Public-Key Encryption Continued

CS/ECE 407

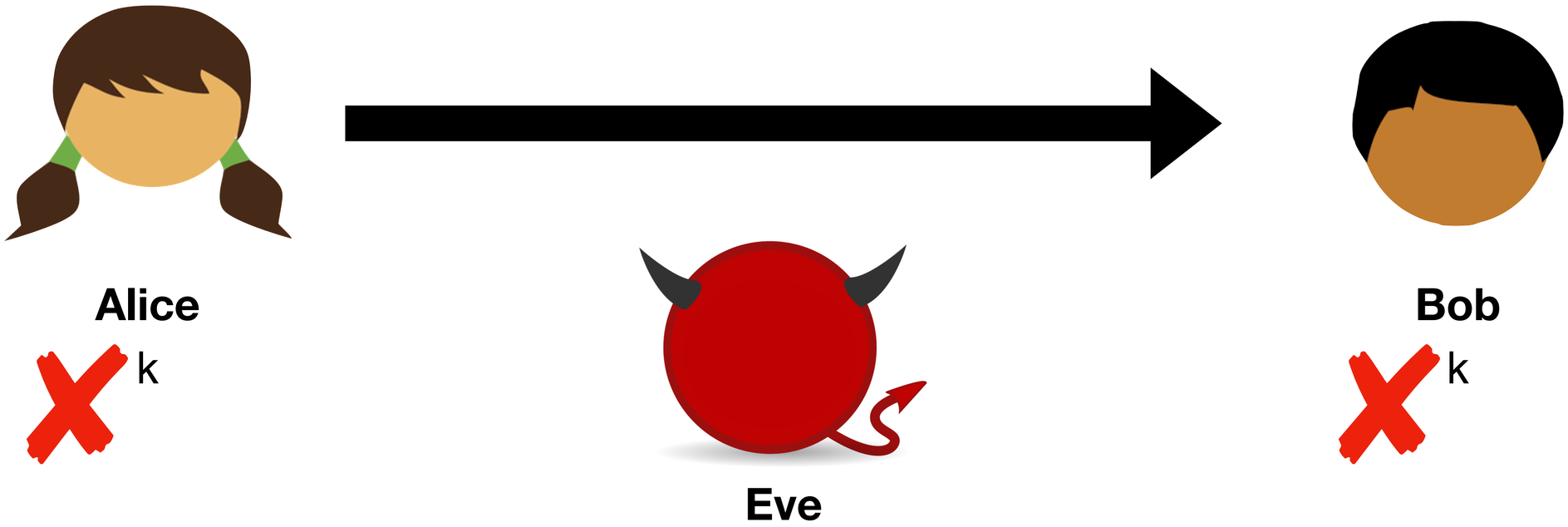
Today's objectives

Review **one-time secrecy** and **CPA security**

Prove that for PK encryption, OTS implies CPA security

Combine public-key and symmetric-key with *hybrid* encryption scheme and key encapsulation mechanisms

Incorporate random oracle to more easily combine schemes



Public Key Cryptography:

Can Alice and Bob securely communicate, even if they are speaking for the very first time?

ElGamal Public Key Encryption

Let g be the generator of some cyclic group G of order q

```
KeyGen():
  sk ←$ Z_q
  pk ← g^sk
  return (pk, sk)
```

```
Enc(pk, m ∈ G):
  y ←$ Z_q
  return (g^y, m·pk^y)
```

```
Dec(sk, (c_1, c_2)):
  s ← c_1^sk
  return c_2·s^-1
```

Public key encryption scheme (KeyGen, Enc, Dec) has **one-time secrecy** if:

```
(pk, sk) ← KeyGen()
count ← 0

key():
  return pk

encrypt(m0, m1):
  if count > 0:
    return error
  count ← count + 1
  ct ← Enc(pk, m0)
  return ct
```

≈

```
(pk, sk) ← KeyGen()
count ← 0

key():
  return pk

encrypt(m0, m1):
  if count > 0:
    return error
  count ← count + 1
  ct ← Enc(pk, m1)
  return ct
```

Public key encryption scheme (KeyGen, Enc, Dec) has **CPA security** if:

```
(pk, sk) ← KeyGen()

key(): return pk

encrypt(m0, m1):
  ct ← Enc(pk, m0)
  return ct
```

≈

```
(pk, sk) ← KeyGen()

key(): return pk

encrypt(m0, m1):
  ct ← Enc(pk, m1)
  return ct
```

For public-key schemes,

one-time secrecy $\Longrightarrow$ CPA security

This is **not true** for symmetric-key schemes!!

