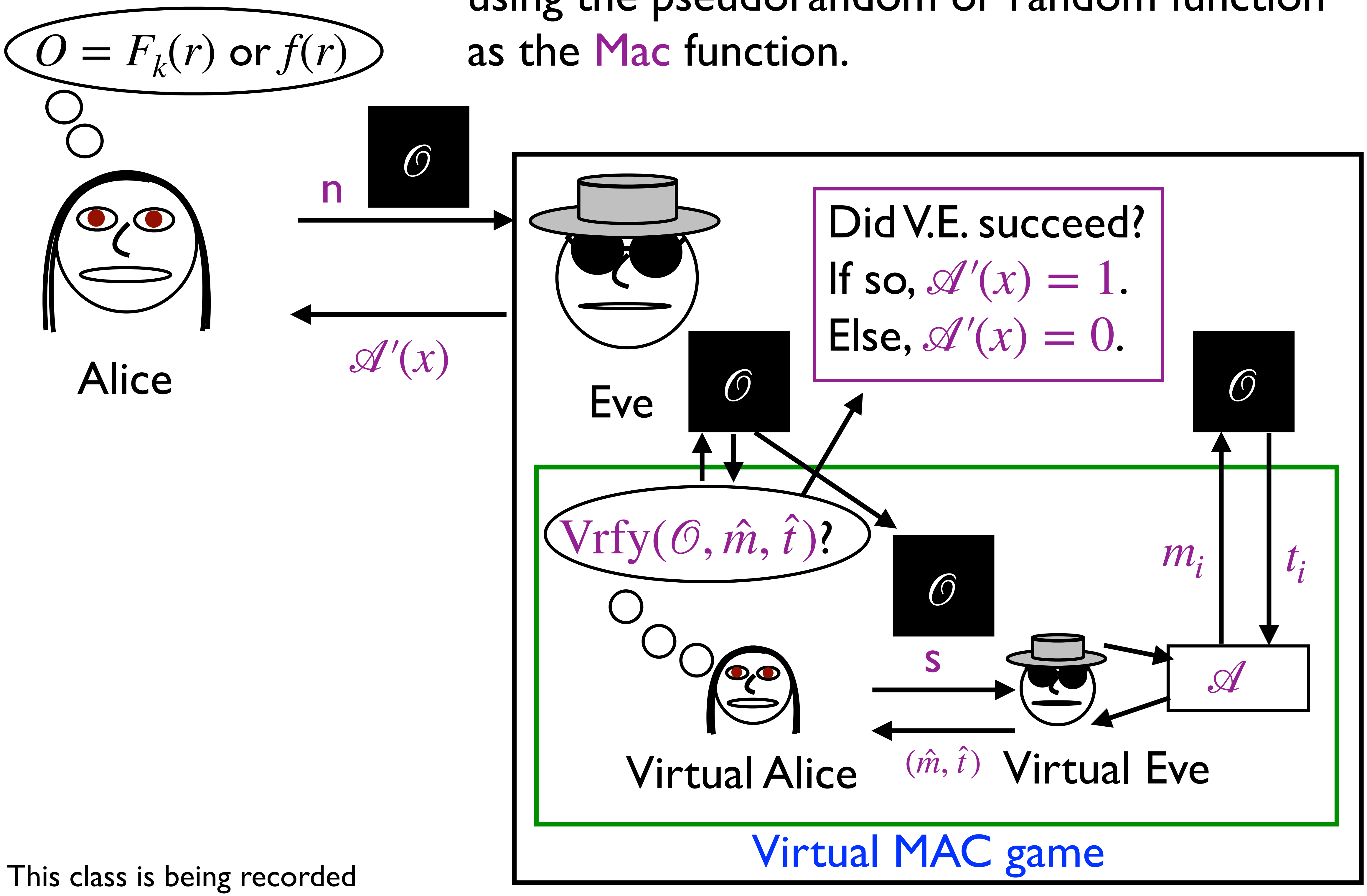
The reduction runs a virtual MAC protocol using the pseudorandom or random function as the Mac function.



