CPA Security, Continued

CS/ECE 407

Attendance — Questions are optional and not graded

Sign in with illinois.edu email

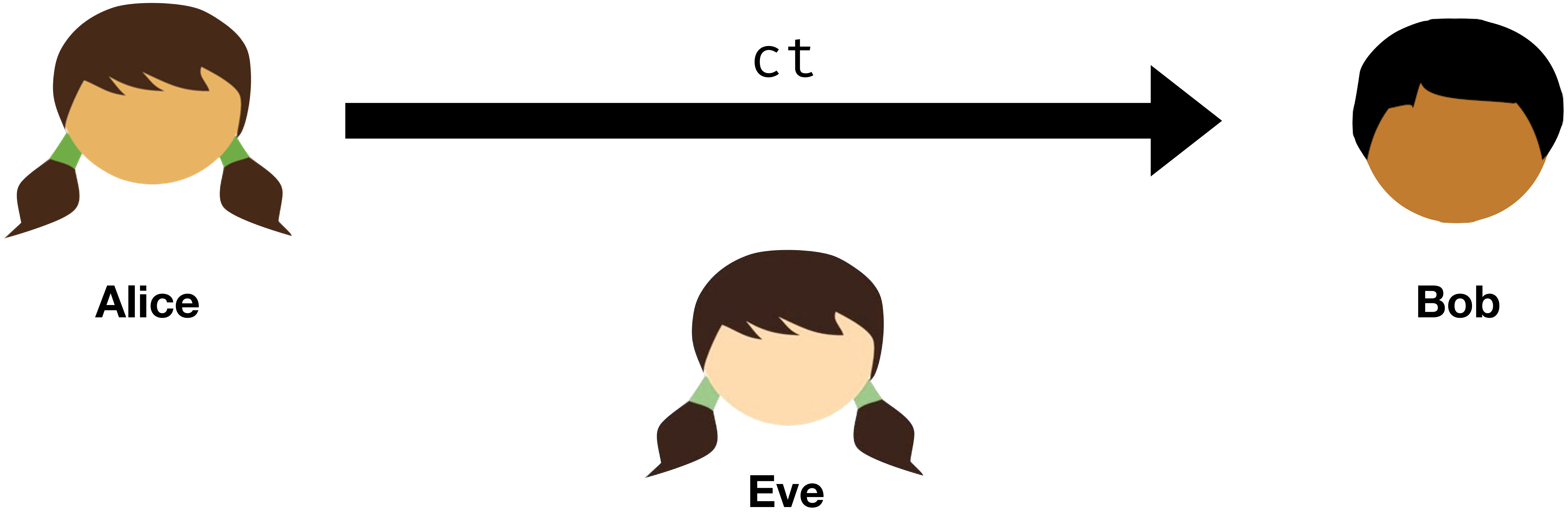
Today's objectives

Recall notion of CPA security

Discuss difference between sampling with/ without replacement

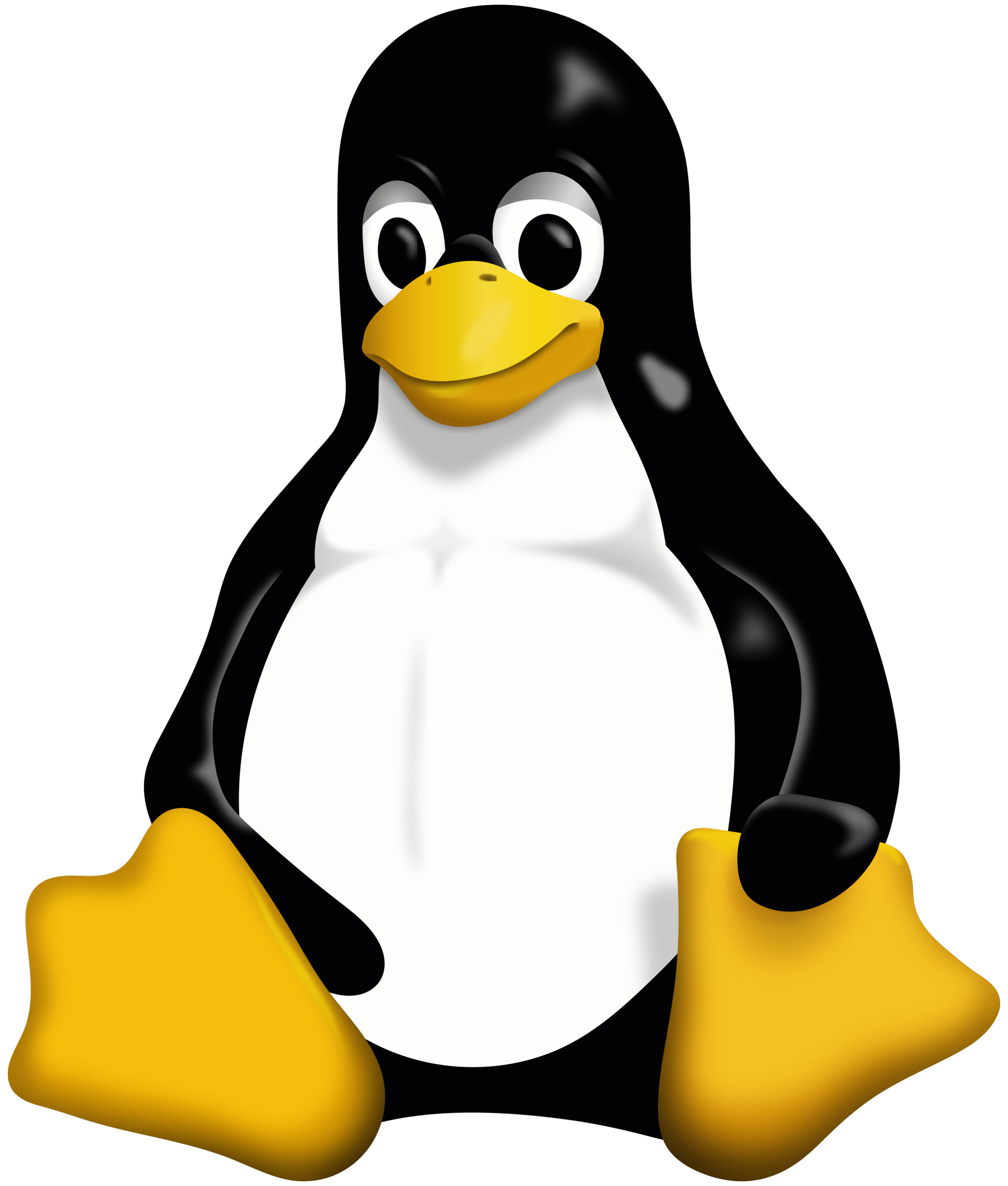
Construct CPA-secure schemes

Prove a randomized scheme is secure

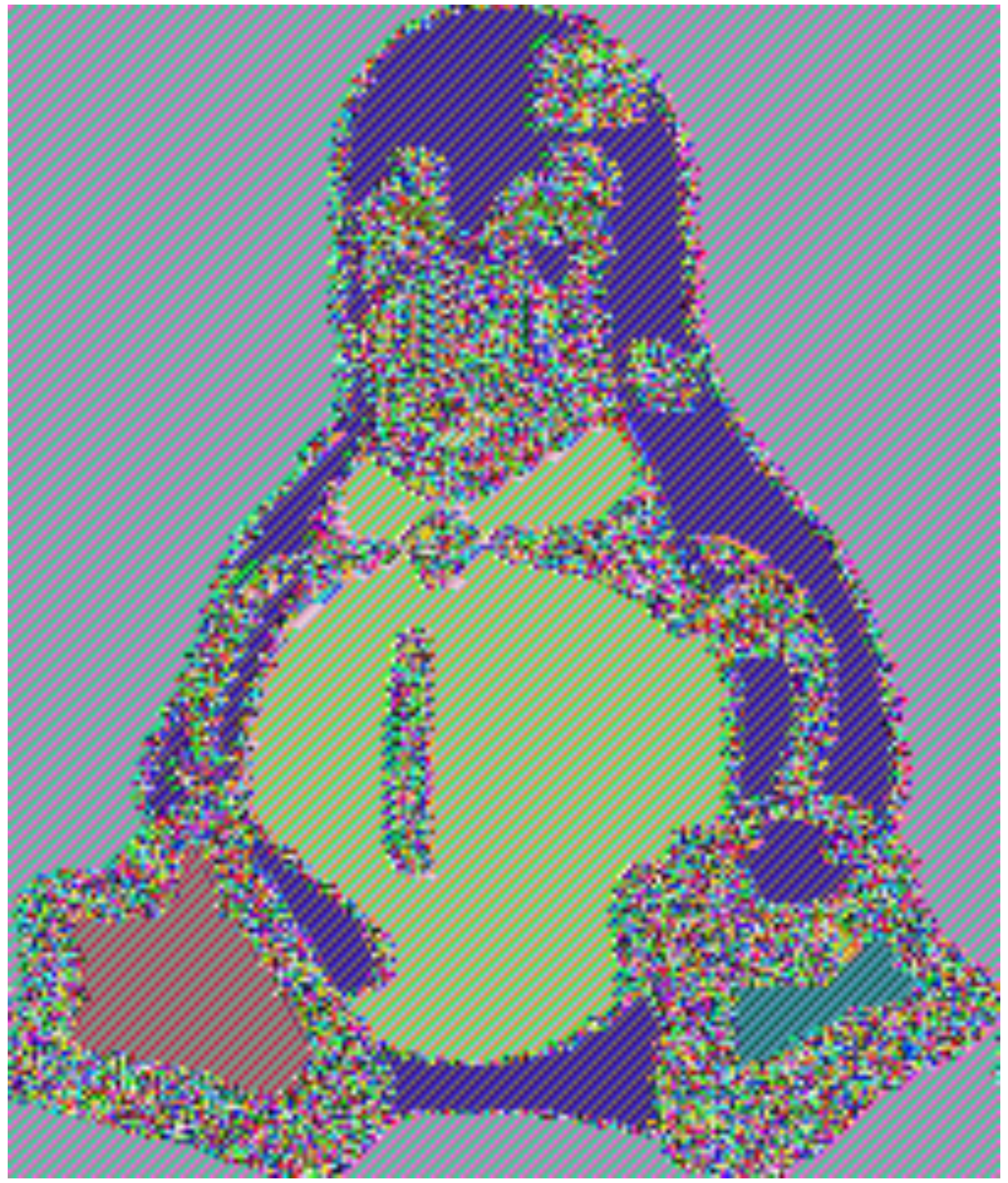

ct
Alice
Eve
Bob

Deterministic Encryption

A cipher (Enc, Dec) is **deterministic** if calling $Enc(k, m)$ on the same inputs twice always produces the same output



“Good” encryption



Naive use of one-time semantically-secure encryption

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

Let $Enc_L(k, m_0, m_1) = Enc(k, m_0)$

Let $Enc_R(k, m_0, m_1) = Enc(k, m_1)$

