Lecture 6: Symmetric Cryptography

CS 181S

Fall 2020

The Big Picture Thus Far…

Attacks
are perpetrated by
threats
that inflict
harm
by exploiting
vulnerabilities
which are controlled by
countermeasures.

Dolev-Yao Threat Model (1983)

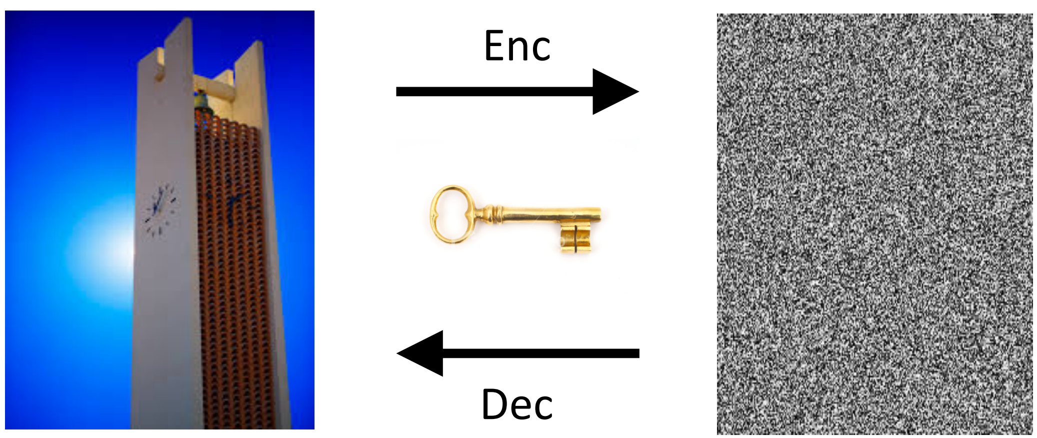
- Assume an attacker with network access and the following capabilities:
  - Can read all messages on the network
  - Can write messages to the network
  - Can block any messages sent over the network (i.e., cause them to be dropped)

Purpose of encryption

- **Threat:** Dolev-Yao attacker
- **Vulnerability:** communication channel between sender and receiver can be read by other principals
- **Harm:** messages containing secret information disclosed to attacker (violating confidentiality)
- **Countermeasure:** **encryption**

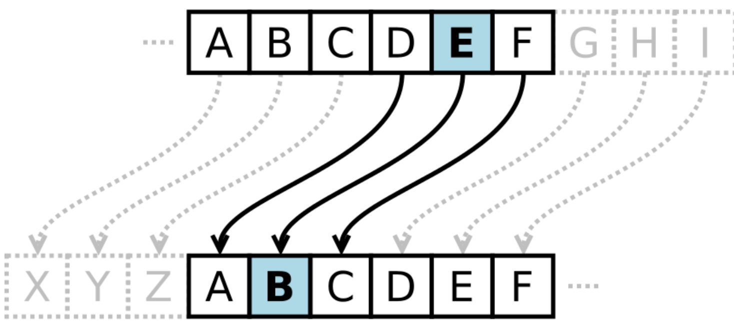
(Symmetric) Encryption algorithms

- $\mathrm{Gen}(1^n)$: generate a **key** of length n
- $\mathrm{Enc}(m; k)$: encrypt **message** under key k
- $\mathrm{Dec}(m; k)$: decrypt **ciphertext** c with key k