Hybrid-#bound

```
pk ← key()
count ← 0

key(): return pk

encrypt(m0, m1):
  count ← count + 1
  if count < bound:
    return Enc(pk, m1)
  else if count = bound:
    return OTS(m0, m1)
  else:
    return Enc(pk, m0)
```

◇ **OTS.Left/OTS.Right**

Hybrid Encryption

Rule of thumb — in terms of cost:

`information-theoretic << symmetric-key << public-key`

Think: orders of magnitude slowdown

Public Key Encryption

A public-key encryption scheme is a triple `(KeyGen, Enc, Dec)`

`KeyGen()` outputs a **key pair** `(pk, sk)`

`Enc(pk, m)` $\rightarrow$ `c`

`Dec(sk, c)` $\rightarrow$ `m`

Correctness: For all `m`:

$$\Pr\left[\mathrm{Dec}(sk, c) = m \;\middle|\; \begin{array}{l}(pk, sk) \leftarrow \mathrm{KeyGen}(1^\lambda) \\ c \leftarrow \mathrm{Enc}(pk, m)\end{array}\right] > 1 - \mathrm{negl}(\lambda)$$

Key Encapsulation Mechanism

A public-key encryption scheme is a triple `(KeyGen, Enc, Dec)`

`KeyGen()` outputs a **key pair** `(pk, sk)`

`Enc(pk, m)` $\rightarrow$ `c`

`Dec(sk, c)` $\rightarrow$ `m`

Correctness:

$$\Pr\left[\mathrm{Decaps}(sk, c) = k \;\middle|\; \begin{array}{l}(pk, sk) \leftarrow \mathrm{KeyGen}(1^{\lambda}) \\ (c, k) \leftarrow \mathrm{Encaps}(pk)\end{array}\right] > 1 - \mathrm{negl}(\lambda)$$

Key Encapsulation Mechanism
(KeyGen, Encaps, Decaps) has **CPA security** if:

$$\left\{ (\mathrm{pk}, c, k) \;\middle|\; \begin{array}{l} (\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{KeyGen}(1^\lambda) \\ (c, k) \leftarrow \mathrm{Encaps}(\mathrm{pk}) \end{array} \right\} \approx \left\{ (\mathrm{pk}, c, k) \;\middle|\; \begin{array}{l} (\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{KeyGen}(1^\lambda) \\ (c, \cdot) \leftarrow \mathrm{Encaps}(\mathrm{pk}) \\ k \leftarrow \{0,1\}^\lambda \end{array} \right\}$$

KEM to CPA Public Key Scheme

Let $\Pi_0 = (\text{KeyGen}, \text{Encaps}, \text{Decaps})$ be a CPA-secure KEM
Let $\Pi_1 = (\text{Enc}, \text{Dec})$ be a CPA-secure symmetric-key scheme

```
KeyGen(1^λ): return Π₀.KeyGen(1^λ)
```

```
Enc(pk, m):
  (c₁, k) ← Π₀.Encaps(pk)
  c₂ ← Π₁.Enc(k, m)
  return (c₁, c₂)
```

```
Dec(sk, (c₁, c₂)):
  k ← Π₀.Decaps(sk, c₁)
  return Π₁.Dec(sk, c₂)
```

Today's objectives

Review **one-time secrecy** and **CPA security**

Prove that for PK encryption, OTS implies CPA security

Combine public-key and symmetric-key with *hybrid* encryption scheme and key encapsulation mechanisms

Incorporate random oracle to more easily combine schemes