



WEEK 3

ENCRYPTING WITH BLOCK CIPHERS

SE 4472 - Information Security



Western
Engineering



WHISPER
LAB

BLOCK CIPHERS

- Encryption of fixed length chunks of bits (called “blocks”)
- Each block is encrypted separately



BLOCK CIPHERS



KEYGEN

Accepts a security parameter k , outputs a random k -bit key

$$\text{Gen} : k \rightarrow \{0, 1\}^k$$



ENCRYPT

Accepts a b -bit plaintext and k -bit key and outputs a b -bit ciphertext

$$\text{Enc} : \{0, 1\}^b \times \{0, 1\}^k \rightarrow \{0, 1\}^b$$



DECRYPT

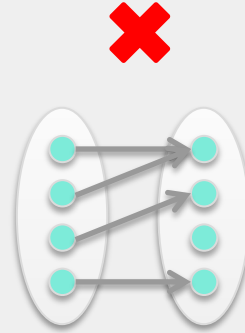
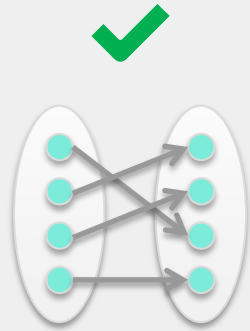
Accepts a b -bit ciphertext and k -bit key and outputs a b -bit plaintext