This class is being recorded

Reduction to Pseudorandom Function

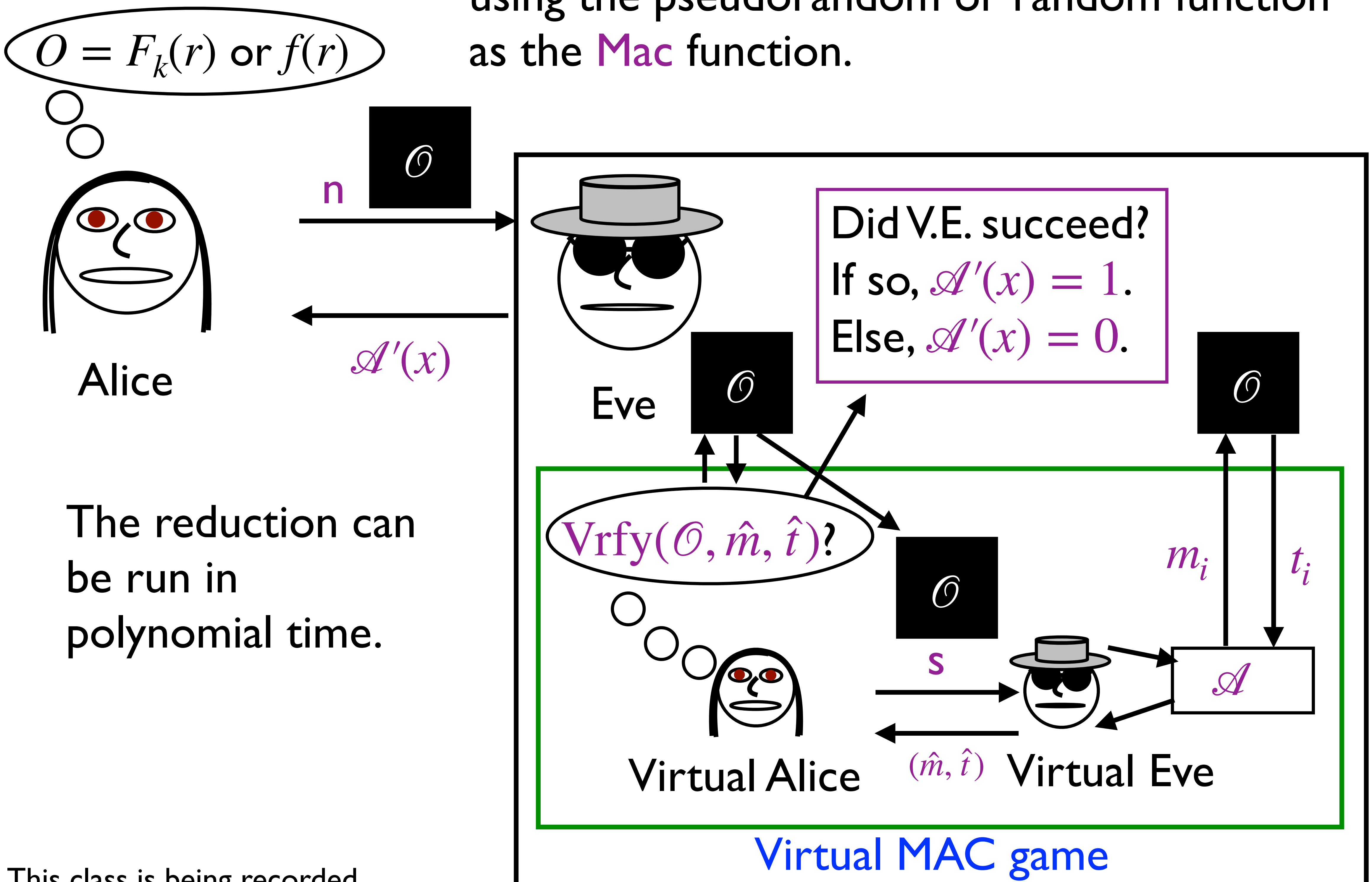
The reduction runs a virtual MAC protocol using the pseudorandom or random function as the Mac function.



This class is being recorded

Reduction to Pseudorandom Function

The reduction runs a virtual MAC protocol using the pseudorandom or random function as the Mac function.



The reduction can be run in polynomial time.

This class is being recorded

Completing the Proof

In this reduction, the real Eve outputs $\mathscr{A}' = 1$ iff virtual Eve succeeds in outputting a correct message-tag pair $(\hat{m}, \hat{t})$.

Completing the Proof

In this reduction, the real Eve outputs $\mathscr{A}' = 1$ iff virtual Eve succeeds in outputting a correct message-tag pair $(\hat{m}, \hat{t})$.

- If the function is random, this happens with probability 2^{-s} by the lemma. That is,

$$\Pr(\mathscr{A}'^f = 1) = 2^{-s}$$

Completing the Proof

In this reduction, the real Eve outputs $\mathscr{A}' = 1$ iff virtual Eve succeeds in outputting a correct message-tag pair $(\hat{m}, \hat{t})$.

- If the function is random, this happens with probability 2^{-s} by the lemma. That is,

$$\Pr(\mathscr{A}'^{f} = 1) = 2^{-s}$$

- If the function is pseudorandom, virtual Eve succeeds with probability $\Pr(\mathrm{Vrfy}(k, \hat{m}, \hat{t}) = \mathrm{valid})$:

$$\Pr(\mathscr{A}'^{F_k} = 1) = \Pr(\mathrm{Vrfy}(k, \hat{m}, \hat{t}) = \mathrm{valid})$$

Completing the Proof

In this reduction, the real Eve outputs $\mathscr{A}' = 1$ iff virtual Eve succeeds in outputting a correct message-tag pair $(\hat{m}, \hat{t})$.

- If the function is random, this happens with probability 2^{-s} by the lemma. That is,

$$\Pr(\mathscr{A}'^f = 1) = 2^{-s}$$

- If the function is pseudorandom, virtual Eve succeeds with probability $\Pr(\text{Vrfy}(k, \hat{m}, \hat{t}) = \text{valid})$:

$$\Pr(\mathscr{A}'^{F_k} = 1) = \Pr(\text{Vrfy}(k, \hat{m}, \hat{t}) = \text{valid})$$

Thus,

$$|\Pr(\mathscr{A}'^{F_k} = 1) - \Pr(\mathscr{A}'^f = 1)| = |\Pr(\text{Vrfy}(k, \hat{m}, \hat{t}) = \text{valid}) - 2^{-s}|$$

Completing the Proof

In this reduction, the real Eve outputs $\mathscr{A}' = 1$ iff virtual Eve succeeds in outputting a correct message-tag pair $(\hat{m}, \hat{t})$.