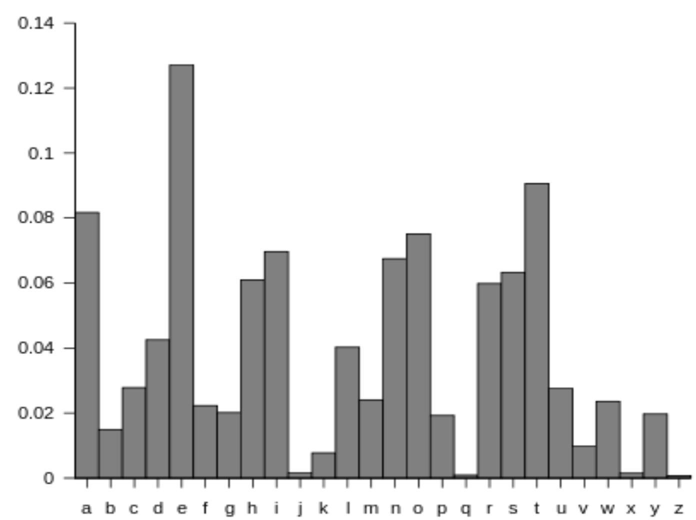
$(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ is a symmetric-key **encryption scheme** aka **cryptosystem**

Classical Crypto: Substitution Ciphers



WKLV LV QRW VR VHFXUH

THIS IS NOT SO SECURE



Classical Crypto: Vigenere Cipher

```
THIS IS NOT SO SECURE
KEYK EY KEY KE YKEYKE

EMHD NR YTS DT RPHTCJ
```

Defining Security

- A crypto system is **secure** if

$$\forall \text{PPT } A, \exists \delta \in O(\frac{1}{2^n})\ s.t\ \forall n, \forall m, m' s.t. |m| = |m'| = n,$$
$$\Pr\left[A\left(Enc(m;k)\right) = m\right] \leq \Pr\left[A\left(Enc(m';k)\right) = m\right] + \delta(n)$$

One-Time Pad

- $\text{Gen}(1^n) :=$ generate a random bitstring of length n
- $\text{Enc}(m; k) := m \oplus k$
- $\text{Dec}(c; k) := c \oplus k$

plaintext	THIS IS SECURE
plaintext	0101010001001000010010010101010011 …
key	0110101010010101010010100000101110 …
ciphertext	0011111011011101000000011010001 01 …

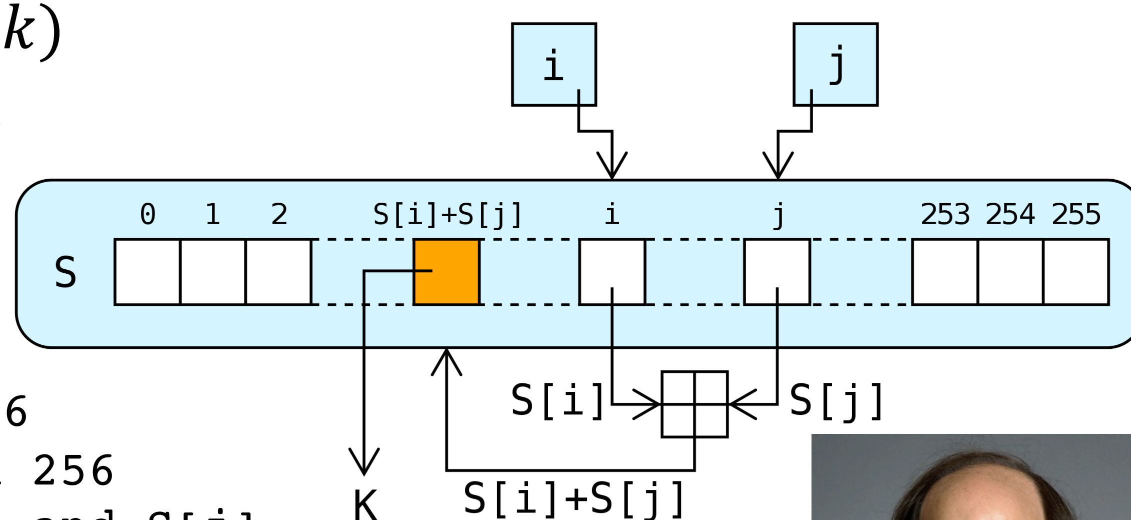
- $\forall m, m' \text{ s.t. } |m| = |m'|,\ \Pr[m \mid c] = \left(\frac{1}{2}\right)^{\text{len}(m)} = \Pr[m' | = c]$

