Midterm Review

CS/ECE 407

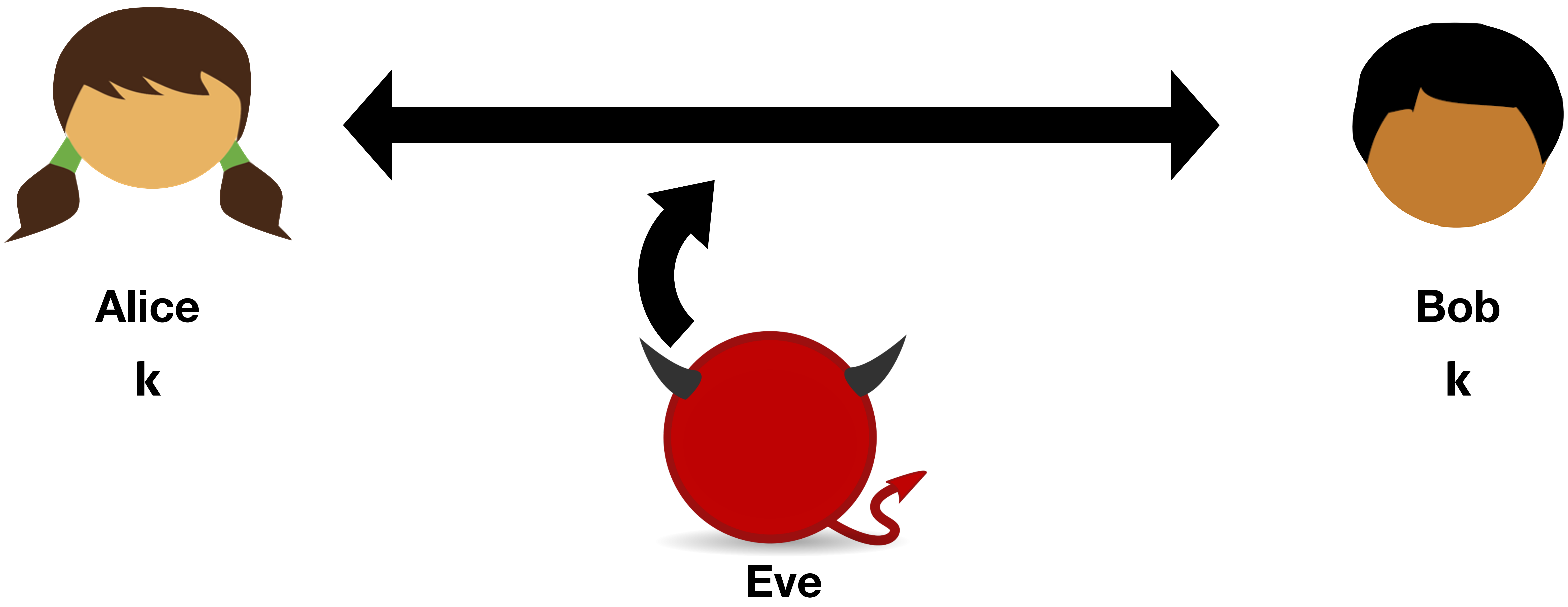
Modern Cryptography

State assumptions

Define security

Design system

Prove: if assumption holds, system meets definition

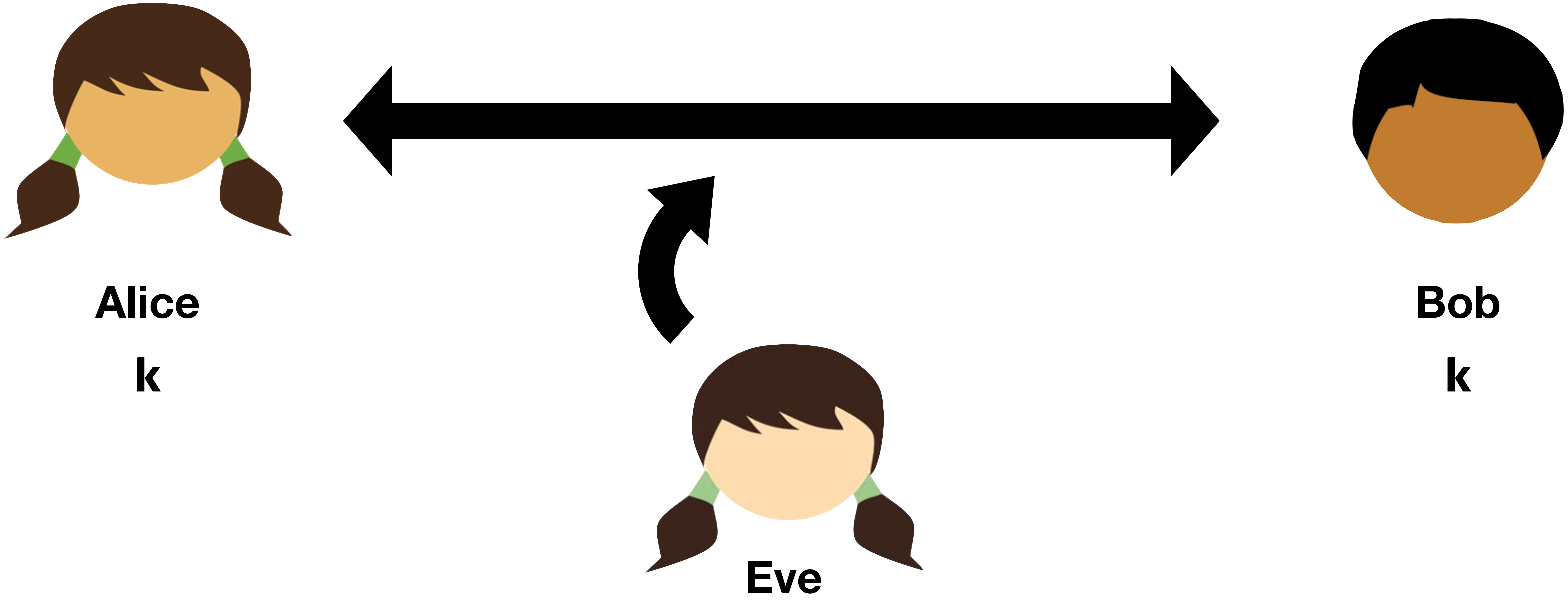


Confidentiality

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?