Where m_0, m_1 are of the same length

$$\left\{ Enc_L(k, \cdot\,, \cdot\,) \;\middle|\; k \leftarrow K \right\} \approx \left\{ Enc_R(k, \cdot\,, \cdot\,) \;\middle|\; k \leftarrow K \right\}$$

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

$$Samp(m) = \{c \mid c \leftarrow C(|m|)\}$$

Ciphertext of length corresponding to message m

$$\{Enc(k, \cdot) \mid k \leftarrow K\} \approx Samp(\cdot)$$

Deterministic encryption does not work — what now?

Randomized:
Cipher samples randomness for each encryption

Statefulness:
Cipher keeps internal state to ensure encryptions are different

Nonce-based:
Alice and Bob pass extra “use-once” values to the Enc/Dec function (basically, Alice and Bob maintain a state on behalf of the cipher)

$$F : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^m$$

F is called a **pseudorandom function family** if the following indistinguishability holds:

$$\left\{ F(k, \cdot\,) \;\middle|\; k \leftarrow \{0,1\}^\lambda \right\} \approx \left\{ f \;\middle|\; f \leftarrow \text{uniform function from } \{0,1\}^n \to \{0,1\}^m \right\}$$

Uniformly sampling k “emulates” a huge random table

Randomized CPA-Secure Encryption

```
Enc(k, m):
  r ← {0,1}^λ
  c0 = F(k, r) ⊕ m
  c = (c0, r)
  return c

Dec(k, (c0, r)):
  return F(k, r) ⊕ c0
```

Main idea: it is unlikely that `Enc` will sample the same r more than once

Sampling With/Without Replacement

```
Samp():
  r ←{0,1}^λ
  return r
```

$\approx$

```
S ← empty-set

Samp():
  r ←{0,1}^λ \ S
  S ← S ∪ { r }
  return r
```

Suppose Adv makes q queries and let $N = 2^\lambda$. What is the chance they observe a collision on the left?