- If the function is random, this happens with probability 2^{-s} by the lemma. That is,

$$\Pr(\mathscr{A}'^{f} = 1) = 2^{-s}$$

- If the function is pseudorandom, virtual Eve succeeds with probability $\Pr(\text{Vrfy}(k, \hat{m}, \hat{t}) = \text{valid})$:

$$\Pr(\mathscr{A}'^{F_k} = 1) = \Pr(\text{Vrfy}(k, \hat{m}, \hat{t}) = \text{valid})$$

Thus,

$$|\Pr(\mathscr{A}'^{F_k} = 1) - \Pr(\mathscr{A}'^{f} = 1)| = |\Pr(\text{Vrfy}(k, \hat{m}, \hat{t}) = \text{valid}) - 2^{-s}|$$

But, by the definition of a pseudorandom function,

$$|\Pr(\mathscr{A}'^{F_k} = 1) - \Pr(\mathscr{A}'^{f} = 1)| < \epsilon(s)$$

Completing the Proof

In this reduction, the real Eve outputs $\mathscr{A}' = 1$ iff virtual Eve succeeds in outputting a correct message-tag pair $(\hat{m}, \hat{t})$.

- If the function is random, this happens with probability 2^{-s} by the lemma. That is,

$$\Pr(\mathscr{A}'^f = 1) = 2^{-s}$$

- If the function is pseudorandom, virtual Eve succeeds with probability $\Pr(\text{Vrfy}(k, \hat{m}, \hat{t}) = \text{valid})$:

$$\Pr(\mathscr{A}'^{F_k} = 1) = \Pr(\text{Vrfy}(k, \hat{m}, \hat{t}) = \text{valid})$$

Thus,

$$|\Pr(\mathscr{A}'^{F_k} = 1) - \Pr(\mathscr{A}'^f = 1)| = |\Pr(\text{Vrfy}(k, \hat{m}, \hat{t}) = \text{valid}) - 2^{-s}|$$

But, by the definition of a pseudorandom function,

$$|\Pr(\mathscr{A}'^{F_k} = 1) - \Pr(\mathscr{A}'^f = 1)| < \epsilon(s)$$

so $\Pr(\text{Vrfy}(k, \hat{m}, \hat{t}) = \text{valid}) < 2^{-s} + \epsilon(s) = \epsilon'(s)$

Longer Messages

Standardized block ciphers have a fixed size. So how do we authenticate longer messages?

Break m up into blocks: $m = m_0 \| m_1 \| m_2 \| \ldots$ and authenticate each one separately: $(m_0, t_0), (m_1, t_1), (m_2, t_2), \ldots$

Vote: Secure? (Yes/No/Unknown)

Longer Messages

Standardized block ciphers have a fixed size. So how do we authenticate longer messages?

Break m up into blocks: $m = m_0 \| m_1 \| m_2 \| \ldots$ and authenticate each one separately: $(m_0, t_0), (m_1, t_1), (m_2, t_2), \ldots$

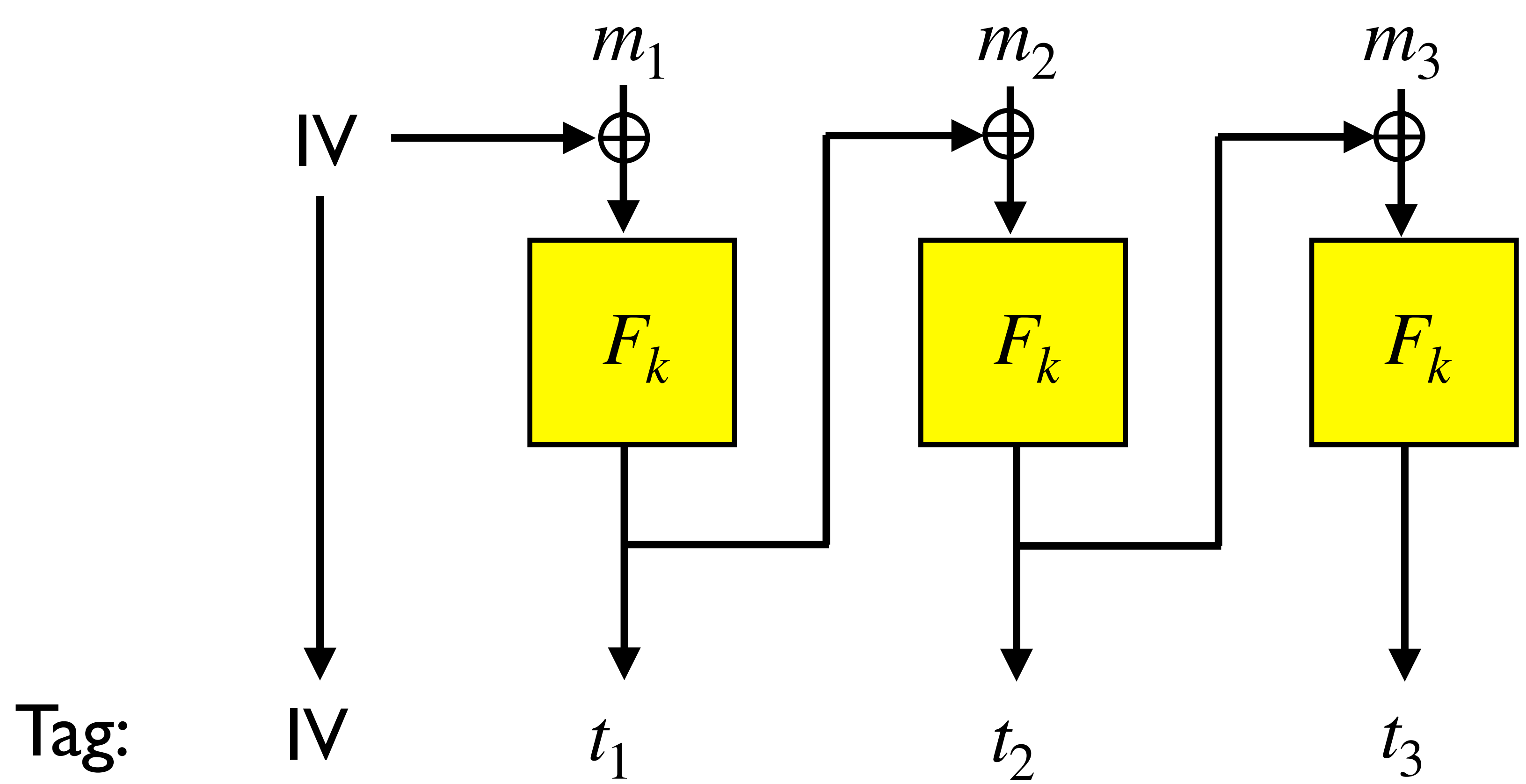
Vote: Secure? (Yes/No/Unknown)

Answer: No. Eve has various attacks.