Exercise 1: One-time Pads

- Explain why one-time pads are no longer used in practice

Stream Ciphers: RC4

- $\text{Gen}(1^n) :=$ generate a random bitstring of length n $\approx \mathbf{128}$
  use that to initialize permutation S of the 256 possible bytes
- $\text{Enc}(m; k) := m \oplus r(k)$
- $\text{Dec}(c; k) := c \oplus r(k)$

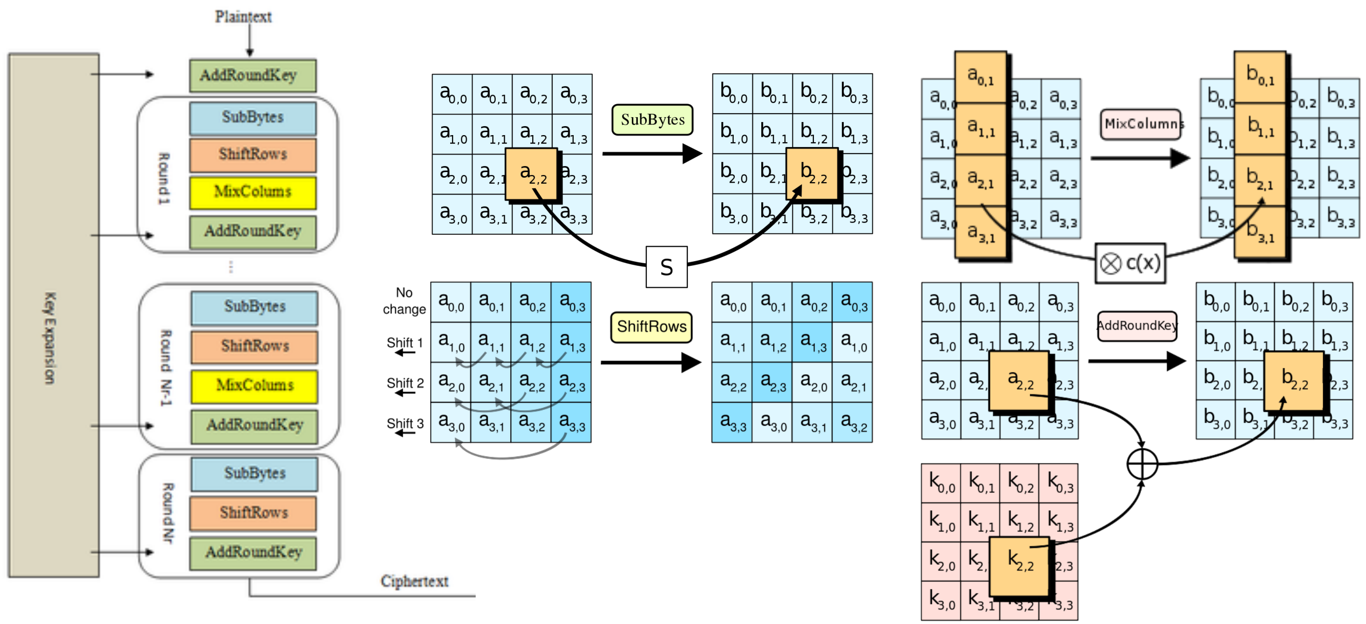



```
i := 0
j := 0
while True:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap values of S[i] and S[j]
    r := S[(S[i] + S[j]) mod 256]
output r
```

- Alternatives: ChaCha20, Speck

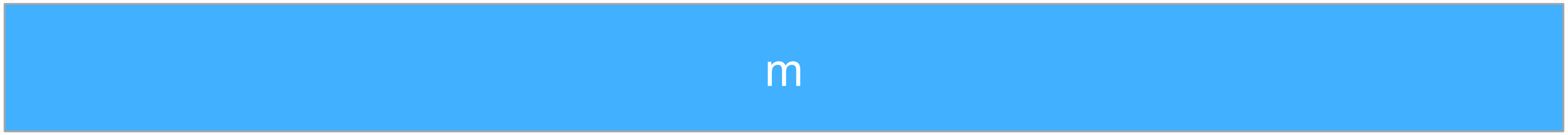
Block Ciphers: AES

- Encryption schemes that operate on fixed-size messages called **blocks**
- Advanced Encryption Standard (AES) result of 2001 NIST competition
- Currently no known practical attacks, approved by NSA for top-secret
- $\mathrm{Gen}(1^n) :=$ generate a random bitstring of length n $= \mathbf{128, 196, 256}$