$$Birthday(q, N) = 1 - \prod_{i=0}^{q-1} \left(1 - \frac{i}{N}\right)$$

Suppose Adv makes q queries and let $N = 2^{\lambda}$. What is the chance they observe a collision on the left?

$$Birthday(q, N) = 1 - \prod_{i=0}^{q-1} \left(1 - \frac{i}{N}\right)$$

probability query i is not a collision, given no previous queries collided

$$Birthday(q, N) = 1 - \prod_{i=0}^{q-1}\left(1 - \frac{i}{N}\right)$$

Assume $q \leq \sqrt{2N}$**.** Then:

$$0.632\frac{q(q-1)}{2N} \leq Birthday(q, N) \leq \frac{q(q-1)}{2N}$$

Sampling With/Without Replacement

```
Samp():
  r ←{0,1}^λ
  return r
```

$\approx$

```
S ← empty-set

Samp():
  r ←{0,1}^λ \ S
  S ← S ∪ { r }
  return r
```

Note that $\sqrt{2N} = \sqrt{2 \cdot 2^\lambda} = 2^{(\lambda+1)/2}$, which is exponential. Therefore no polytime adversary can issue $\sqrt{2N}$ queries, so we can apply the bound

Sampling With/Without Replacement

```
Samp():
  r ←{0,1}^λ
  return r
```

$\approx$

```
S ← empty-set

Samp():
  r ←{0,1}^λ \ S
  S ← S ∪ { r }
  return r
```

Any poly time adversary issuing q queries has advantage at most $O(q^2/2^\lambda)$ to distinguish, which is negligible

Randomized CPA-Secure Encryption

```
Enc(k, m):
  r ←{0,1}^λ
  c0 = F(k, r) ⊕ m
  c = (c0, r)
  return c

Dec(k, (c0, r)):
  return F(k, r) ⊕ c0
```

Main idea: it is unlikely that `Enc` will sample the same r more than once

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

$$Samp(m) = \left\{ c \,|\, c \leftarrow C(\,|\,m\,|\,) \right\}$$

Ciphertext of length corresponding to message m

$$\left\{ Enc(k, \cdot\,) \;\middle|\; k \leftarrow K \right\} \approx Samp(\,\cdot\,)$$

```
k ← {0,1}^λ
Oracle(m):
  r ← {0,1}^λ
  c0 = F(k, r) ⊕ m
  c = (c0, r)
  return c
```

```
k ← {0,1}^λ
Oracle(m):
  r ← {0,1}^λ
  c0 = F(k, r) ⊕ m
  c = (c0, r)
  return c
```

$$\approx$$

```
k ← {0,1}^λ
S ← empty-set
Oracle(m):
  r ← {0,1}^λ \ S
  S ← S ∪ { r }
  c0 = F(k, r) ⊕ m
  c = (c0, r)
  return c
```

```
k ← {0,1}^λ
Oracle(m):
  r ← {0,1}^λ
  c0 = F(k, r) ⊕ m
  c = (c0, r)
  return c
```

≈ by birthday bound

```
k ← {0,1}^λ
S ← empty-set
Oracle(m):
  r ← {0,1}^λ \ S
  S ← S ∪ { r }
  c0 = F(k, r) ⊕ m
  c = (c0, r)
  return c
```

```
k ← {0,1}^λ
Oracle(m):
   r ← {0,1}^λ
   c0 = F(k, r) ⊕ m
   c = (c0, r)
   return c
```

≈ by birthday bound

```
k ← {0,1}^λ
S ← empty-set
Oracle(m):
   r ← {0,1}^λ \ S
   S ← S ∪ { r }
   c0 = F(k, r) ⊕ m
   c = (c0, r)
   return c
```

≈

```
f ← uniform function
S ← empty-set
Oracle(m):
   r ← {0,1}^λ \ S
   S ← S ∪ { r }
   c0 = f(r) ⊕ m
   c = (c0, r)
   return c
```

```
k ← {0,1}^λ
Oracle(m):
   r ← {0,1}^λ
   c0 = F(k, r) ⊕ m
   c = (c0, r)
   return c
```

≈ by birthday bound

```
k ← {0,1}^λ
S ← empty-set
Oracle(m):
   r ← {0,1}^λ \ S
   S ← S ∪ { r }
   c0 = F(k, r) ⊕ m
   c = (c0, r)
   return c
```