$$\text{Dec} : \{0, 1\}^b \times \{0, 1\}^k \rightarrow \{0, 1\}^b$$

AN IDEAL BLOCK CIPHER



AN IDEAL BLOCK CIPHER



Encryption is **injective**
Every plaintext maps to a unique ciphertext

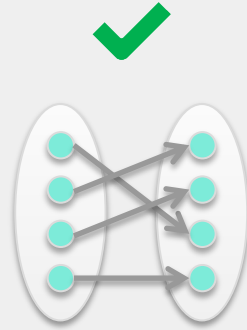
AN IDEAL BLOCK CIPHER



Encryption is **surjective**

Every ciphertext is associated with some plaintext

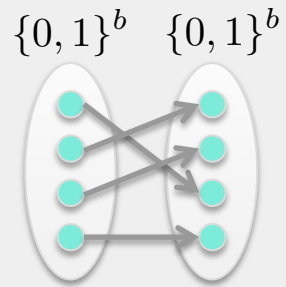
AN IDEAL BLOCK CIPHER



Injective + Surjective = Bijective

Every plaintext is associated with a unique ciphertext, and vice versa

AN IDEAL BLOCK CIPHER



Encryption is a permutation
(shuffle) of b -bit strings

AN IDEAL BLOCK CIPHER



PLAINTEXT SPACE

There are 2^b plaintexts



CIPHERTEXT SPACE

There are 2^b ciphertexts



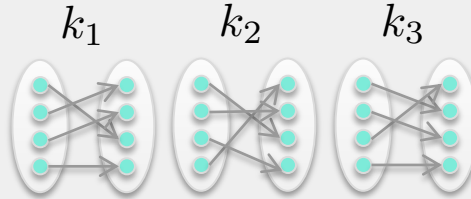
PERMUTATION SPACE

There are $2^b!$ permutations of 2^b elements



KEY SPACE

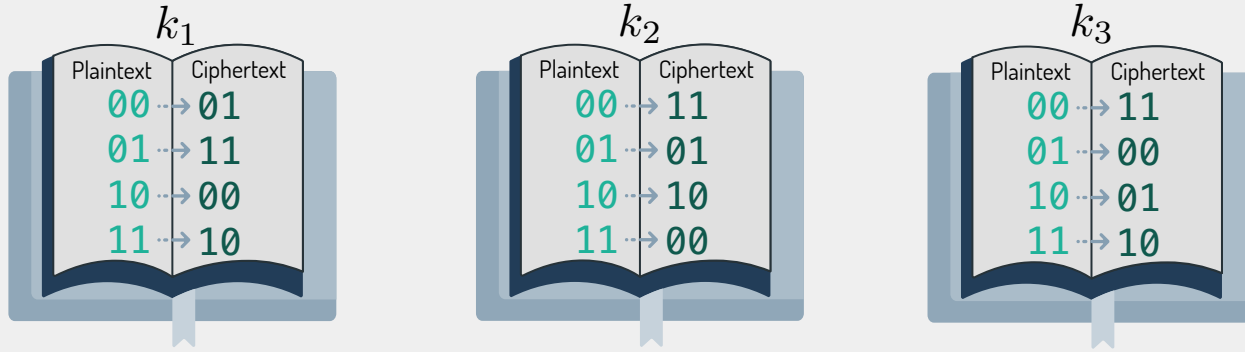
There are 2^k keys



PERMUTATION SELECTION

Each key “chooses” some permutation from the permutation space

AN ELECTRONIC CODEBOOK



So a block cipher is like a collection of codebooks
Which codebook did you pick? Shh, it's a secret

ENCRYPTING LARGE PLAINTEXTS

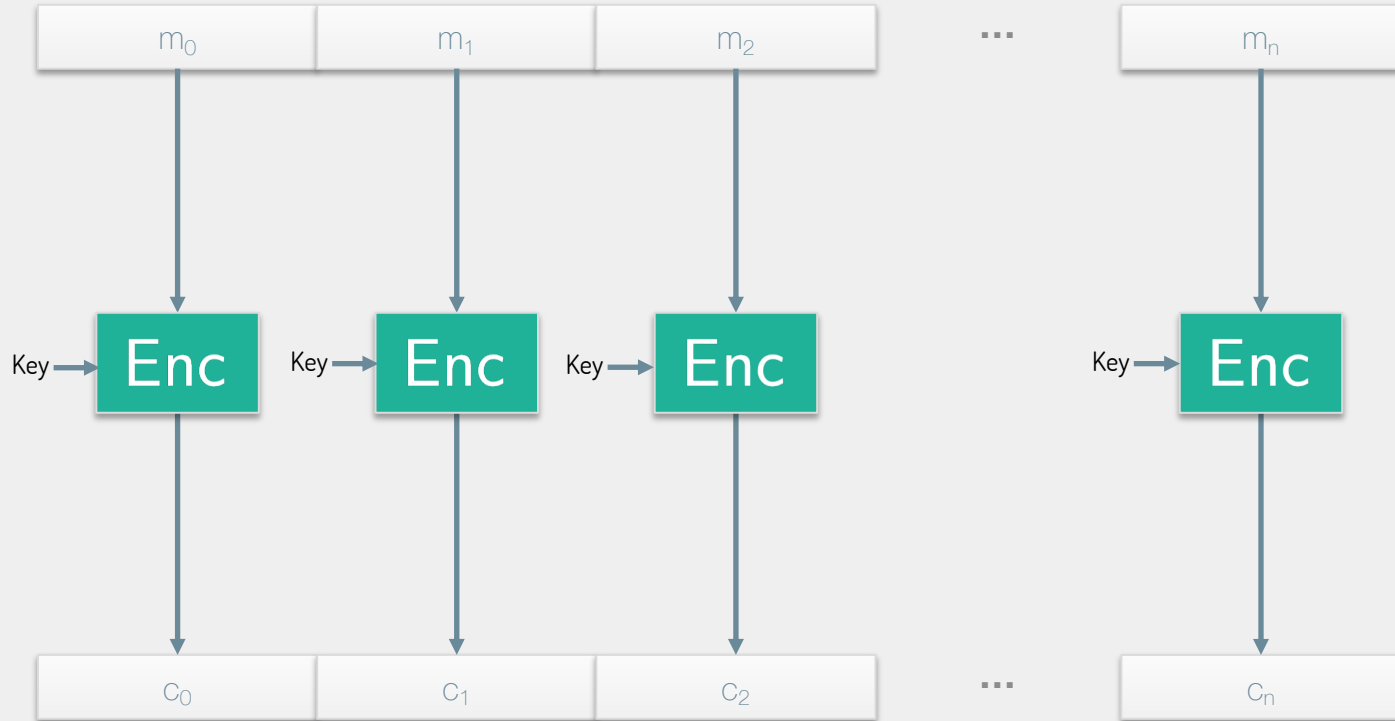


IF BLOCK CIPHERS ACT ON SHORT
BLOCKS, HOW DO WE ENCRYPT A
LONG MESSAGE?