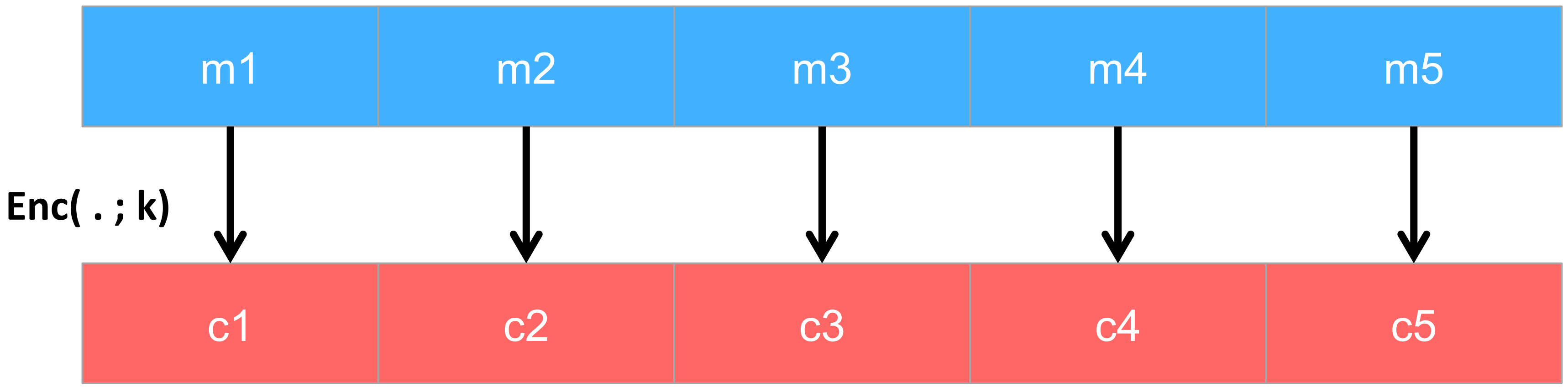
The obvious idea...

- Divide long message into short chunks, each the size of a block
- Encrypt each block with the block cipher



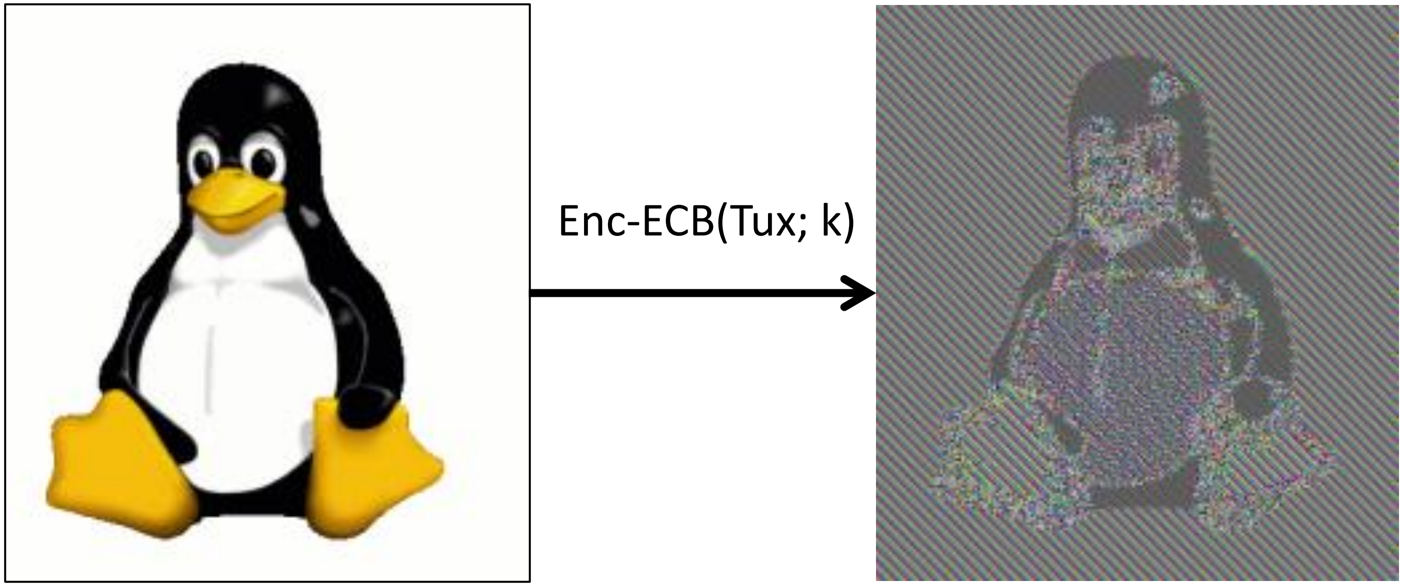
The obvious idea...