Confidentiality

Can Alice and Bob prevent Eve from listening?

Symmetric Cipher

A **cipher** over $(\mathrm{K}, \mathrm{M}, \mathrm{C})$ is two *algorithms*:

$$Enc : \mathrm{K} \times \mathrm{M} \to \mathrm{C} \qquad Enc(k, m) := k \oplus m$$

$$Dec : \mathrm{K} \times \mathrm{C} \to \mathrm{M} \qquad Dec(k, ct) := k \oplus ct$$

Correctness:

For every message $m \in \mathrm{M}$:

$$\Pr\left[Dec(k, c) = m \;\middle|\; \begin{array}{l} k \leftarrow_\$ K \\ c \leftarrow Enc(k, m) \end{array} \right] = 1 \qquad k \oplus (k \oplus m) = m \checkmark$$

Perfect Secrecy:

For every message $m \in \mathrm{M}$:

$$\left\{ c \;\middle|\; \begin{array}{l} k \leftarrow_\$ K \\ c = Enc(k, m) \end{array} \right\} \equiv \left\{ c \;\middle|\; c \leftarrow_\$ C \right\} \checkmark$$

Perfect Secrecy:

For every message $m \in \mathrm{M}$, the following are identically distributed:

$$\left\{ c \;\middle|\; \begin{array}{l} k \leftarrow_\$ K \\ c = Enc(k, m) \end{array} \right\} \equiv \left\{ c \;\middle|\; c \leftarrow_\$ C \right\}$$

Theorem [Shannon 1949]: *Any cipher achieving perfect secrecy requires that* $|\mathrm{K}| \geq |\mathrm{M}|$*.*

Bad News! We will need another approach!

Key idea: what if we can make something that *looks* random, but actually isn't

Negligible Function

A function μ is **negligible** *if for any positive polynomial p there exists an N such that for all $n > N$:*

$$\mu(n) < \frac{1}{p(n)}$$

“μ approaches zero really fast”

Indistinguishability

Let X, Y be two probability ensembles, and let A be an arbitrary (probabilistic) program that outputs 0 or 1. A's **advantage** is as follows:

$$\text{Advantage}_A(\lambda) = \left| \Pr\left[b = 1 \;\middle|\; \begin{array}{l} x \leftarrow_\$ X_\lambda \\ b \leftarrow A(1^\lambda, x) \end{array} \right] - \Pr\left[b = 1 \;\middle|\; \begin{array}{l} y \leftarrow_\$ Y_\lambda \\ b \leftarrow A(1^\lambda, y) \end{array} \right] \right|$$

We say that X, Y are **indistinguishable,** written $X \approx Y$ if for every polynomial-time program A:

