Encrypting Long and Variable-Length Messages

Block Cipher Modes of Operation

CS/ECE 407

Today's objectives

Discuss **Block Cipher Modes of Operation**

See how to encrypt long messages

Explain problem of **variable length messages**

Show how to **pad** messages to achieve CPA security

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

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

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

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

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

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

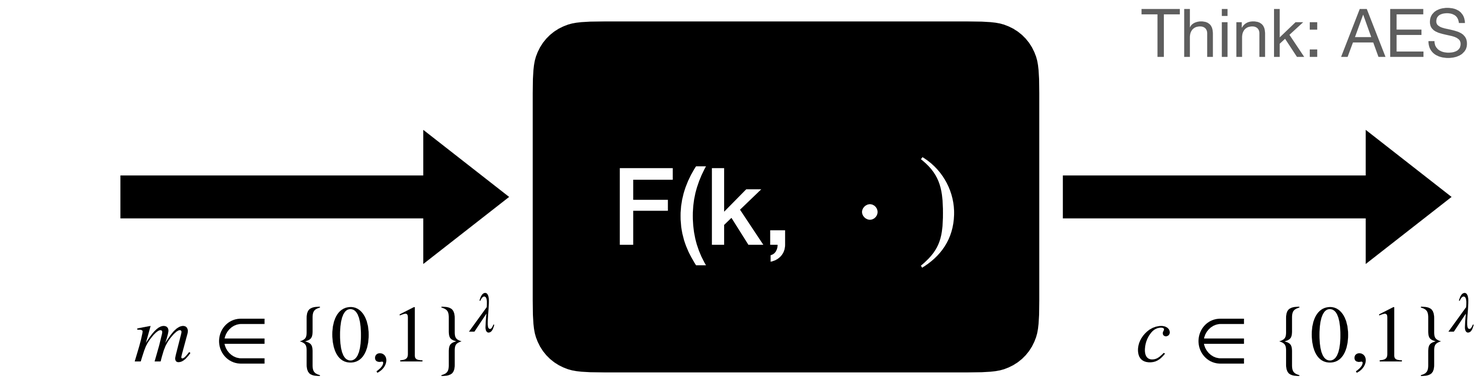
$$\approx$$

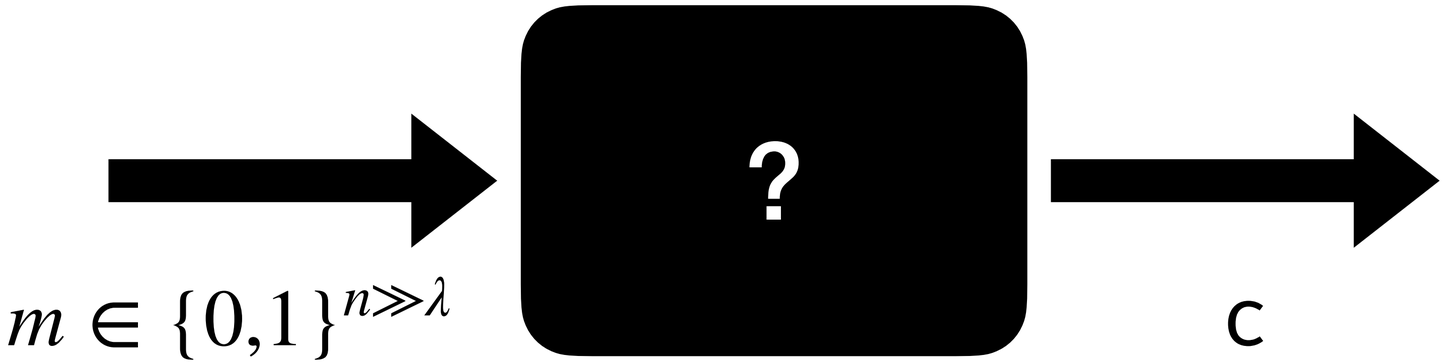
$$\left\{ f \;\middle|\; f \leftarrow \text{uniform permutation from } \{0,1\}^n \to \{0,1\}^n \right\}$$