- Divide long message into short chunks, each the size of a block
- Encrypt each block with the block cipher



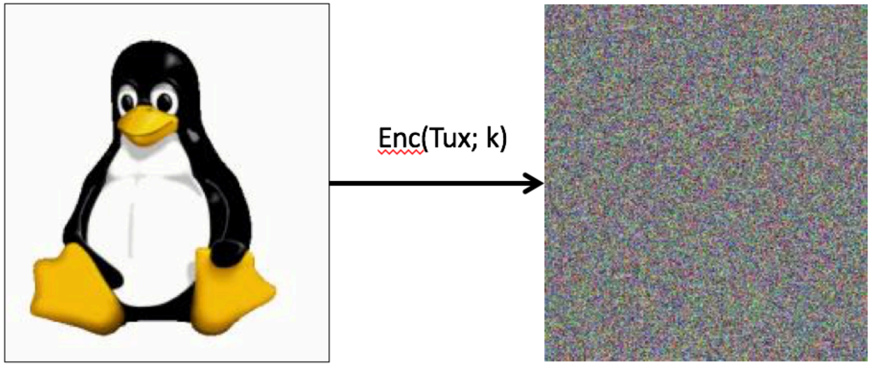
- Called *electronic code book* (ECB) mode

...is a bad idea



Good modes

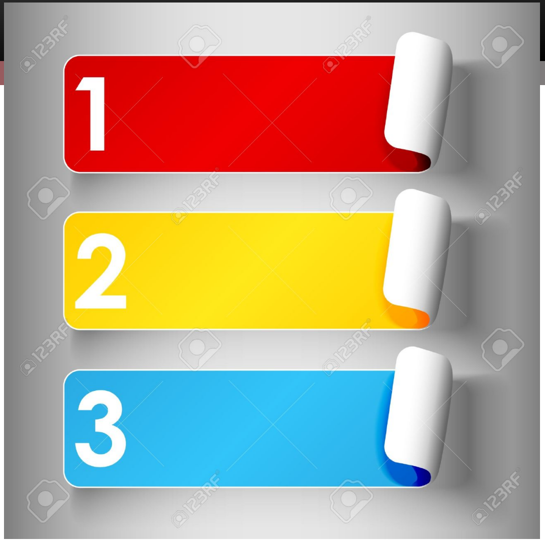
- Cipher Block Chaining (CBC) mode
  - idea: XOR previous ciphertext block into current plaintext block
- Counter (CTR) mode
  - idea: derive one-time pad from increasing counter
- With both:
  - every ciphertext block depends in some way upon previous plaintext or ciphertext blocks
  - so even if plaintext blocks repeat, ciphertext blocks don't
  - so *intra-message* repetition doesn't disclose information