$\text{Advantage}_A(\lambda)$ is negligible

best strategy is only negligibly better than guessing

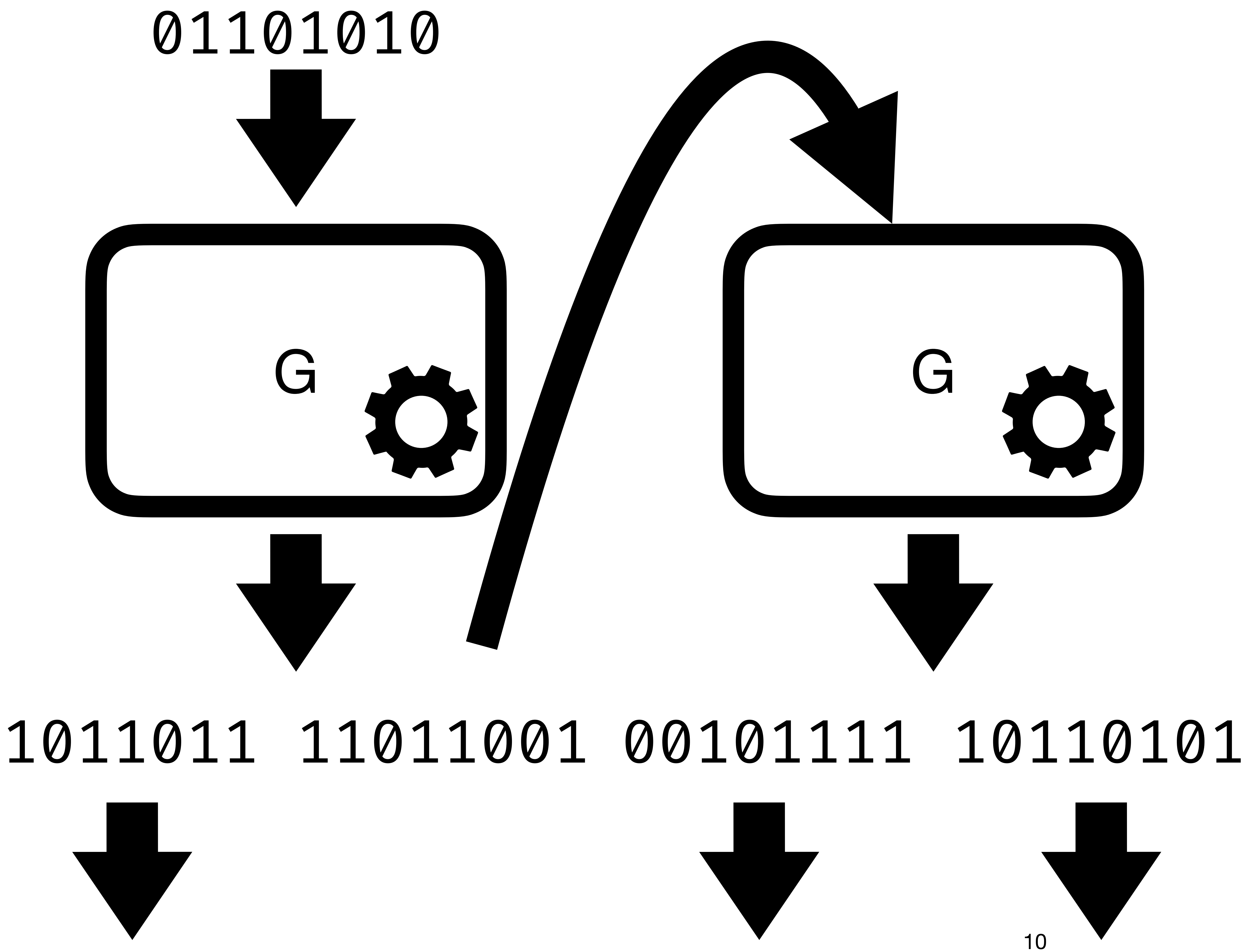
PRG security

Let G be a poly-time deterministic algorithm that on an input of length λ outputs a string of length $\lambda + s(\lambda)$.

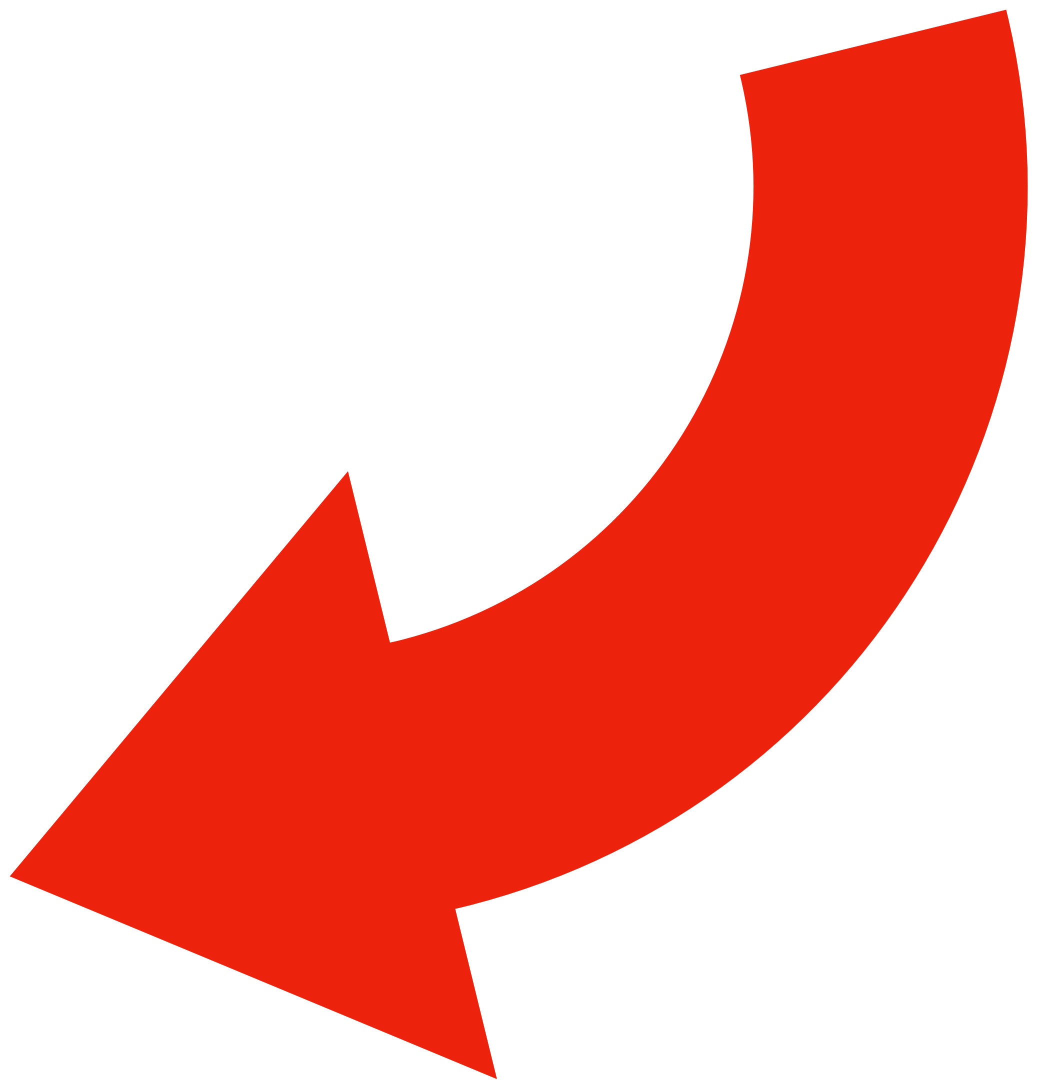
G is a PRG if $s(\lambda)$ is always positive, and:

$$\left\{ G(k) \;\middle|\; k \leftarrow_{\$} \{0,1\}^{\lambda} \right\}_{\lambda} \approx \left\{ r \;\middle|\; r \leftarrow_{\$} \{0,1\}^{\lambda+s(\lambda)} \right\}_{\lambda}$$

Stretching the output of a PRG



This is a secure PRG



Security Reduction

$\texttt{G}$ is a PRG $\Longrightarrow$ $\texttt{G'}$ is a PRG

$\texttt{G}$ is a not PRG $\Longleftarrow$ $\texttt{G'}$ is not a PRG

Let G be a length-doubling PRG

$$G'(k) = \left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' = G(k) \\ r_1, r_2 = G(k') \end{array} \right\}$$

