MACs and Authenticated Encryption

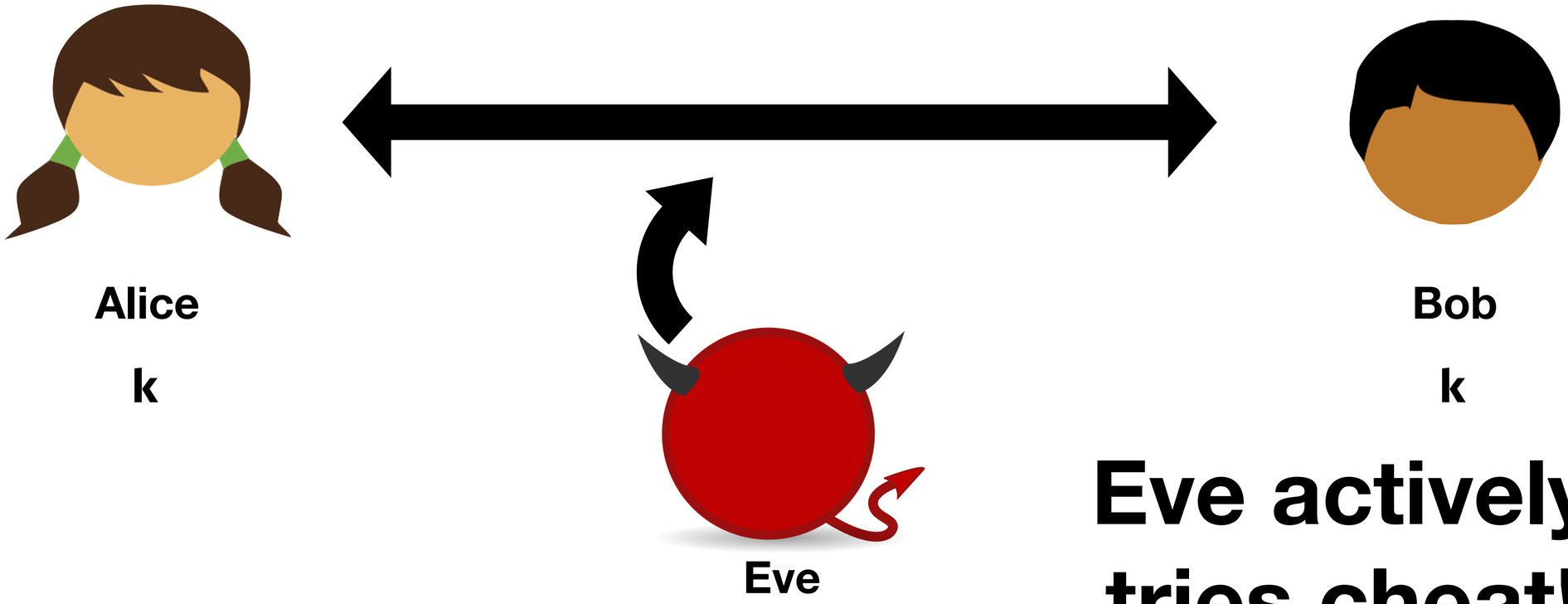
CS/ECE 407

Today's objectives

Revisit definition of Message Authentication Codes (MACs)

Achieve MACs for long messages
  Look at schemes that do and do not work.

Discuss strong notions of security that achieve authentication



Eve actively tries cheat!

Confidentiality

Can Alice and Bob prevent Eve from listening?

Authenticity

Can Bob be sure Eve did not send the message?

Can Bob be sure Eve did not alter a message from Alice?

A cipher (Enc, Dec) has **security against a chosen plaintext attack (CPA)** if:

```
k ← K

encrypt(m0, m1):
  return Enc(k, m0)
```

≈

```
k ← K

encrypt(m0, m1):
  return Enc(k, m1)
```

A cipher (Enc, Dec) has **security against a chosen ciphertext attack (CCA)** if:

```
k ← K
S ← empty-set

encrypt(m0, m1):
  c ← Enc(k, m0)
  S ← S ∪ {c}
  return c

decrypt(c):
  if c ∈ S:
    return error
  return Dec(k, c)
```