Good modes

- Problem: block ciphers are *deterministic*: inter-message repetition is visible to attacker
- Both CBC and CTR modes require an additional parameter: a *nonce*
  - Enc(m; nonce; k)
  - Dec(c; nonce; k)
  - CBC calls the nonce an *initialization vector* (IV)
- Different nonces make each encryption different than others
  - Hence inter-message repetition doesn't disclose information

Nonces



A nonce is a number used once

Must be

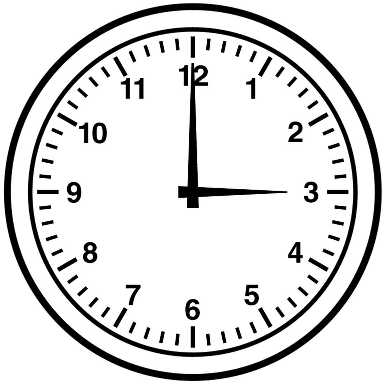
- **unique:** never used before in lifetime of system

and/or (depending on intended usage)

- **unpredictable:** attacker can't guess next nonce given all previous nonces in lifetime of system

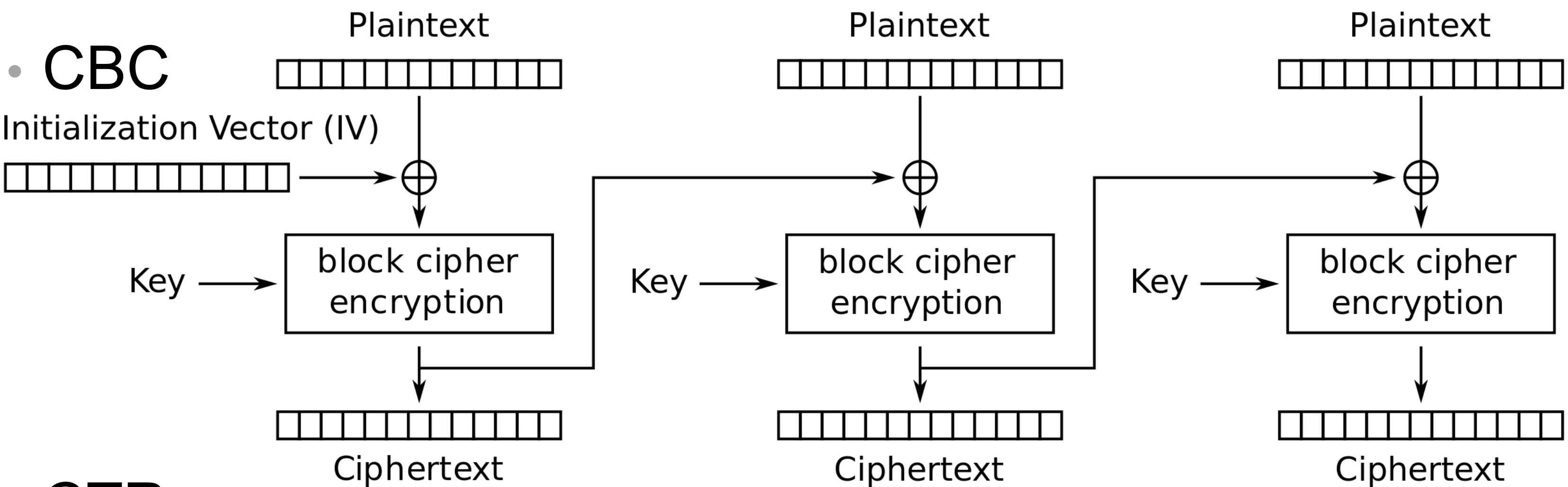
Nonce sources

- **counter**
  - requires state
  - easy to implement
  - can overflow
  - highly predictable
- **clock:** just a counter
- **random number generator**
  - might not be unique, unless drawn from large space
  - might or might not be unpredictable
  - generating randomness:
    - standard library generators often are not cryptographically strong, i.e., unpredictable by attackers
    - cryptographically strong randomness is a black art