Claim: G' is a PRG

G' is a PRG means:

$$\left\{ G'(k) \;\middle|\; k \leftarrow \{0,1\}^{\lambda} \right\} \approx \left\{ r \;\middle|\; r \leftarrow \{0,1\}^{3\lambda} \right\}$$

$$\left\{ G'(k) \;\middle|\; k \leftarrow \{0,1\}^{\lambda} \right\}$$

$$\left\{ G'(k) \;\middle|\; k \leftarrow \{0,1\}^\lambda \right\} \equiv \left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{c} k \leftarrow \{0,1\}^\lambda \\ r_0, k' = G(k) \\ r_1, r_2 = G(k') \end{array} \right\}$$

By definition of G'

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} k \leftarrow \{0,1\}^\lambda \\ r_0, k' = G(k) \\ r_1, r_2 = G(k') \end{array} \right\}$$

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} k \leftarrow \{0,1\}^\lambda \\ r_0, k' = G(k) \\ r_1, r_2 = G(k') \end{array} \right\}$$

$$\equiv$$

Refactoring

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' \leftarrow \left\{ G(k) \;\middle|\; k \leftarrow \{0,1\}^\lambda \right\} \\ r_1, r_2 = G(k') \end{array} \right\}$$

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' \leftarrow \left\{ G(k) \;\middle|\; k \leftarrow \{0,1\}^\lambda \right\} \\ r_1, r_2 = G(k') \end{array} \right\}$$

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' \leftarrow \left\{ G(k) \;\middle|\; k \leftarrow \{0,1\}^\lambda \right\} \\ r_1, r_2 = G(k') \end{array} \right\}$$

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' \leftarrow \left\{ G(k) \;\middle|\; k \leftarrow \{0,1\}^\lambda \right\} \\ r_1, r_2 = G(k') \end{array} \right\}$$

$\approx$ By PRG security of G

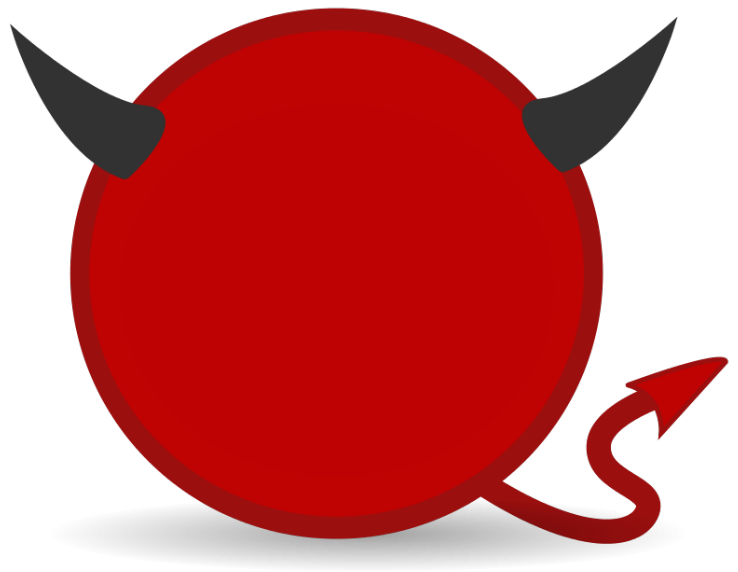
$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' \leftarrow \{0,1\}^{2\lambda} \\ r_1, r_2 = G(k') \end{array} \right\}$$

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' \leftarrow \left\{ G(k) \;\middle|\; k \leftarrow \{0,1\}^\lambda \right\} \\ r_1, r_2 = G(k') \end{array} \right\}$$

$\approx$ By PRG security of G

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' \leftarrow \{0,1\}^{2\lambda} \\ r_1, r_2 = G(k') \end{array} \right\}$$

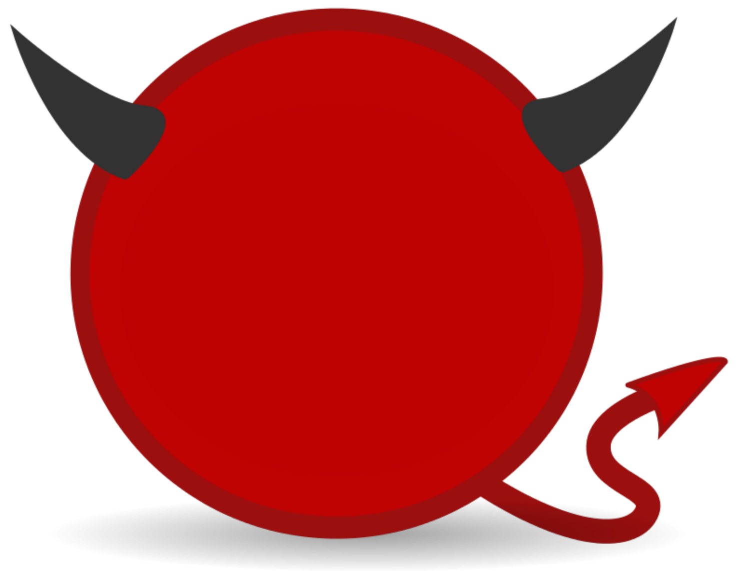
Why does this work?



$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' \leftarrow \left\{ G(k) \;\middle|\; k \leftarrow \{0,1\}^\lambda \right\} \\ r_1, r_2 = G(k') \end{array} \right\}$$

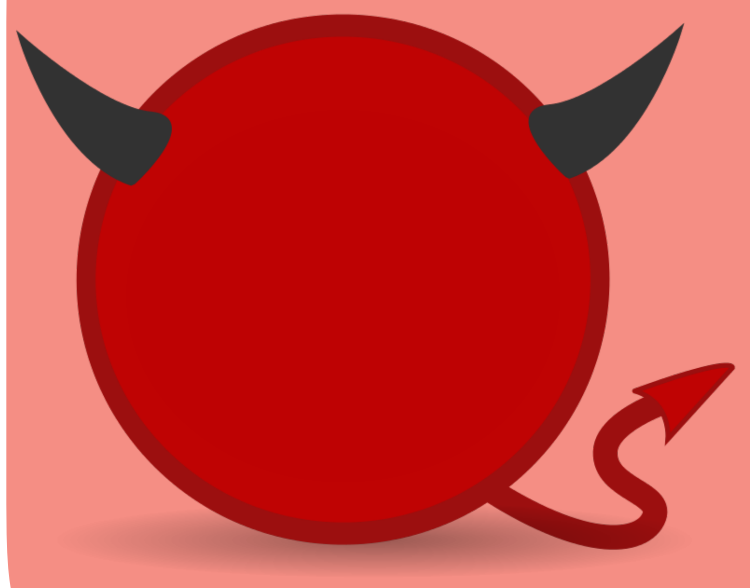
Adversary A

$\approx$ By PRG security of G



$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' \leftarrow \{0,1\}^{2\lambda} \\ r_1, r_2 = G(k') \end{array} \right\}$$