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

Block Cipher Modes of Operation

Block Cipher Modes of Operation





Block Cipher Modes of Operation

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

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

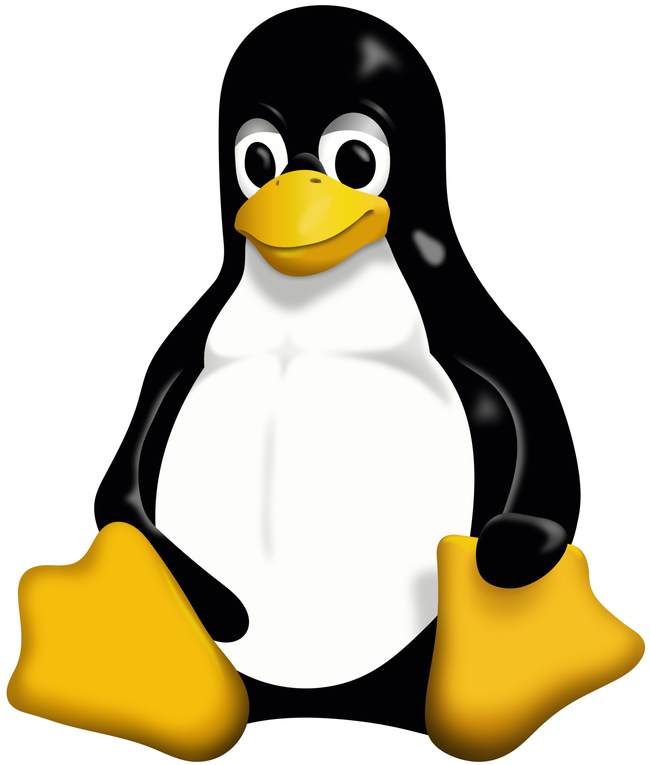
Counter (CTR) Mode

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

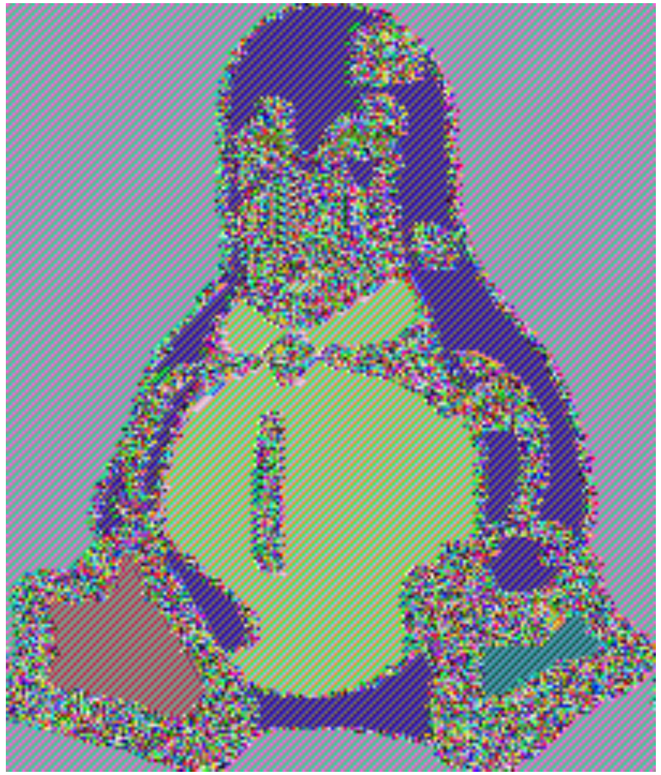
```
Enc(k, m_1 | … | m_n):
  for i in 1 to n
    c_i ← F(k, m_i)
  return c_1 | … | c_n

Dec(k, c_1 | … | c_n):
  for i in 1 to n
    m_i ← F^{-1}(k, c_i)
  return m_1 | … | m_n
```

ECB Mode: Do not use!!!



