CPA Security Continued

CS/ECE 407

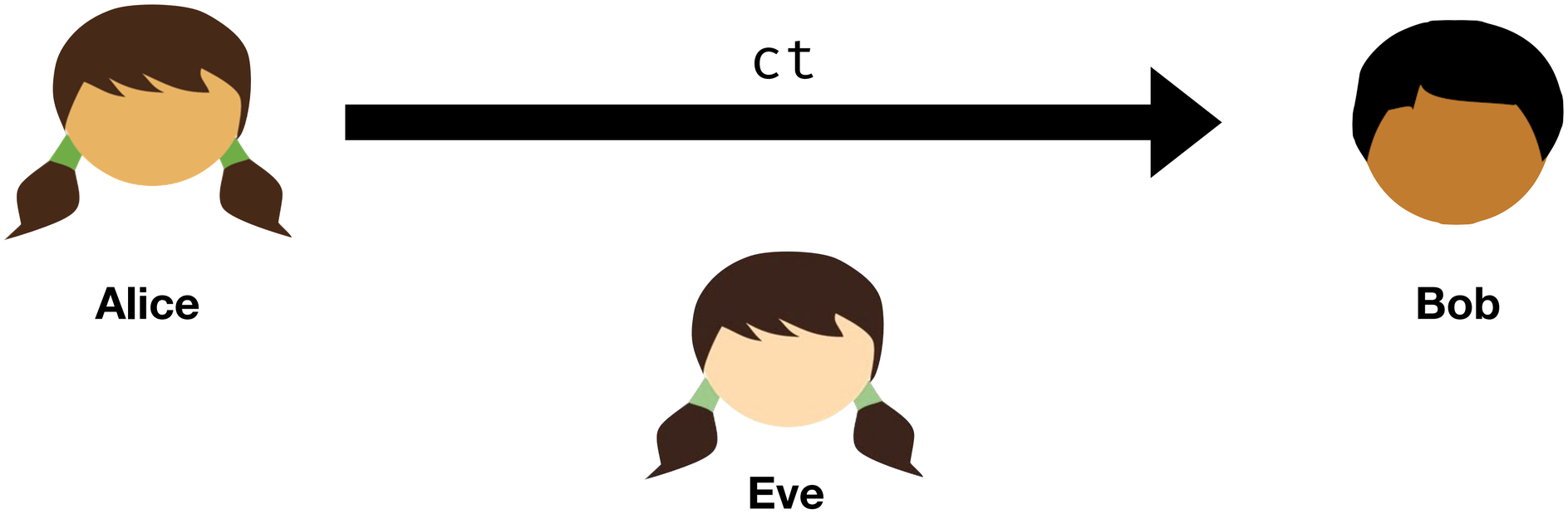
Attendance:

Today's objectives

Examine CPA Security

Understand the limitations of deterministic encryption, see how to circumvent this problem

Construct CPA-secure schemes



A cipher (Enc, Dec) has **one-time semantic security** if:

```
eavesdrop(m0, m1):
  k ←$ {0,1}^λ
  ct ← Enc(k, m0)
  return ct
```

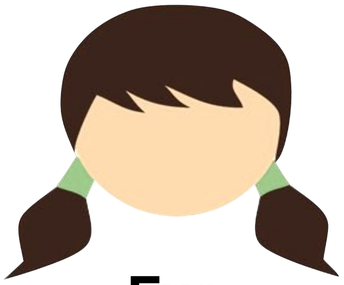
$\overset{c}{\approx}$

```
eavesdrop(m0, m1):
  k ←$ {0,1}^λ
  ct ← Enc(k, m1)
  return ct
```