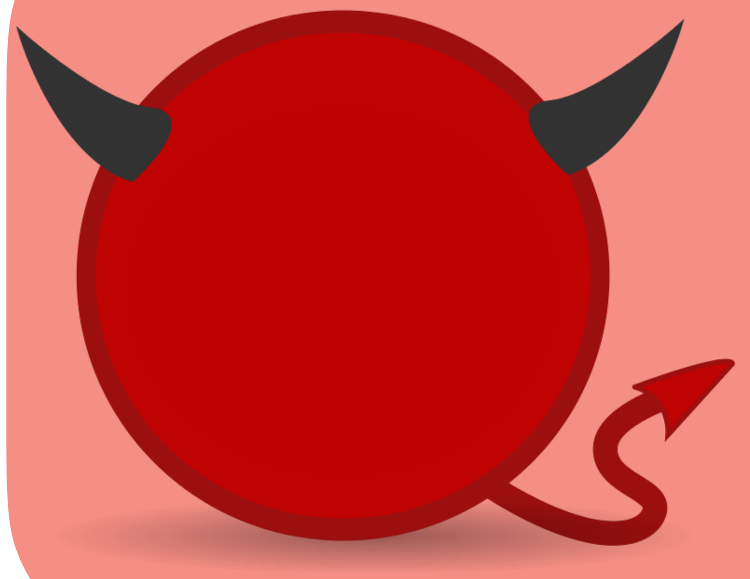
Why does this work?



$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' \leftarrow \left\{ G(k) \;\middle|\; k \leftarrow \{0,1\}^\lambda \right\} \\ r_1, r_2 = G(k') \end{array} \right\}$$

Adversary B

$\approx$ By PRG security of G



$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' \leftarrow \{0,1\}^{2\lambda} \\ r_1, r_2 = G(k') \end{array} \right\}$$

Why does this work?

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' \leftarrow \{0,1\}^{2\lambda} \\ r_1, r_2 = G(k') \end{array} \right\}$$

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' \leftarrow \{0,1\}^{2\lambda} \\ r_1, r_2 = G(k') \end{array} \right\}$$

$$\equiv$$

Refactoring

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0 \leftarrow \{0,1\}^{\lambda} \\ r_1, r_2 \leftarrow \left\{ G(k') \;\middle|\; k' \leftarrow \{0,1\}^{\lambda} \right\} \end{array} \right\}$$

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0 \leftarrow \{0,1\}^\lambda \\ r_1, r_2 \leftarrow \left\{ G(k') \;\middle|\; k' \leftarrow \{0,1\}^\lambda \right\} \end{array} \right\}$$

$$\approx \qquad \text{By PRG security of } G$$

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0 \leftarrow \{0,1\}^\lambda \\ r_1, r_2 \leftarrow \{0,1\}^{2\lambda} \end{array} \right\}$$

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0 \leftarrow \{0,1\}^{\lambda} \\ r_1, r_2 \leftarrow \{0,1\}^{2\lambda} \end{array} \right\}$$

$$\equiv$$

Refactoring

$$\left\{ r \;\middle|\; r \leftarrow \{0,1\}^{3\lambda} \right\}$$

$$\left\{ G'(k) \;\middle|\; k \leftarrow \{0,1\}^{\lambda} \right\}$$

$$\approx$$

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' \leftarrow \{0,1\}^{2\lambda} \\ r_1, r_2 = G(k') \end{array} \right\}$$

$$\approx$$

$$\left\{ r \;\middle|\; r \leftarrow \{0,1\}^{3\lambda} \right\}$$

This ensemble is not part of the security definition

It is a ***hybrid experiment***

$$\left\{ G'(k) \;\middle|\; k \leftarrow \{0,1\}^{\lambda} \right\}$$

$$\approx$$

$$\left\{ r_0, r_1, r_2 \;\middle|\; \begin{array}{l} r_0, k' \leftarrow \{0,1\}^{2\lambda} \\ r_1, r_2 = G(k') \end{array} \right\}$$

$X \approx Y$ and $Y \approx Z \implies X \approx Z$

$$\approx$$

$$\left\{ r \;\middle|\; r \leftarrow \{0,1\}^{3\lambda} \right\}$$

$$f : \{0,1\}^\lambda \to \{0,1\}^n$$

f is called a **one-way function** if for any PPT program A and for all inputs x the following probability is negligible:

$$\Pr\left[\ f(A(f(x)) = f(x)\ \middle|\ x \leftarrow \{0,1\}^\lambda\right]$$

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

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

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

Uniformly sampling k “emulates” a huge random table

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

Block length

F is called a **pseudorandom permutation (or block cipher)** if:

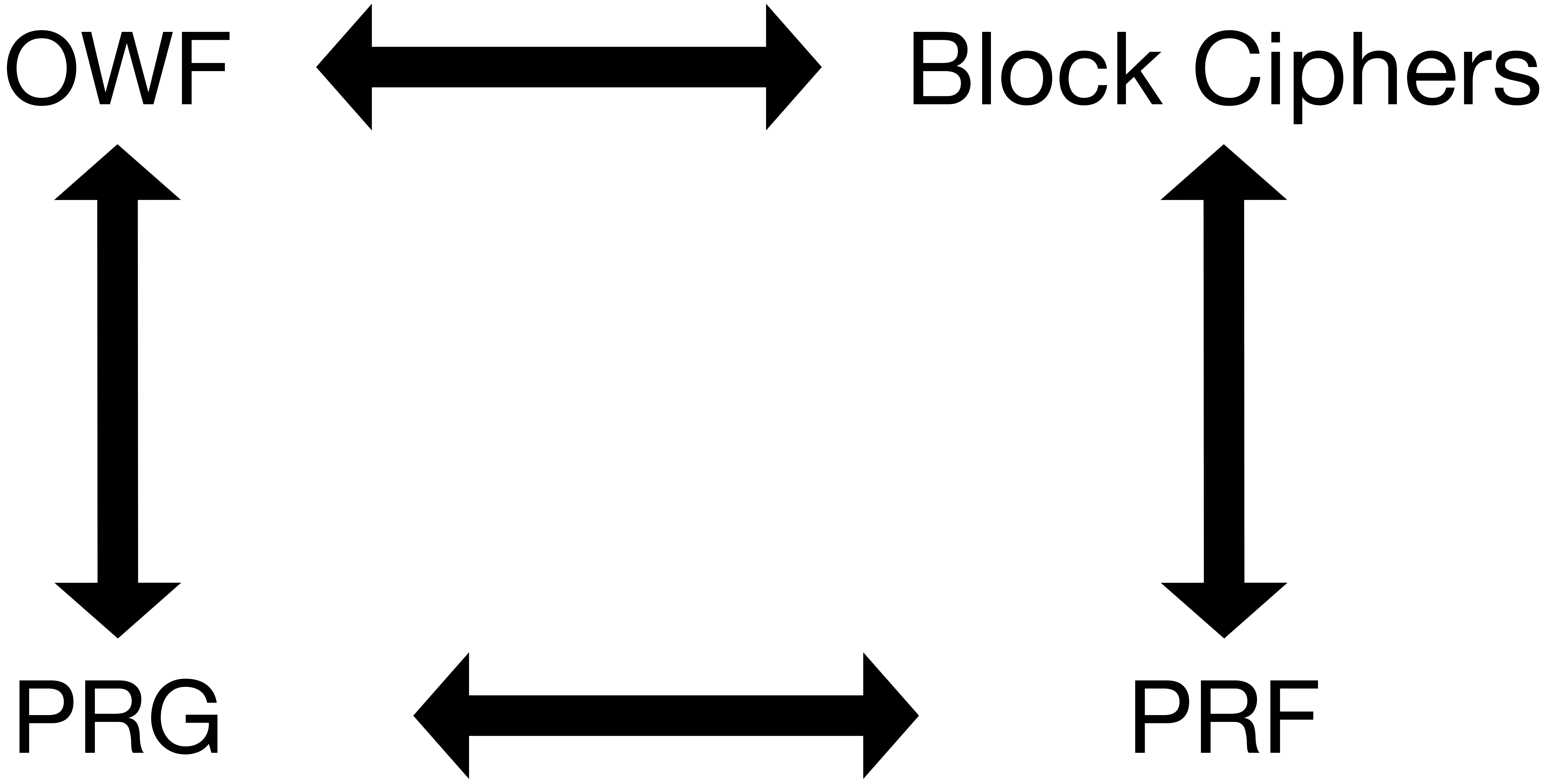
$$\left\{ F(k, \cdot\,) \;\middle|\; k \leftarrow \{0,1\}^\lambda \right\}$$

$$\approx$$

$$\left\{ f \;\middle|\; f \leftarrow \text{uniform permutation from } \{0,1\}^n \to \{0,1\}^n \right\}$$

And there exists efficient F^{-1} s.t. $F^{-1}(k, F(k,x)) = x$

AES is a block cipher



Any one of these implies all the others

OWFs exist $\implies P \neq NP$

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

```
k ← K

encrypt(m0, m1):
  if |m0| ≠ |m1|:
    return error
  ct ← Enc(k, m0)
  return ct
```

$\approx$

```
k ← K

encrypt(m0, m1):
  if |m0| ≠ |m1|:
    return error
  ct ← Enc(k, m1)
  return ct
```