- Could change the order: $(m_2, t_2), (m_1, t_1), (m_0, t_0), \ldots$ is a valid set of tags for the message $m_2 \| m_1 \| m_0 \| \ldots$
- Could truncate: (m_0, t_0) by itself is a valid tag for the message m_0.
- Could switch blocks from multiple messages: Given (m_i, t_i) and (m_i', t_i'), she could send $(m_0, t_0), (m_1', t_1'), (m_2, t_2), \ldots$, which is a valid set of tags for $m_0 \| m_1' \| m_2 \| \ldots$ This is different from either of the original messages.

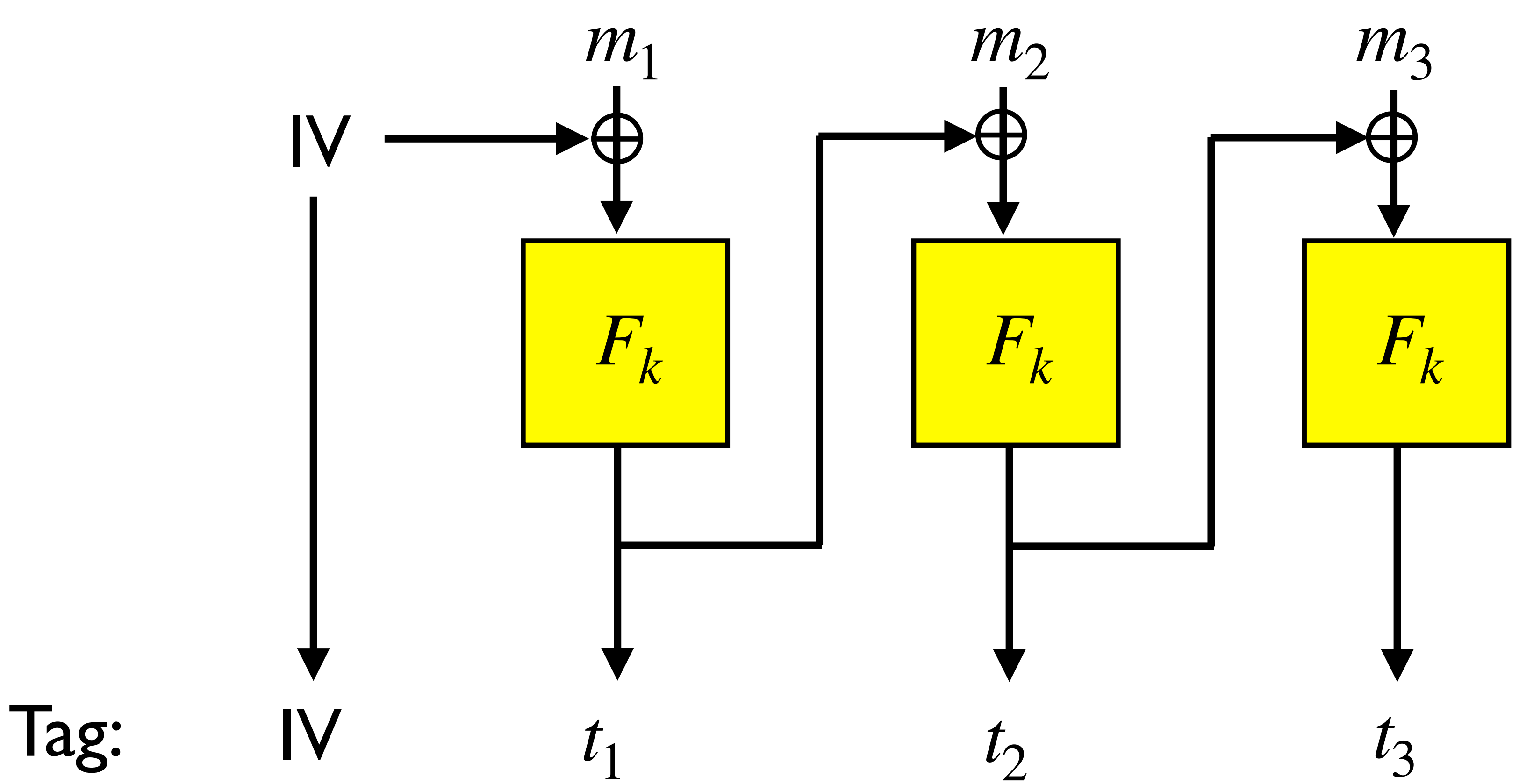
CBC Mode for MACs

What about CBC mode for MACs?