CIPHER MODES OF OPERATION

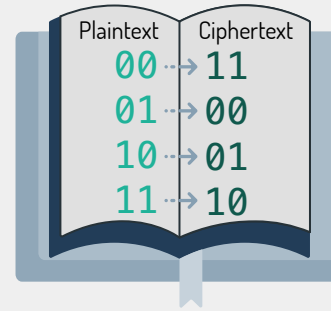


ELECTRONIC CODEBOOK MODE (ECB)



ELECTRONIC CODEBOOK MODE (ECB)

0	0	1	1	0	0	0	1	0	0	0	1	0	1	0	1	1	0	1	0	1	0	0	0	
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	
1	1	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1



APPROACH

ENCRYPT EACH BLOCK SEPARATELY

ELECTRONIC CODEBOOK MODE (ECB)

ECB with a 128-bit block size (for a given key)

The codebook has 2^{128} entries

PLAINTEXT

CIPHERTEXT

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	67 8E EE 31 4C FC 24 2E F7 62 9E 39 D3 08 A1 4E
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01	51 03 35 71 62 68 87 E3 CB 16 86 BD EF C6 13 C3
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02	3B 1D F9 59 A8 31 C6 70 AF A8 F2 4A B9 B8 27 4B
⋮	⋮
⋮	⋮
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FD	F1 34 B9 FD 4B 01 43 33 C7 20 C2 17 29 A5 CB 11
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FE	D1 2E 8D B2 4F F4 F3 77 E8 9F 92 31 A3 38 48 B0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	B0 5B 46 3E 58 2A AE FC 5A 40 58 E9 B4 5F 34 B3

ELECTRONIC CODEBOOK MODE (ECB)



Wait a second

Doesn't that mean this:

FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

always encrypts to this?:

B0 5B 46 3E 58 2A AE FC 5A 40 58 E9 B4 5F 34 B3

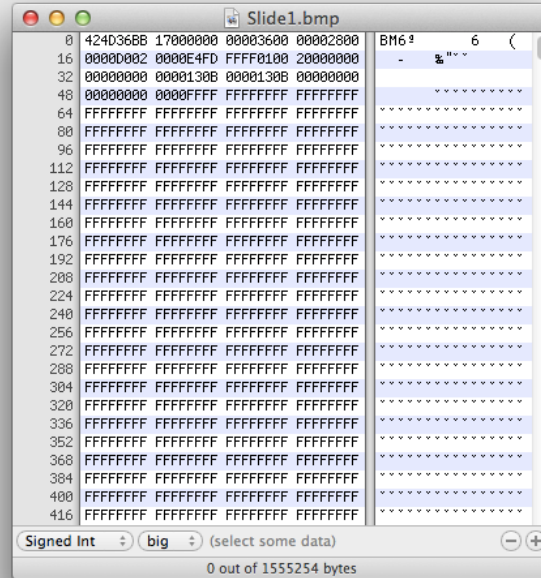
ELECTRONIC CODEBOOK MODE (ECB)

Slide1.bmp

SE 4472: Electronic
Codebook Mode

What could
possibly go wrong?