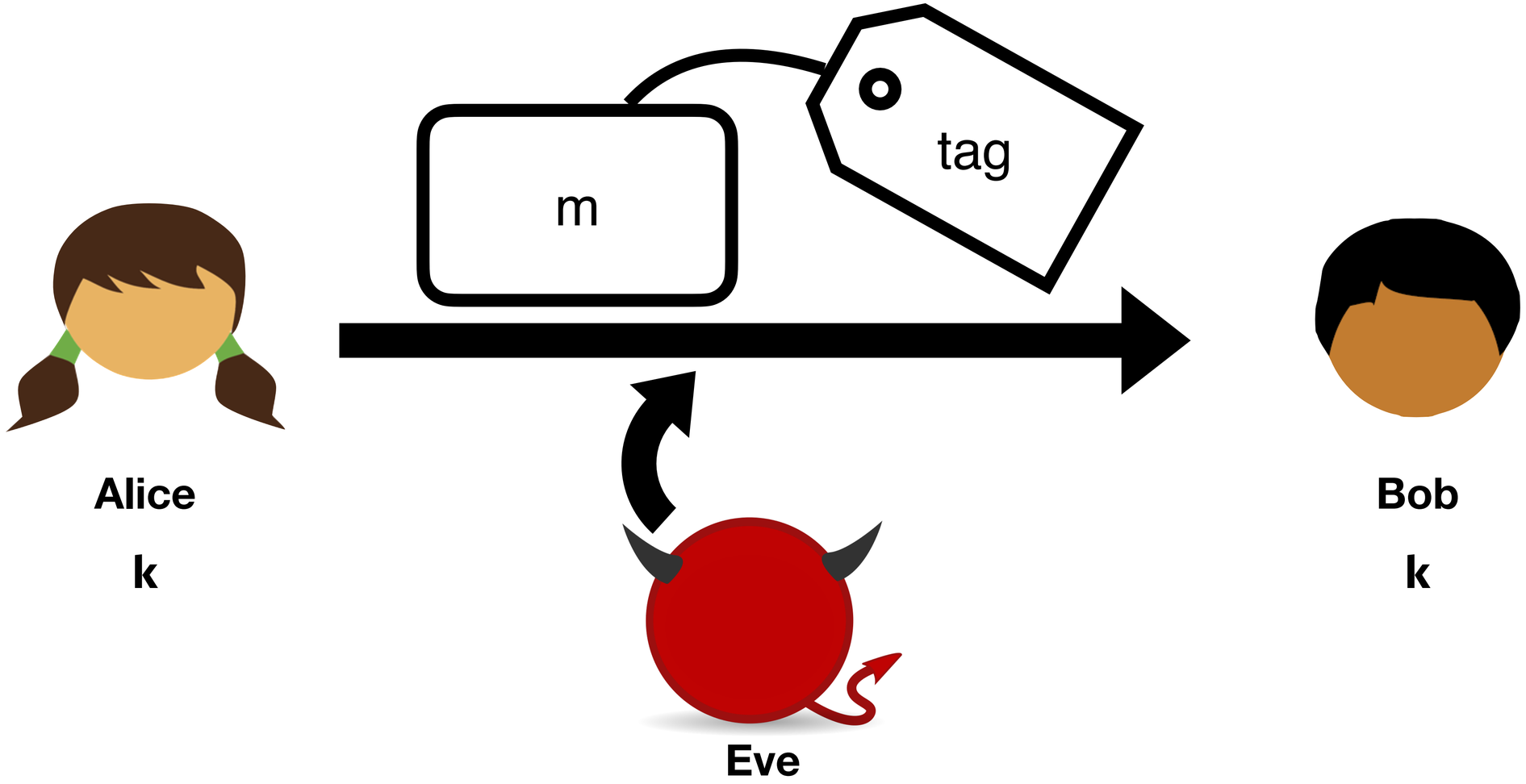
≈

```
k ← K
S ← empty-set

encrypt(m0, m1):
  c ← Enc(k, m1)
  S ← S ∪ {c}
  return c

decrypt(c):
  if c ∈ S:
    return error
  return Dec(k, c)
```

Message Authentication Codes (MACs)



"Eve cannot change m without breaking the tag"

Message Authentication Codes

A **MAC scheme** is an algorithm `tag` such that:

```
k ← K                          Real

get(m):
  return tag(k, m)

check(m, t):
  return tag(k, m) = t
```

$\overset{c}{\approx}$

```
k ← K                          Ideal
S ← empty-set

get(m):
  t ← tag(k, m)
  S ← S ∪ {(m, t)}
  return t

check(m, t):
  return (m, t) ∈ S
```

PRF ⇒ MAC (for block-length messages)

Let F be a PRF

```
tag(k, m):
  return F(k, m)
```

PRF ⇒ MAC (for block-length messages)

Let F be a PRF

```
tag(k, m):
  return F(k, m)
```

What about long messages?

What about arbitrary length messages?