≈ by PRF security

```
f ← uniform function
S ← empty-set
Oracle(m):
   r ← {0,1}^λ \ S
   S ← S ∪ { r }
   c0 = f(r) ⊕ m
   c = (c0, r)
   return c
```

```
f ← uniform function
S ← empty-set
Oracle(m):
  r ← {0,1}^λ \ S
  S ← S ∪ { r }
  c0 = f(r) ⊕ m
  c = (c0, r)
  return c
```

```
f ← uniform function
S ← empty-set
Oracle(m):
  r ← {0,1}^λ \ S
  S ← S ∪ { r }
  c0 = f(r) ⊕ m
  c = (c0, r)
  return c
```

≡

```
S ← empty-set
Oracle(m):
  r ← {0,1}^λ \ S
  S ← S ∪ { r }
  r' ← {0,1}^|m|
  c0 = r' ⊕ m
  c = (c0, r)
  return c
```

```
f ← uniform function
S ← empty-set
Oracle(m):
  r ← {0,1}^λ \ S
  S ← S ∪ { r }
  c0 = f(r) ⊕ m
  c = (c0, r)
  return c
```

≡

f is uniform, no row used more than once

```
S ← empty-set
Oracle(m):
  r ← {0,1}^λ \ S
  S ← S ∪ { r }
  r' ← {0,1}^|m|
  c0 = r' ⊕ m
  c = (c0, r)
  return c
```

```
f ← uniform function
S ← empty-set
Oracle(m):
  r ← {0,1}^λ \ S
  S ← S ∪ { r }
  c0 = f(r) ⊕ m
  c = (c0, r)
  return c
```

≡

f is uniform, no row used more than once

```
S ← empty-set
Oracle(m):
  r ← {0,1}^λ \ S
  S ← S ∪ { r }
  r' ← {0,1}^|m|
  c0 = r' ⊕ m
  c = (c0, r)
  return c
```

r' is a one-time pad ≡

```
S ← empty-set
Oracle(m):
  r ← {0,1}^λ \ S
  S ← S ∪ { r }
  c0 ← {0,1}^|m|
  c = (c0, r)
  return c
```

```
f ← uniform function
S ← empty-set
Oracle(m):
  r ← {0,1}^λ \ S
  S ← S ∪ { r }
  c0 = f(r) ⊕ m
  c = (c0, r)
  return c
```

≡

f is uniform, no row used more than once

```
S ← empty-set
Oracle(m):
  r ← {0,1}^λ \ S
  S ← S ∪ { r }
  r' ← {0,1}^|m|
  c0 = r' ⊕ m
  c = (c0, r)
  return c
```

r' is a one-time pad ≡

```
S ← empty-set
Oracle(m):
  r ← {0,1}^λ \ S
  S ← S ∪ { r }
  c0 ← {0,1}^|m|
  c = (c0, r)
  return c
```

≈

by birthday bound

```
Oracle(m):
  r ← {0,1}^λ
  c0 ← {0,1}^|m|
  c = (c0, r)
  return c
```

```
Oracle(m):
  r ← {0,1}^λ
  c0 ← {0,1}^|m|
  c = (c0, r)
  return c
```

```
Oracle(m):
  r ← {0,1}^λ
  c0 ← {0,1}^|m|
  c = (c0, r)
  return c
```

$\equiv$

```
Samp(m):
  c ← C
  return c
```

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

$$Samp(m) = \{c \,|\, c \leftarrow C(|m|)\}$$

Ciphertext of length corresponding to message m

$$\{Enc(k, \cdot) \;\;|\;\; k \leftarrow K\} \approx Samp(\cdot)$$

Stateful CPA-Secure Encryption

```
Enc(k, m):
  global counter ← 0
  c0 ← F(k, counter) ⊕ m
  c ← (c0, counter)
  counter ← counter + 1
  return c

Dec(k, (c0, counter)):
  return F(k, counter) ⊕ c0
```

Nonce-based CPA-Secure Encryption

```
Enc(k, nonce, m):
  c0 ← F(k, nonce) ⊕ m
  c ← (c0, nonce)
  return c

Dec(k, (c0, nonce)):
  return F(k, nonce) ⊕ c0
```

Requires changing slightly the definition of CPA security:

Adversary is not allowed to call encrypt with same nonce more than once

Nonce — “number used once”