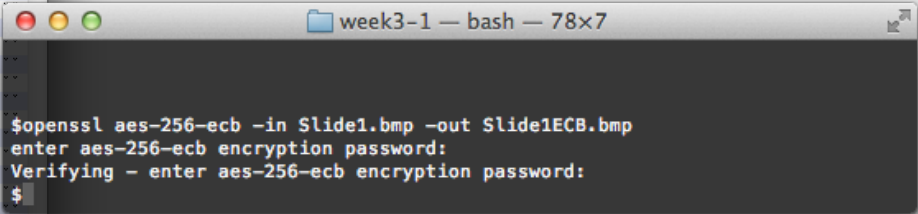
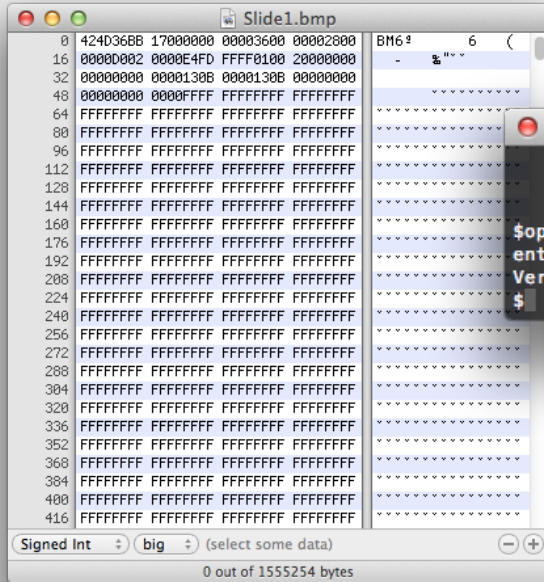
hexdump



```
Slide1.bmp
0  424D36BB 17000000 00003600 00002800  BM6  6
16 00000002 0000E4FD FFFF0100 20000000  -
32 00000000 0000130B 0000130B 00000000
48 00000000 0000FFFF FFFFFFFF FFFFFFFF
64 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
80 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
96 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
112 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
128 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
144 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
160 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
176 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
192 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
208 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
224 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
240 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
256 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
272 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
288 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
304 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
320 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
336 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
352 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
368 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
384 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
400 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
416 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF

Signed Int  big  (select some data)
0 out of 1555254 bytes
```

ELECTRONIC CODEBOOK MODE (ECB)



ELECTRONIC CODEBOOK MODE (ECB)

```
0 424D366B 17000000 00003600 00002800 BM6 6 (
16 00000002 0000E4FD FFFF0100 20000000 -  " " " "
32 00000000 0000130B 0000130B 00000000
48 00000000 0000FFFF FFFFFFFF FFFFFFFF
64 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
80 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
96 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
112 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
128 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
144 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
160 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
176 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
192 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
208 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
224 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
240 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
256 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
272 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
288 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
304 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
320 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
336 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
352 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
368 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
384 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
400 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
416 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
```

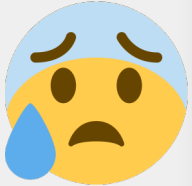
Signed Int big (select some data) - +
0 out of 1555254 bytes

```
0 B5C649C5 6DE08755 D7BE3D9E 581D7B66 μΔI≈mİğUφe=ÜX {f
16 12428419 BA3929D6 EA0C1CFC 1B21EDAB BŃ j9)=f , !ı'
32 0ECCD6FEE 51CEDC20 9FA91912 35FCA7A3 ōoŌQe< ūō 5,βε
48 B289E63D 1ACF2C3A EA4C15D6 D37736D8 ≤đē= e,:İL +*m6ē
64 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
80 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
96 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
112 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
128 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
144 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
160 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
176 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
192 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
208 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
224 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
240 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
256 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
272 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
288 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
304 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
320 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
336 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
352 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
368 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
384 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
400 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
416 B05B463E 582AAEFC 5A4058E9 B45F34B3 [F>X**E,Z0XÈ¥_4≥
```

Signed Int big (select some data) - +
0 out of 1555264 bytes

BLOCKS OF WHITE PIXELS ENCRYPT TO RANDOM-LOOKING COLOURS

ELECTRONIC CODEBOOK MODE (ECB)



```
Slide1.bmp
0 424D36EB 17000000 00003600 00002800 BM6: 6 (
16 00000002 0000E4FD FFFF0100 20000000 -  " " " "
32 00000000 0000130B 0000130B 00000000
48 00000000 0000FFFF FFFFFFFF FFFFFFFF
64 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
80 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
96 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
112 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
128 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
144 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
160 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
176 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
192 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
208 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
224 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
240 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
256 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
272 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
288 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
304 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
320 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
336 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
352 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
368 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
384 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
400 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
416 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
```