Deterministic encryption cannot achieve CPA security — 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
```

Block Cipher Modes of Operation

Electronic Codebook (ECB) Mode —
WARNING: NOT RECOMMENDED!

Cipher Block Chaining (CBC) Mode — Very common in practice

Counter (CTR) Mode

Goal: Use a block cipher to **efficiently** encrypt a long message

Padding:

`pad(m)` : takes input message, outputs string whose length is multiple of block length

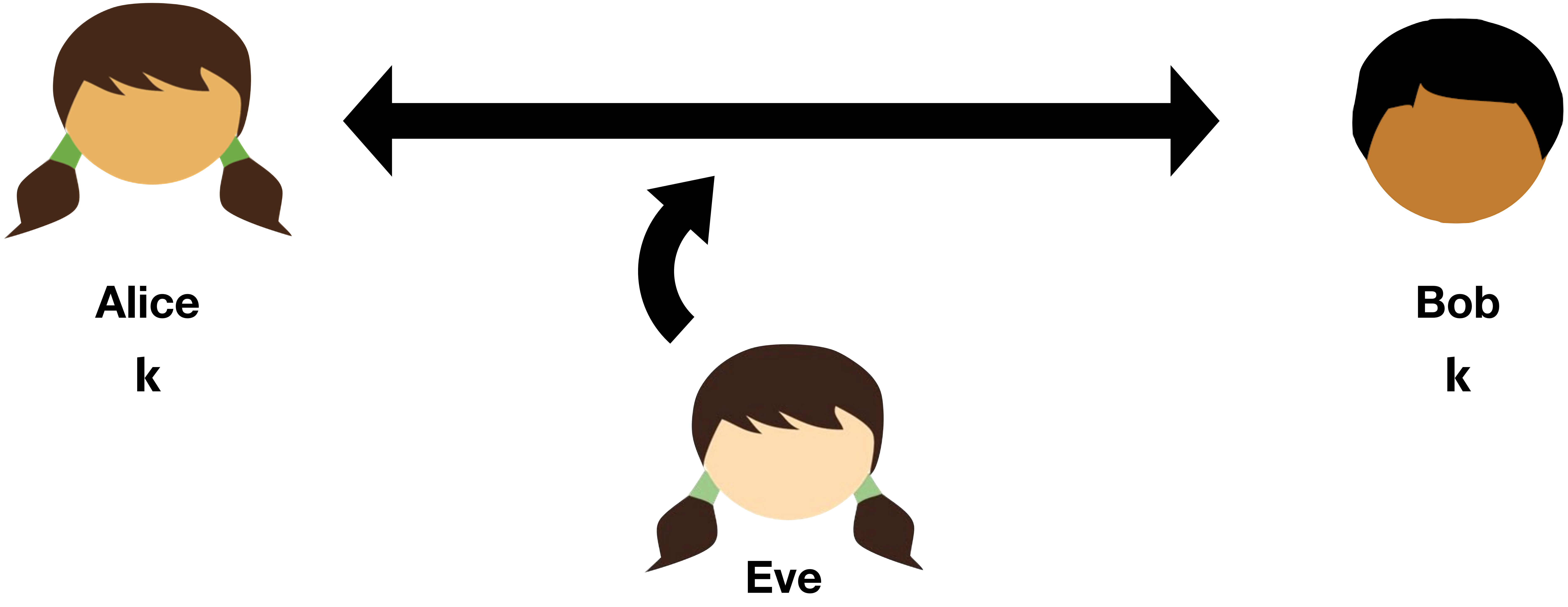
`unpad(m)` : inverse of pad

Correctness: `unpad(pad(m)) = m`

✓

Suggestion: Pad by a single 1, then pad with 0s until multiple of block length
To unpad, strip last 1 and all following 0s

Exercise: suppose that m is already a multiple of the block length.
Does Alice need to pad it?

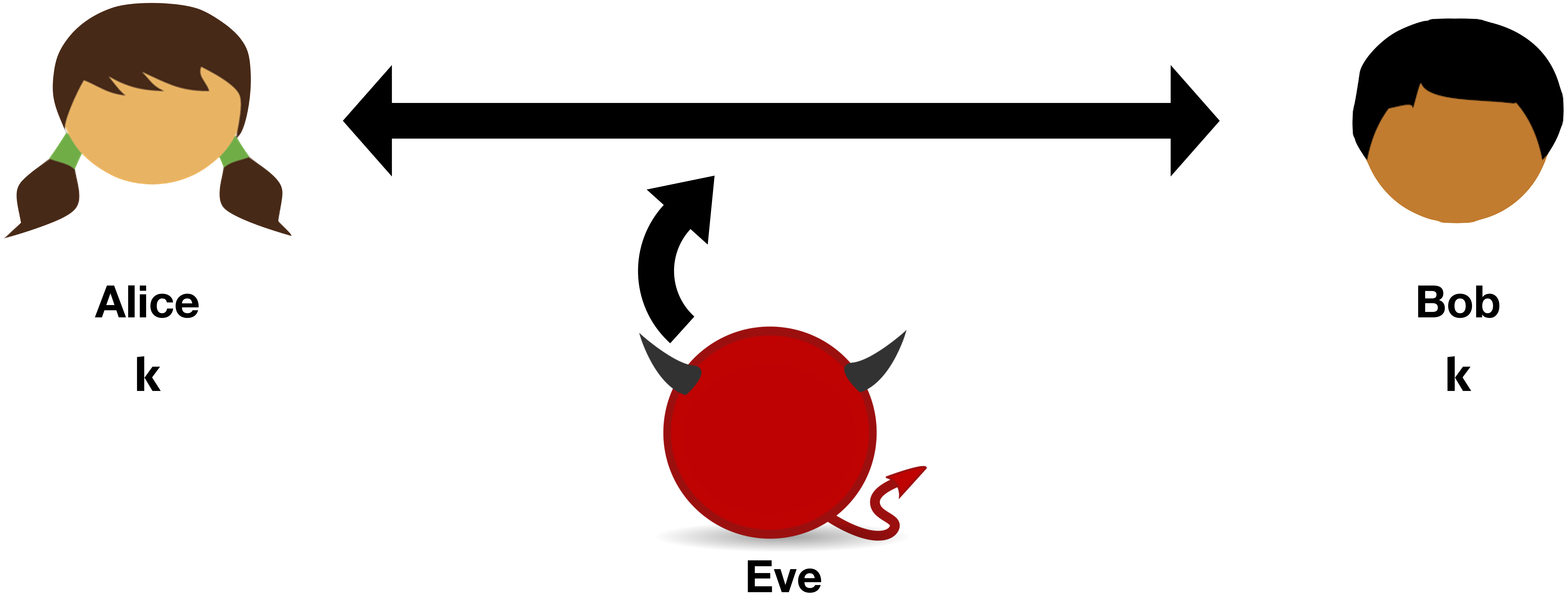


Confidentiality

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?



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):
  if |m0| ≠ |m1|:
    return error
  ct ← Enc(k, m0)
  return ct
```

$\approx$

```
k ← K

encrypt(m0, m1):
  if |m0| ≠ |m1|:
    return error
  ct ← Enc(k, m1)
  return ct
```