ct0, ct1

Alice

Bob

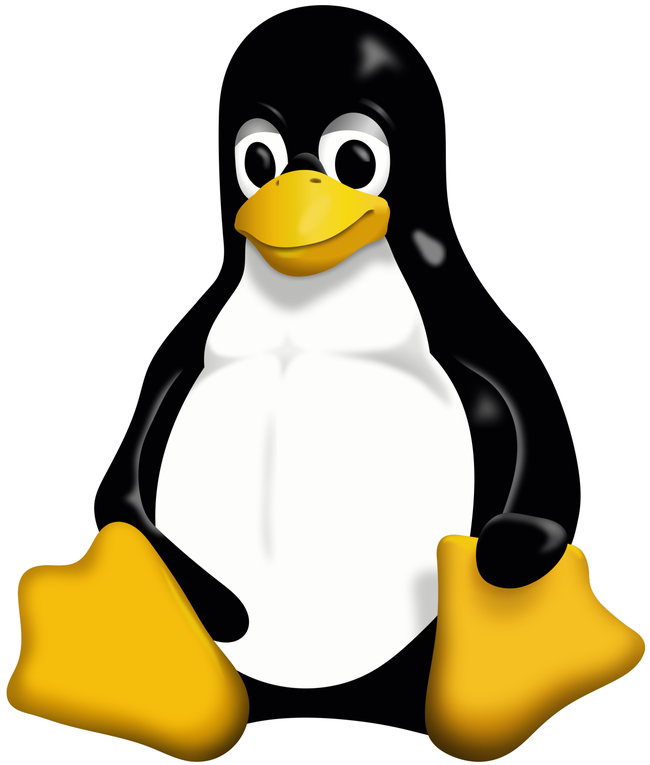


Eve

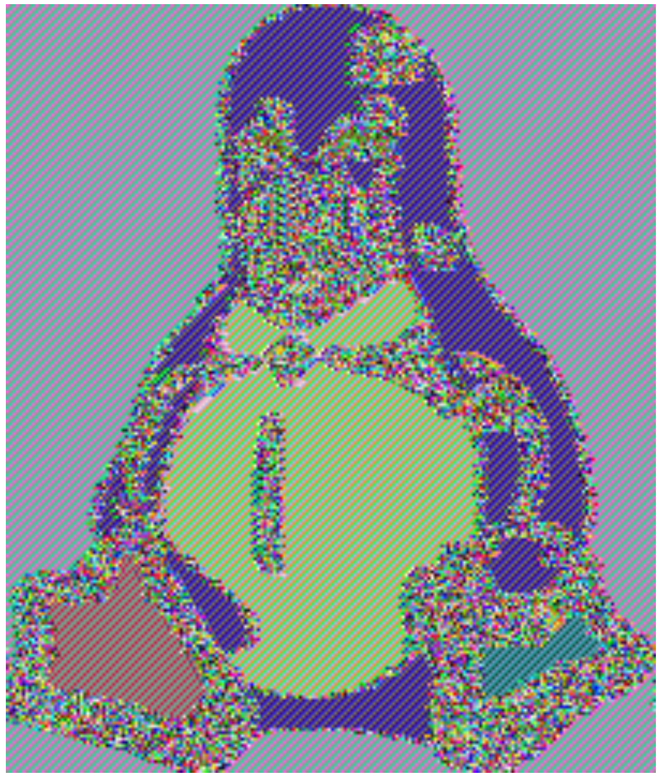
A ciph (Enc) has **one-time s security** if:

```
eavesdrop(m0, m1):
  k ←$ {0,1}^λ
  ct ← Enc(k, m
  return ct
```

```
eavesdrop(m0, m1):
  k ←$ {0,1}^λ
  ct ← Enc(k, m1)
  return ct
```



“Good” encryption



Naive use of one-time semantically-secure encryption

A cipher (Enc, Dec) has **one-time semantic security** if:

```
eavesdrop(m0, m1):
  k ←$ {0,1}^λ
  ct ← Enc(k, m0)
  return ct
```

$\overset{c}{\approx}$

```
eavesdrop(m0, m1):
  k ←$ {0,1}^λ
  ct ← Enc(k, m1)
  return ct
```

A cipher (Enc, Dec) has **one-time semantic security** if:

```
eavesdrop(m0, m1):
  k ←$ {0,1}^λ
  ct ← Enc(k, m0)
  return ct
```

$\overset{c}{\approx}$

```
eavesdrop(m0, m1):
  k ←$ {0,1}^λ
  ct ← Enc(k, m1)
  return ct
```

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

```
k ←$ {0,1}^λ
eavesdrop(m0, m1):
  ct ← Enc(k, m0)
  return ct
```

$\overset{c}{\approx}$

```
k ←$ {0,1}^λ
eavesdrop(m0, m1):
  ct ← Enc(k, m1)
  return ct
```

Deterministic encryption does not work — what now?

Statefulness:
Cipher keeps internal state to ensure encryptions are different

Randomized:
Cipher samples randomness for each encryption

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:

```
k ←$ {0,1}^λ

apply(x):
  return F(k, x)
```

$\overset{c}{\approx}$

```
D ← empty-dictionary

apply(x):
  if x is not in D:
    D[x] ←$ {0,1}^m
  return D[x]
```

Deterministic encryption does not work — what now?

Statefulness:
Cipher keeps internal state to ensure encryptions are different

Randomized:
Cipher samples randomness for each encryption

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)

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
```

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 provably unlikely that Enc will sample the same r more than once

Proof of security is more nuanced here

Related to the *birthday paradox*

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, r) ⊕ c0
```

Requires changing slightly the definition of CPA security:

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

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

```
k ←$ {0,1}^λ
eavesdrop(m0, m1):
  ct ← Enc(k, m0)
  return ct
```

$\overset{c}{\approx}$

```
k ←$ {0,1}^λ
eavesdrop(m0, m1):
  ct ← Enc(k, m1)
  return ct
```

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

```
k ←$ {0,1}^λ
S ← empty-set
eavesdrop(nonce, m0, m1):
  if nonce in S:
    return error
  insert nonce to S
  ct ← Enc(k, nonce, m0)
  return ct
```

$\overset{c}{\approx}$

```
k ←$ {0,1}^λ
S ← empty-set
eavesdrop(nonce, m0, m1):
  if nonce in S:
    return error
  insert nonce to S
  ct ← Enc(k, nonce, m1)
  return ct
```

Today's objectives

Examine CPA Security

Understand the limitations of deterministic encryption, see how to circumvent this problem

Construct CPA-secure schemes