Nonce-based definition requires slight change to CPA security definition

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

⬇

```
EncL(k, nonce, m0, m1):
  global S ← empty-set
  if nonce in S: return "FAIL"
  else:
    S ← S ∪ { nonce }
    return Enc(k, nonce, m0)
```

Adversary is not allowed to call encrypt with same nonce more than once

Modern Cryptography

State assumptions

Define security

Design system

Prove: if assumption holds, system meets definition

Modern Cryptography

State assumptions

F is a PRF

Define security

Design system

Prove: if assumption holds, system meets definition

Modern Cryptography

State assumptions

F is a PRF
OWFs exist

Define security

Design system

Prove: if assumption holds, system meets definition

Modern Cryptography

State assumptions

F is a PRF
OWFs exist

Define security

CPA-secure cipher

Design system

Prove: if assumption holds, system meets definition

Modern Cryptography

State assumptions — F is a PRF; OWFs exist

Define security — CPA-secure cipher

Design system — Randomized cipher

Prove: if assumption holds, system meets definition

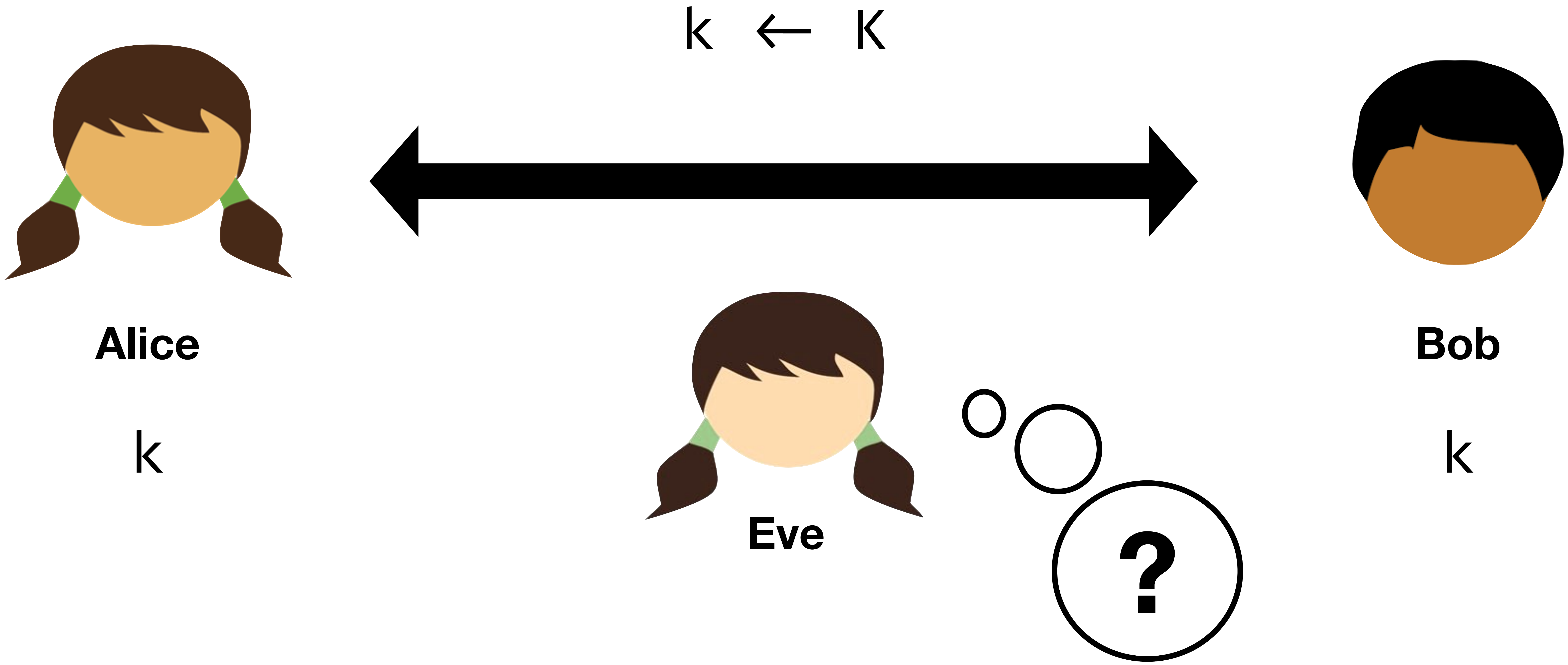
Modern Cryptography

State assumptions — F is a PRF; OWFs exist

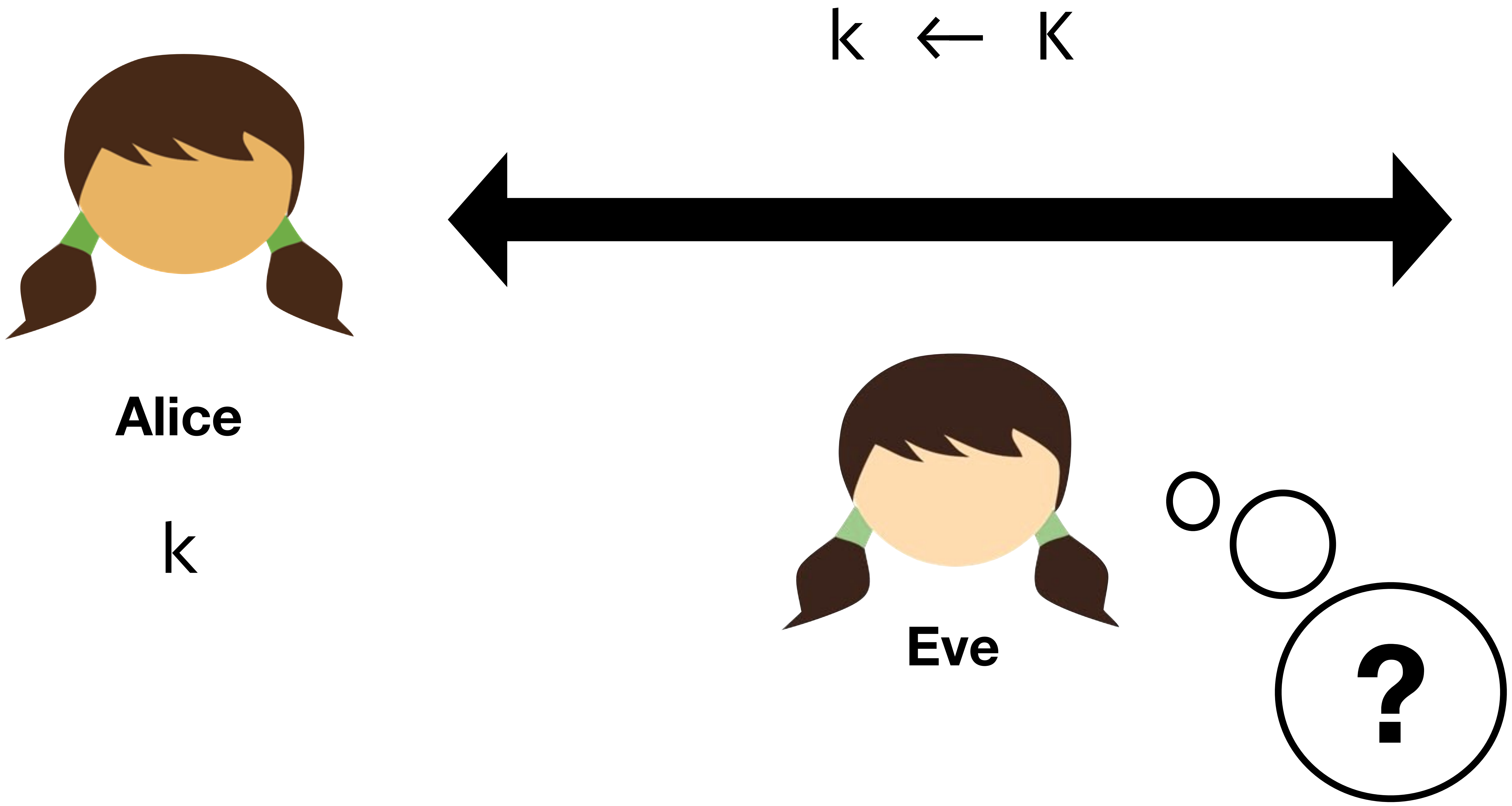
Define security — CPA-secure cipher

Design system — Randomized cipher

Prove: if assumption holds, system meets definition



If F is indeed a PRF, and Alice and Bob have a shared key k, they can indeed communicate essentially without limit in the presence of passive Eve



Next steps:

- Efficiency for long messages — “block cipher modes”
- Authenticity

Today's objectives

Recall notion of CPA security

Discuss difference between sampling with/without replacement

Construct CPA-secure schemes

Prove a randomized scheme is secure