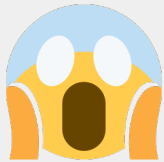
Signed Int big (select some data) 0 out of 1555254 bytes

```
SlideEnc.bmp
0 B5C649C5 6DE08755 D7BE3D9E 581D7B66 μΔI≈mİgUϕe=ÜX {f
16 12428419 BA3929D6 EA0C1CFC 1B21EDAB BŃ j9)=f !İ'
32 0ECCD6FEE 51CEDC20 9FA91912 35FCA7A3 0o0Qe< ü0 5,βE
48 B289E63D 1ACF2C3A EA4C15D6 D37736D8 ≤δē= e,:İL +*M6ē
64 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
80 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
96 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
112 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
128 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
144 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
160 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
176 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
192 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
208 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
224 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
240 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
256 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
272 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
288 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
304 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
320 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
336 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
352 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
368 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
384 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
400 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
416 B05B463E 582AAEFC 5A4058E9 B45F34B3 **[F>*X*,Z0XÈ¥_4≥
```

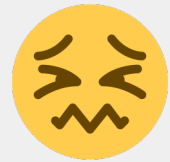
Signed Int big (select some data) 0 out of 1555264 bytes

PROBLEM: THEY ALWAYS ENCRYPT TO THE SAME COLOURS

ELECTRONIC CODEBOOK MODE (ECB)



SE 4472: Electronic
Codebook Mode



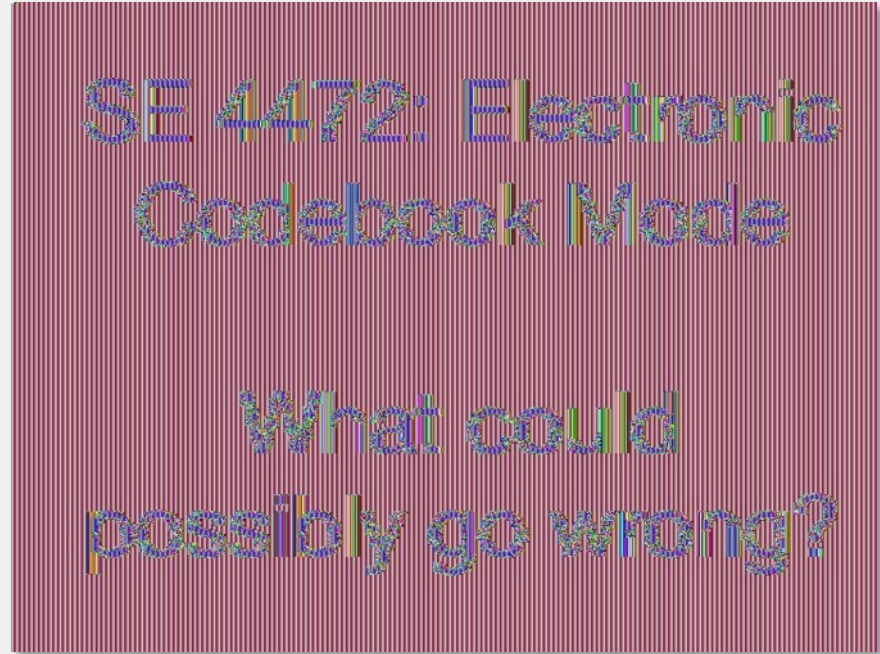
What could
possibly go wrong?

ELECTRONIC CODEBOOK MODE (ECB)

Is this cipher mode
indistinguishable under



eavesdropping?
(IND-EAV)



Move Fast and Roll Your Own Crypto A Quick Look at the Confidentiality of Zoom Meetings

By Bill Marczak and John Scott-Railton April 3, 2020

Read our description of Zoom's [waiting room vulnerability](#), as well as [frequently asked question](#) about Zoom and encryption issues.

This report examines the encryption that protects meetings in the popular Zoom teleconference app. We find that Zoom has “rolled their own” encryption scheme, which has significant weaknesses. In addition, we identify potential areas of concern in Zoom’s infrastructure, including observing the transmission of meeting encryption keys through China.

Key Findings

- Zoom [documentation](#) claims that the app uses “AES-256” encryption for meetings where possible. However, we find that in each Zoom meeting, a single AES-128 key is used in ECB mode by all participants to encrypt and decrypt audio and video. The use of ECB mode is not recommended because patterns present in the plaintext are preserved during encryption.