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)
```

$\approx$

```
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) 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)
```

How (informally) can we get CCA security?

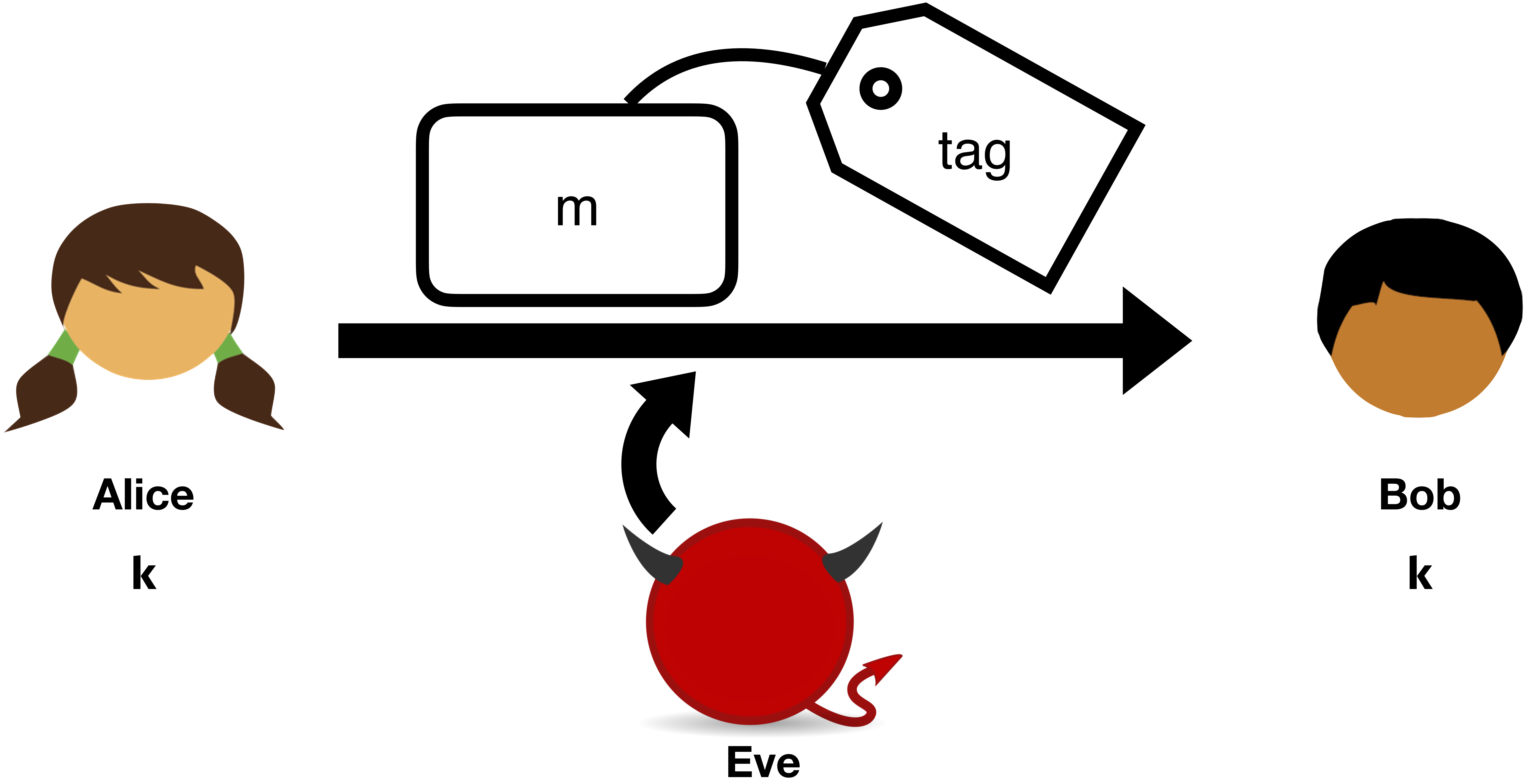
If adversary changes a ciphertext:

The decryption is “unrelated” to the original message

The decrypt procedure *detects* that when ciphertexts have been changed

If Enc/Dec are malleable, they will not achieve CCA security

Message Authentication Codes (MACs)



"Eve cannot change m without breaking the tag"

Message Authentication Codes

A **MAC scheme** with key space K is an algorithm `tag` such that:

```
k ← K                          Real

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

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

$\approx$

```
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
```

Message Authentication Codes

PRF $\Rightarrow$ MAC Scheme

Straightforward for messages
matching input length of PRF

CBC/ECBC MAC allow computing
MACs of longer messages

CBC-MAC

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

If F is a secure PRF, then CBC-MAC is a secure MAC for messages of length $n\lambda$

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

Encrypt-then-MAC

Given CPA encryption scheme E and MAC scheme M

K = E.K × M.K
M = E.K
C = E.C × M.T

```
Enc((ke, km), m):
  c ← E.Enc(ke, m)
  t ← M.tag(km, c)
  return (c, t)
```

```
KeyGen():
  ke ←$ {0,1}^λ
  km ←$ {0,1}^λ
  return (ke, km)
```

```
Dec((ke, km), c):
  if t ≠ M.tag(km, c)
    return error
  return E.Dec(ke, c)
```

Encrypt-then-MAC is CCA secure

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)
```

$\approx$

```
encrypt(m):
  c ←$ C
  return c

decrypt(c):
  return error
```

Encrypt-then-MAC is also authenticated

Alice Bob

Random Oracle Model

0) Define assumptions
1) Define Security
2) Specify a system where parties have black-box access to RO
3) Prove that if assumptions hold, system is secure

---

4) *Replace all instances of RO by concrete function H, and pray*

Random Oracle Heuristic

Random Oracle

$$\left\{ f \;\middle|\; f \leftarrow \text{uniform function from } \{0,1\}^* \to \{0,1\}^{\lambda} \right\}$$

Consider a family of hash functions

$$H : \{0,1\}^\lambda \times \{0,1\}^* \rightarrow \{0,1\}^\lambda$$

A hash family H is **collision resistant** if no poly-time adversary can produce a hash collision

Namely, for any poly-time adversary A, the following probability is negligible

$$\Pr\left[H(s, x_0) = H(s, x_1) \;\middle|\; \begin{array}{l} s \leftarrow \{0,1\}^\lambda \\ (x_0, x_1) \leftarrow A(s) \end{array} \right] < \text{negl}(\lambda)$$