Vote: Secure? (Yes/No/Unknown)

CBC Doesn't Work for MACs



Not secure. Eve can change the IV *and* m_1 with it to leave the tags the same:

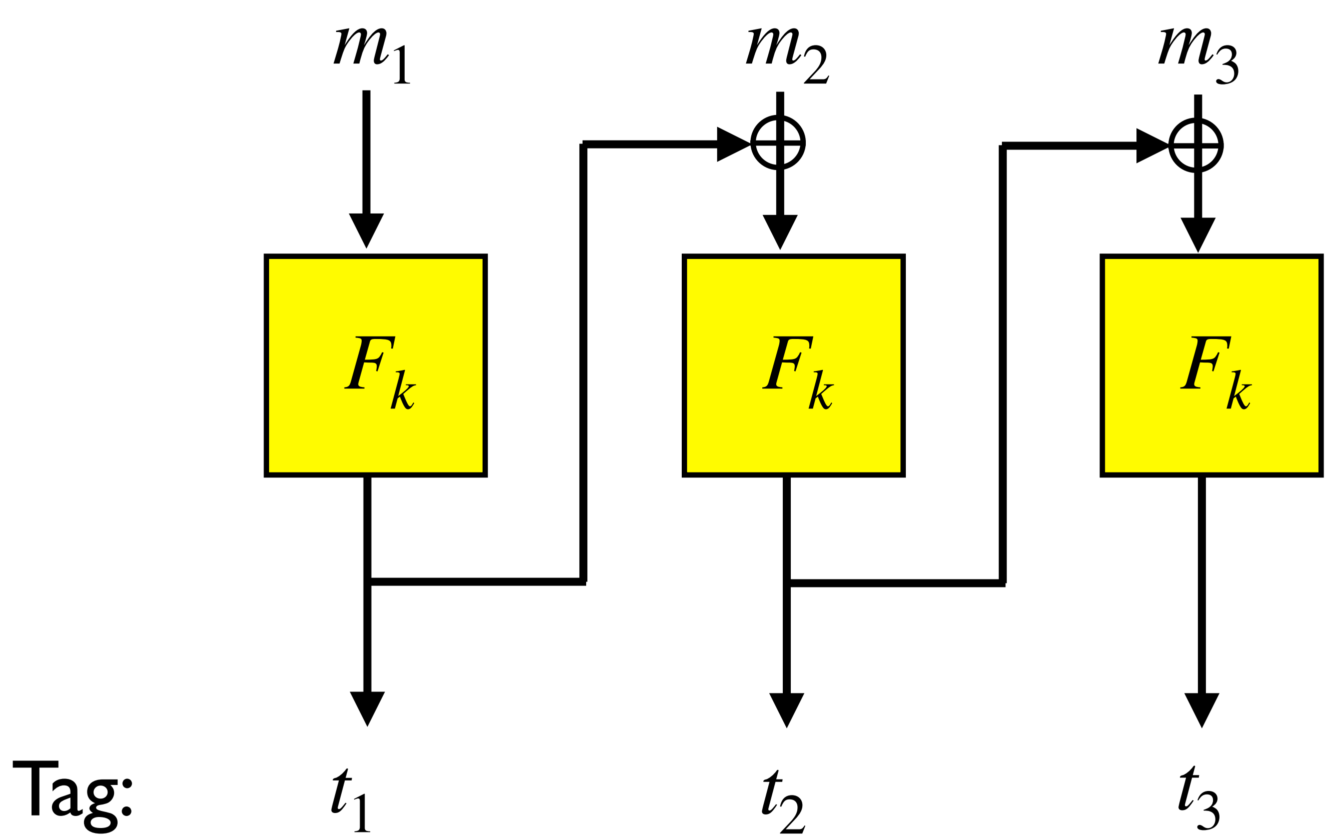
E.g.: IV = 010100, $m_1 = 110000$, so $IV \oplus m_1 = 100100$

But if IV = 110001, $\hat{m}_1 = 010101$, then it is still true that $IV \oplus \hat{m}_1 = 100100$, so the message $\hat{m}_1 \| m_2 \| m_3$ has the same tag as the original message $m_1 \| m_2 \| m_3$.