Attempt

```
tag(k, m_0, …, m_{n-1}):
  t := 0^λ
  for i in {0,…,n-1}:
    t := t ⊕ F(k, m_i)
  return t
```

Attempt

```
tag(k, m_0, …, m_{n-1}):
  t := 0^λ
  for i in {0,…,n-1}:
    t := t ⊕ F(k, m_i)
  return t
```

CBC-MAC

```
tag(k, m_0, …, m_{n-1}):
  t := 0^λ
  for i in {0,…,n-1}:
    t := F(k, m_i ⊕ t)
  return t
```

CBC-MAC

```
tag(k, m_0, …, m_{n-1}):
  t := 0^λ
  for i in {0,…,n-1}:
    t := F(k, m_i ⊕ t)
  return t
```

If F is a secure PRF, then CBC-MAC is a secure MAC for messages of length nλ

CBC-MAC

```
tag(k, m_0, …, m_{n-1}):
  t := 0^λ
  for i in {0,…,n-1}:
    t := F(k, m_i ⊕ t)
  return t
```

If F is a secure PRF, then CBC-MAC is a secure MAC for messages of length nλ

If F is a secure PRF, then CBC-MAC is a secure **PRF** for messages of length nλ

CBC-MAC

```
tag(k, m_0, …, m_{n-1}):
  t := 0^λ
  for i in {0,…,n-1}:
    t := F(k, m_i ⊕ t)
  return t
```

If F is a secure PRF, then CBC-MAC is a secure MAC for messages of length nλ

If F is a secure PRF, then CBC-MAC is a secure **PRF** for messages of length nλ

What about if messages are variable length?

CBC-MAC

```
tag(k, m₀, …, mₙ₋₁):
  t := 0^λ
  for i in {0,…,n-1}:
    t := F(k, mᵢ ⊕ t)
  return t
```

CBC-MAC is not secure for variable-length messages

If F is a secure PRF, then CBC-MAC is a secure MAC for messages of length nλ

If F is a secure PRF, then CBC-MAC is a secure MAC for messages of length nλ

What about messages are variable

ECBC-MAC

```
KeyGen():
   k0 ← {0,1}^λ
   k1 ← {0,1}^λ
   return (k0, k1)

tag((k0, k1), m0, …, mn-1):
   t := 0^λ
   for i in {0,…,n-2}:
      t := F(k0, mi ⊕ t)
   return F(k1, mn-1 ⊕ t)
```

If F is a secure PRF, then ECBC-MAC is a secure MAC

ECBC-MAC

```
KeyGen():
   k0 ← {0,1}^λ
   k1 ← {0,1}^λ
   return (k0, k1)

tag((k0, k1), m0, …, mn-1):
   t := 0^λ
   for i in {0,…,n-2}:
      t := F(k0, mi ⊕ t)
   return F(k1, mn-1 ⊕ t)
```

If F is a secure PRF, then ECBC-MAC is a secure MAC

Again, one can show ECBC-MAC is a PRF

Authenticated Encryption

A cipher (Enc, Dec) has **security against a chosen ciphertext attack (CCA)** if:

```
k ← K
S ← empty-set

encrypt(m0, m1):
  c ← Enc(k, m0)
  S ← S ∪ {c}
  return c

decrypt(c):
  if c ∈ S:
    return error
  return Dec(k, c)
```

≈

```
k ← K
S ← empty-set

encrypt(m0, m1):
  c ← Enc(k, m1)
  S ← S ∪ {c}
  return c

decrypt(c):
  if c ∈ S:
    return error
  return Dec(k, c)
```

A cipher (Enc, Dec) is an **authenticated encryption (AE) scheme** if:

```
k ← K
S ← empty-set

encrypt(m):
  c ← Enc(k, m)
  S ← S ∪ {c}
  return c

decrypt(c):
  if c ∈ S:
    return error
  return Dec(k, c)
```

≈

```
encrypt(m):
  c ← C
  return c

decrypt(c):
  return error
```

A cipher (Enc, Dec) is an **authenticated encryption (AE) scheme** if:

```
k ← K
S ← empty-set

encrypt(m):
  c ← Enc(k, m)
  S ← S ∪ {c}
  return c

decrypt(c):
  if c ∈ S:
    return error
  return Dec(k, c)
```

≈

```
encrypt(m):
  c ← C
  return c

decrypt(c):
  return error
```

In fact, we already saw Encrypt-then-MAC achieves this

A cipher (Enc, Dec) is an **authenticated encryption scheme with associated data (AEAD)** if:

```
k ← K
S ← empty-set

encrypt(d, m):
  c ← Enc(k, d, m)
  S ← S ∪ {(d, c)}
  return c

decrypt(d, c):
  if (d, c) ∈ S:
    return error
  return Dec(k, d, c)
```

d is from some set of "associated data"

Each d should be distinct

≈

```
encrypt(d, m):
  c ← C
  return c

decrypt(d, c):
  return error
```

This is the "gold standard" in symmetric-key security

A cipher (Enc, Dec) is an **authenticated encryption scheme with associated data (AEAD)** if:

“Associated data” — any non-private information that establishes a unique “context” (a nonce) for a ciphertext

- **Timestamps**
- **Session IDs/User IDs**
- **Message header in some internet protocol**
- **Message ID**
- **…**

Today's objectives

Revisit definition of Message Authentication Codes (MACs)

Achieve MACs for long messages

Explain a scheme that *does not* work

Explain a scheme that *does* work

Discuss strong notions of security that achieve authentication