“Good” encryption



ECB Mode

Cipher Block Chaining (CBC) Mode

```
Enc(k, m_1 | … | m_n):
  c_0 ←$ {0,1}^λ
  for i in 1 to n
    c_i ← F(k, m_i ⊕ c_i-1)
  return c_0 | c_1 | … | c_n

Dec(k, c_0 | c_1 | … | c_n):
  for i in 1 to n
    m_i ← F^-1(k, c_i) ⊕ c_i-1
  return m_1 | … | m_n
```

“initialization vector”

Counter (CTR) Mode

```
Enc(k, m_1 | … | m_n):
  r ←$ {0,1}^λ
  for i in 1 to n
    c_i ← F(k, r + i) ⊕ m_i
  return r | c_1 | … | c_n

Dec(k, r | c_1 | … | c_n):
  for i in 1 to n
    m_i ← F(k, r + i) ⊕ c_i
return m_1 | … | m_n
```

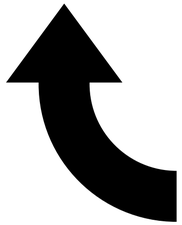
“initialization vector”

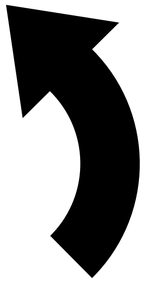
Block Cipher Modes of Operation

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

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

Counter (CTR) Mode — Allows parallelism





CPA secure for messages that are multiples of block length

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

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

What if message length is not a multiple of the block length?

Padding:

Consider:

$$\texttt{ct} \leftarrow \texttt{Enc}(\texttt{k}, 0^{\lambda-1})$$

How should we handle this?

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`

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` appends 0s until m is multiple of block length

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` appends 0s until m is multiple of block length

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

Padding:

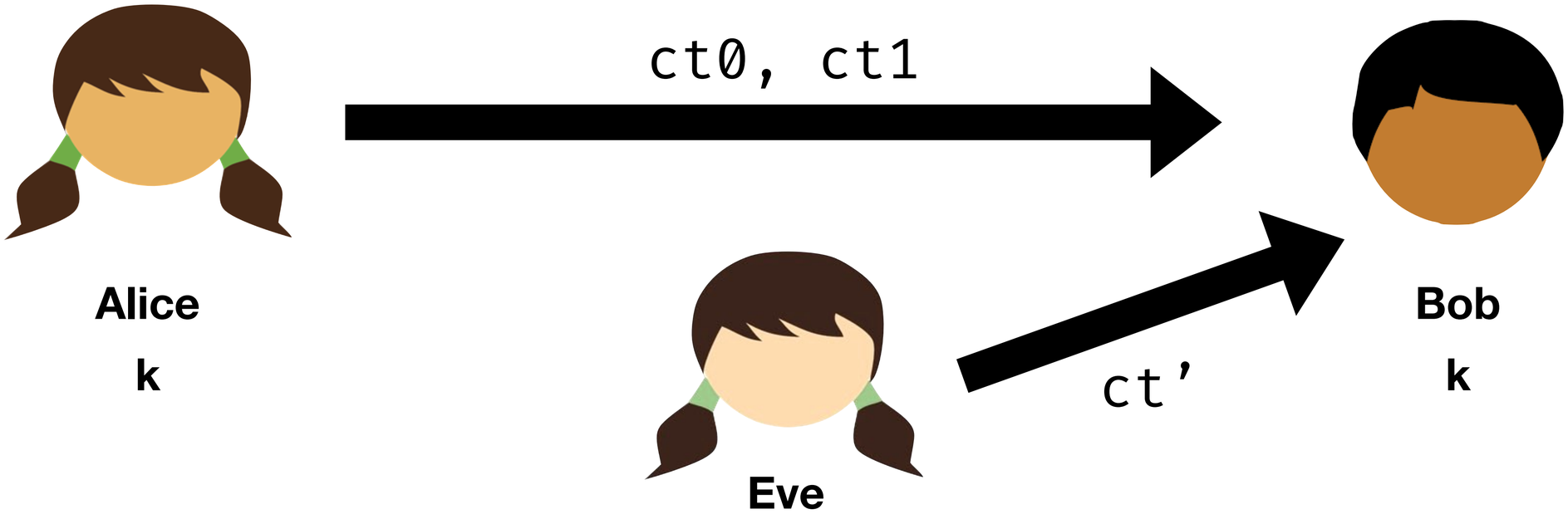
`pad(m)` : takes input message, outputs string whose length is multiple of block length