This class is being recorded

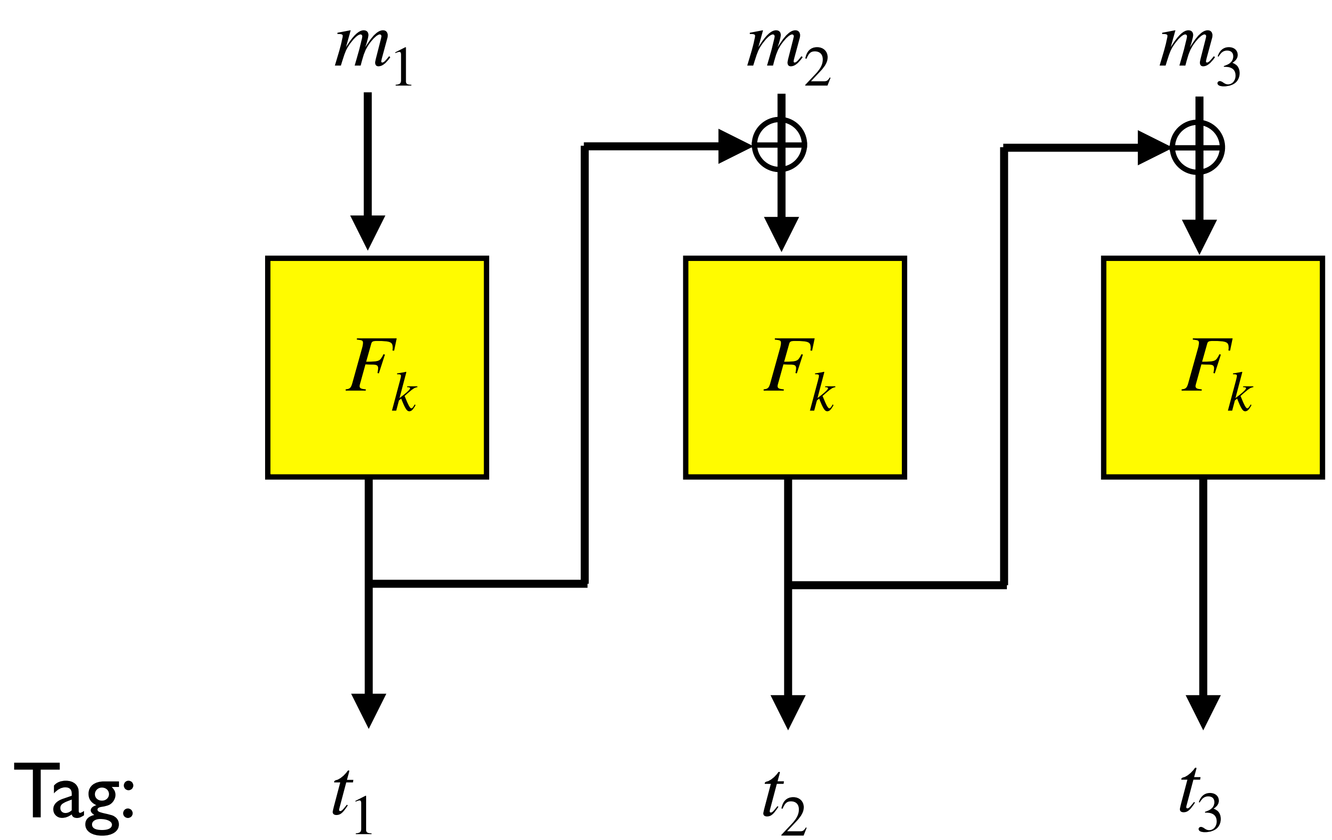
CBC for MACs with no IV

The IV is causing this problem since Eve can change it and m_1 together. But we don't need it: For encryption, we needed randomness to ensure CPA security, but MACs don't need that.



Vote: Secure? (Yes/No/Unknown)

It Still Doesn't Work



Not secure. Eve can still combine messages to make a new one:

E.g.: Given $(m_1, m_2, \ldots), (t_1, t_2, \ldots)$ and $(m_1', m_2', \ldots), (t_1', t_2', \ldots)$, Eve knows $t_2' = F_k(t_1' \oplus m_2')$.

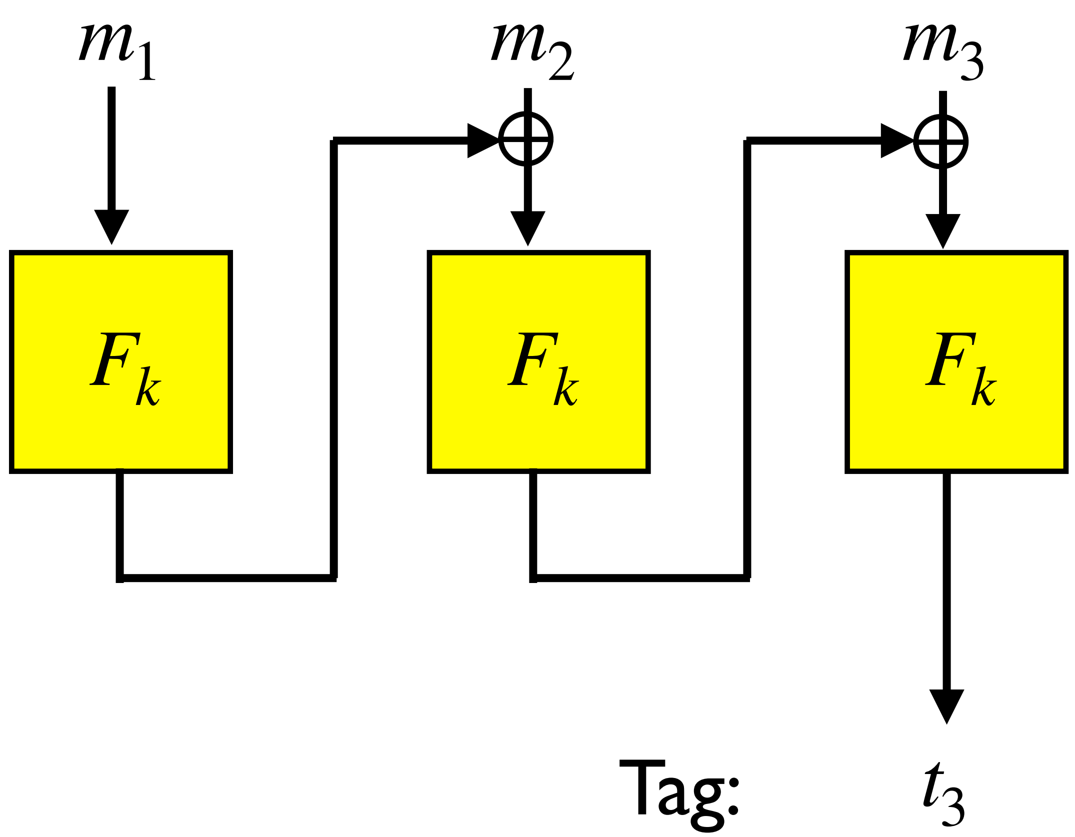
Let $m_2'' = t_1 \oplus t_1' \oplus m_2'$. Then $t_1 \oplus m_2'' = t_1' \oplus m_2'$, which means $F_k(t_1 \oplus m_2) = t_2'$ and the pair