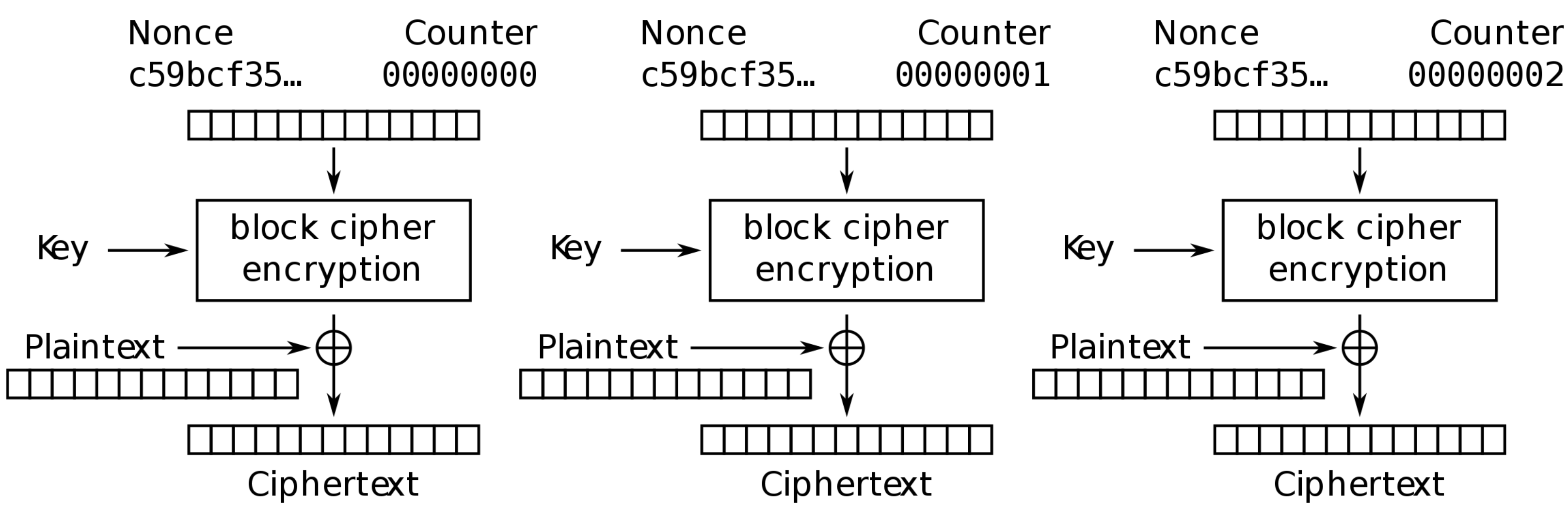


Good Block Modes

- CBC

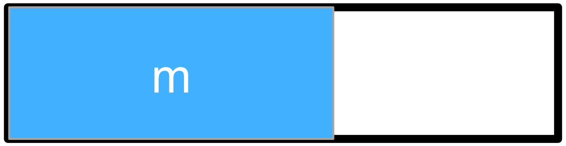


- CTR



Padding

What if the message length isn't *exactly* a multiple of block length? End up with final block that isn't full:



Non-solution: pad out final block with 0's (not reversible)

Solution: Let B be the number of bytes that need to be added to final plaintext block to reach block length. Pad with B copies of the byte representing B. Called *PKCS #5 or #7 padding.*

Exercise 2: Block Modes

- Which of the good block modes we talked about (CBC and CTR) would parallelize efficiently?

Exercise 3: Feedback

1. Rate how well you think this recorded lecture worked
   1. Better than an in-person class
   2. About as well as an in-person class
   3. Less well than an in-person class, but you still learned something
   4. Total waste of time, you didn't learn anything

2. How much time did you spend on this video lecture (including time spent on exercises)?

3. Do you have any comments or feedback?