`unpad(m)` : inverse of pad

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

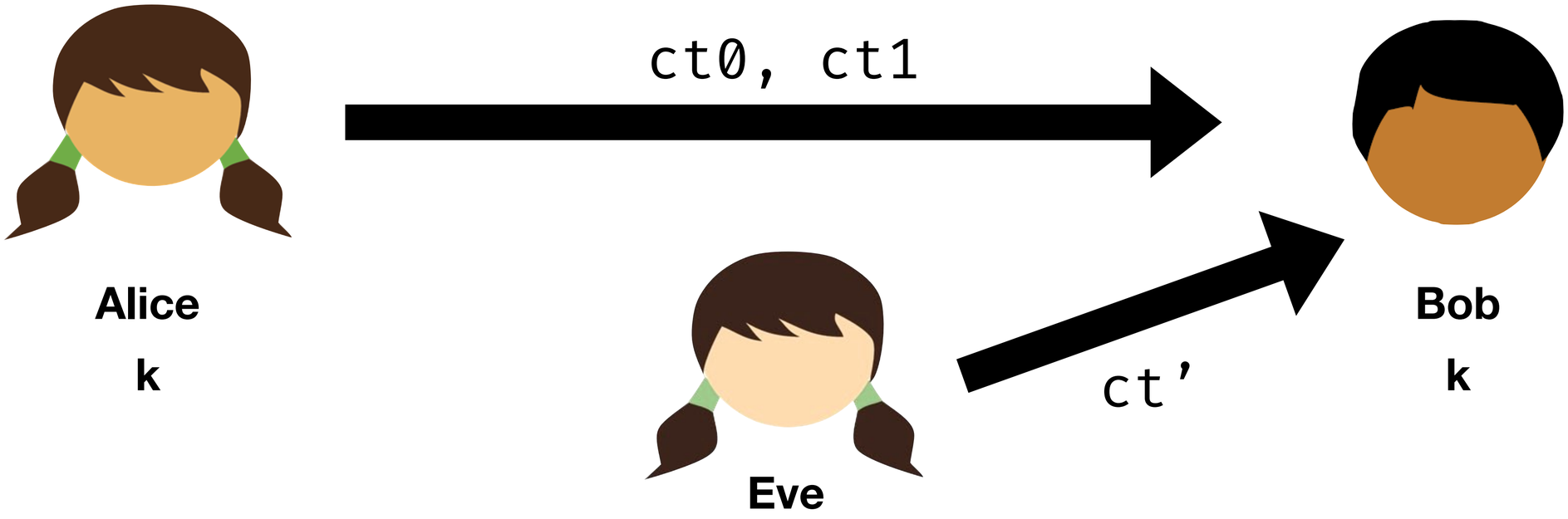
Suggestion: Pad by a single 1, then pad with 0s until multiple of block length
To unpad, strip last 1 and all following 0s

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



Alice and Bob can now exchange arbitrary numbers of arbitrary-length messages with confidentiality

However, we have no notion of **authenticity**



Alice and Bob can now exchange arbitrary numbers of arbitrary-length messages with confidentiality

However, we have no notion of **authenticity**

So far our definition of security provides no way for Bob to check that a ciphertext is a “good one"

Today's objectives

Discuss **Block Cipher Modes of Operation**

See how to encrypt long messages

Explain problem of **variable length messages**

Show how to **pad** messages to achieve CPA security