$$(m_1, m_2'', m_3', \ldots), (t_1, t_2', t_3', \ldots) \quad \text{is valid.}$$

This class is being recorded

Another Try

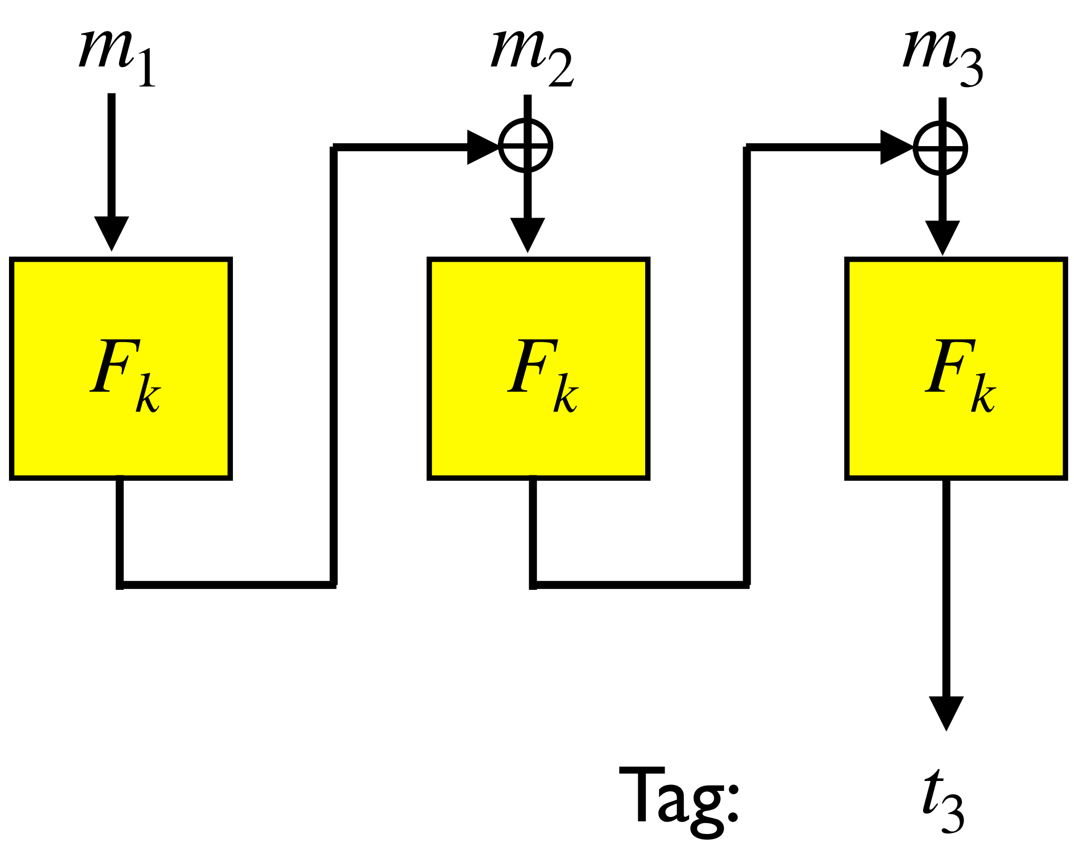
The problem seems to be the intermediate tags t_1, t_2 give Eve too much information. Maybe we should get rid of them:



Only output the final tag.

Vote: Secure? (Yes/No/Unknown)

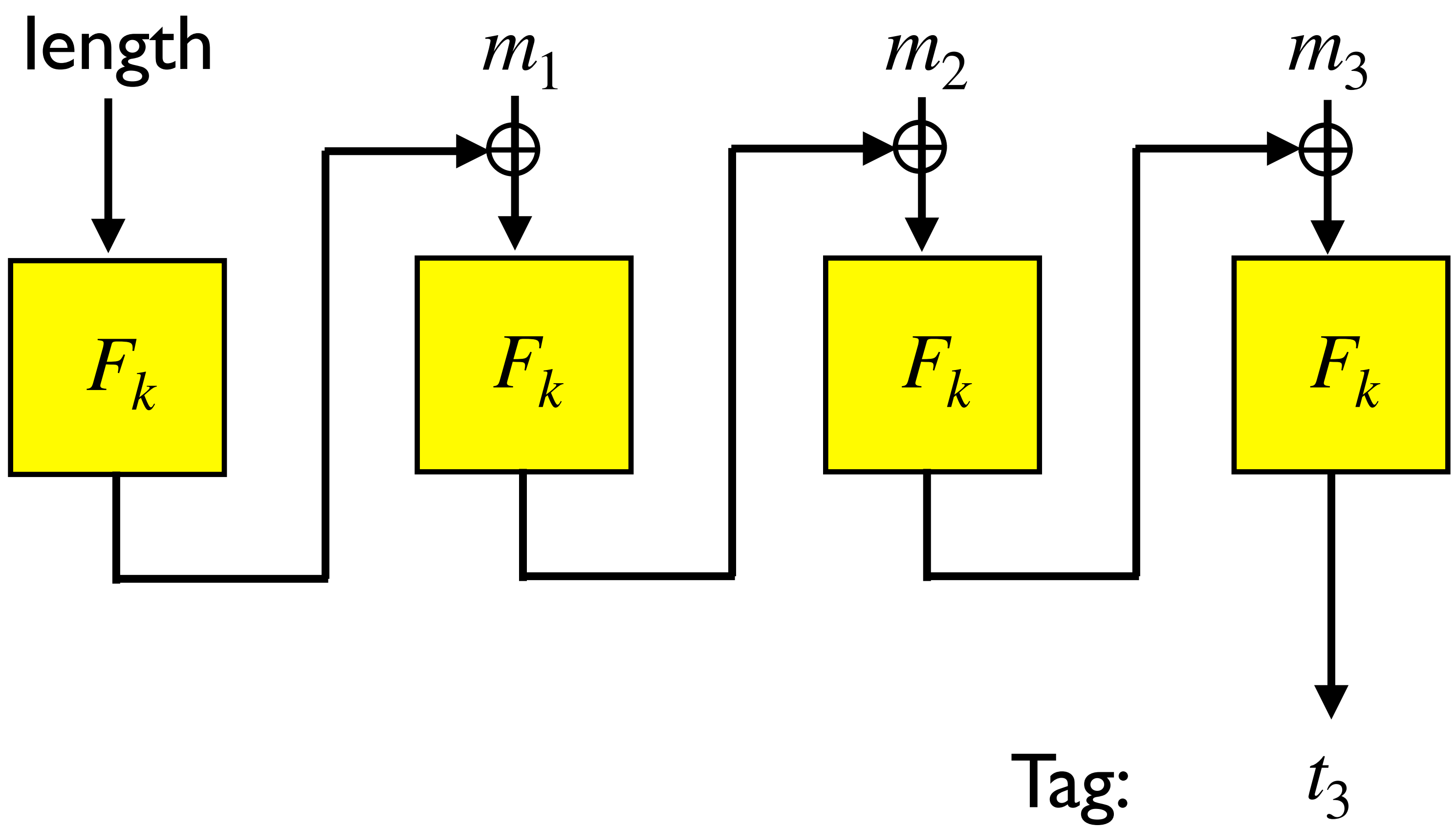
Almost There



Secure only if length fixed. Otherwise, Eve can get the tags t_1, t_2, etc. by querying the oracle for messages m_1, then $m_1 \| m_2$, etc.

Then Eve can proceed as before.

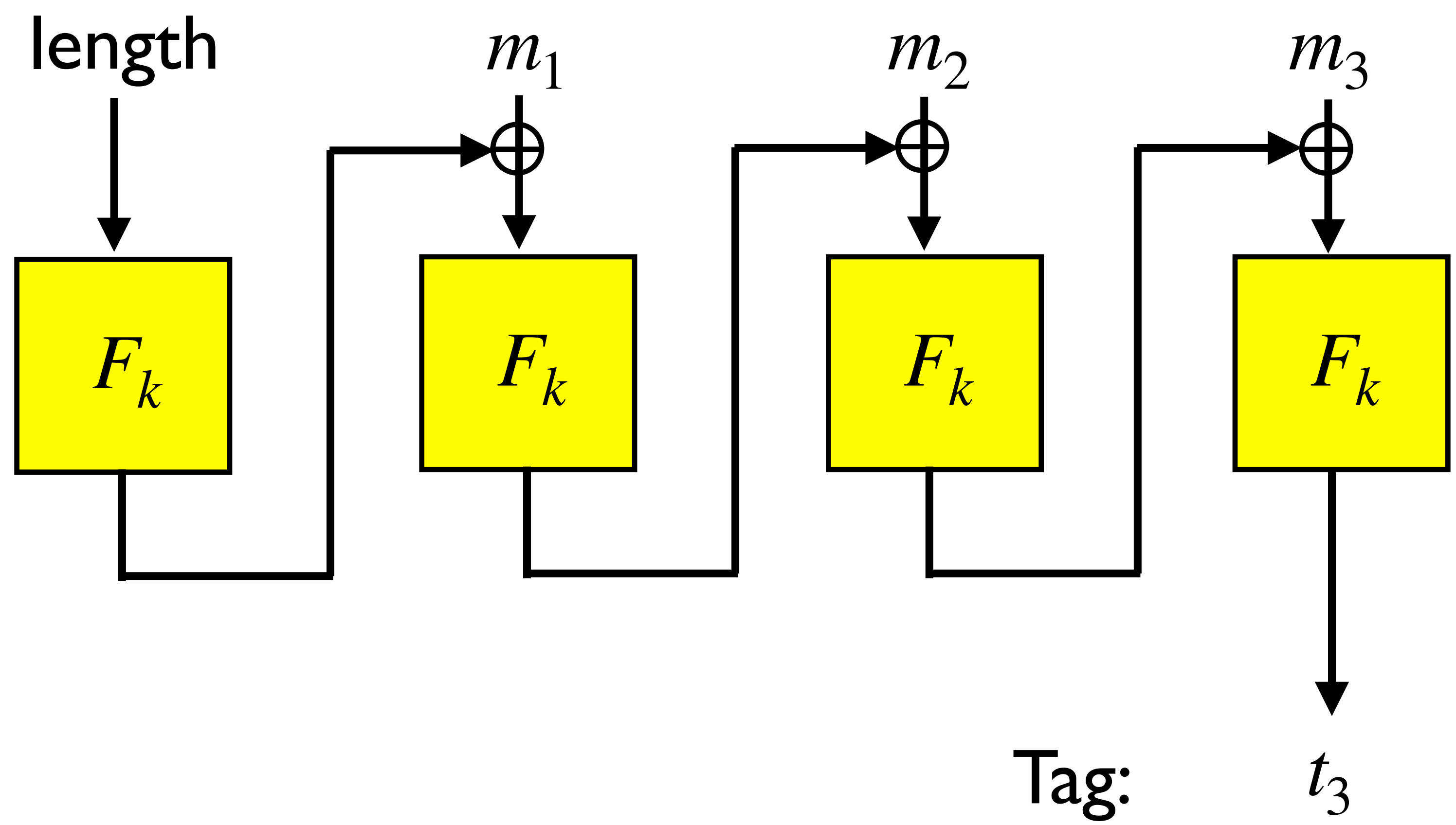
CBC with Length Included



We can authenticate the length as the first part of the message.

Vote: Secure? (Yes/No/Unknown)

CBC with Length Included



We can authenticate the length as the first part of the message.

Vote: Secure? (Yes/No/Unknown)

Yes! Finally.