ELECTRONIC CODEBOOK MODE (ECB)



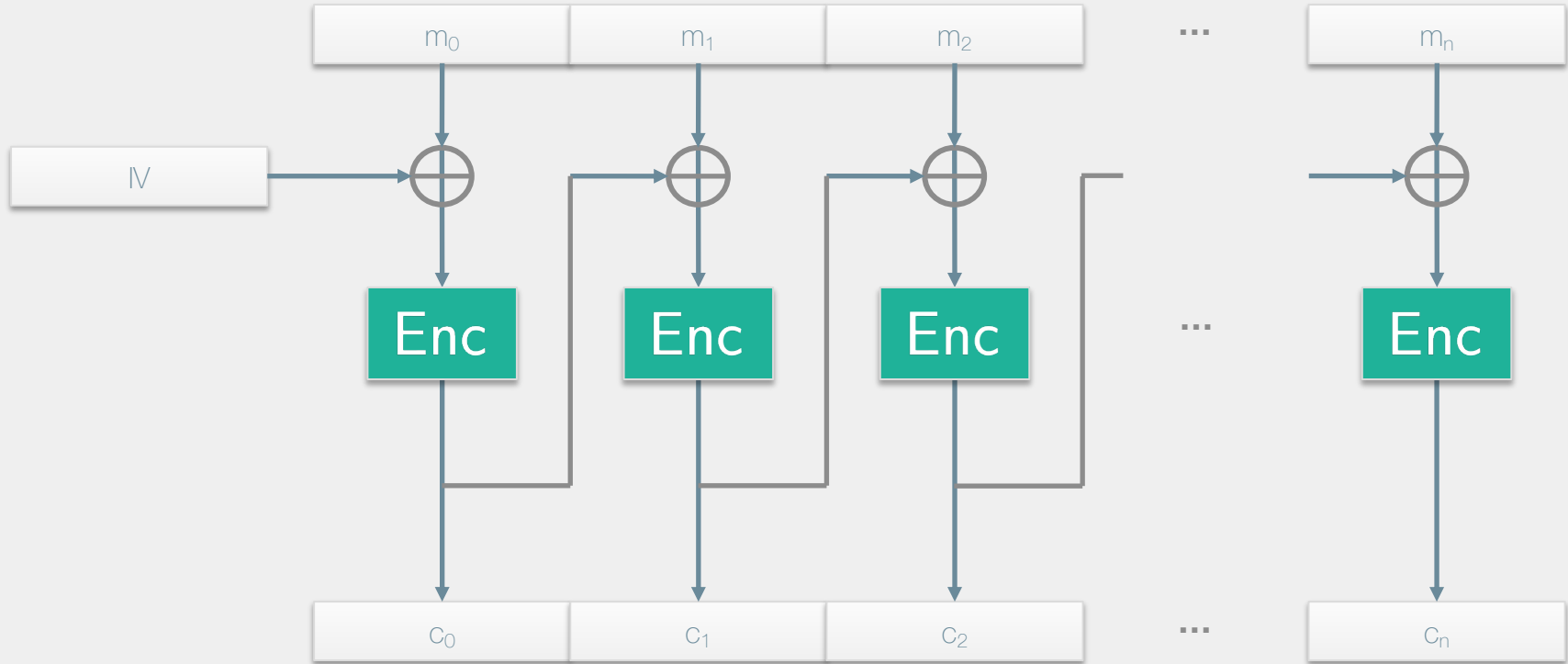
WHAT IF WE COMBINED BLOCKS
TOGETHER SOMEHOW TO BREAK
UP LOCAL STRUCTURE?

ELECTRONIC CODEBOOK MODE (ECB)

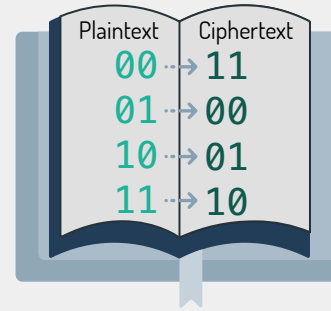
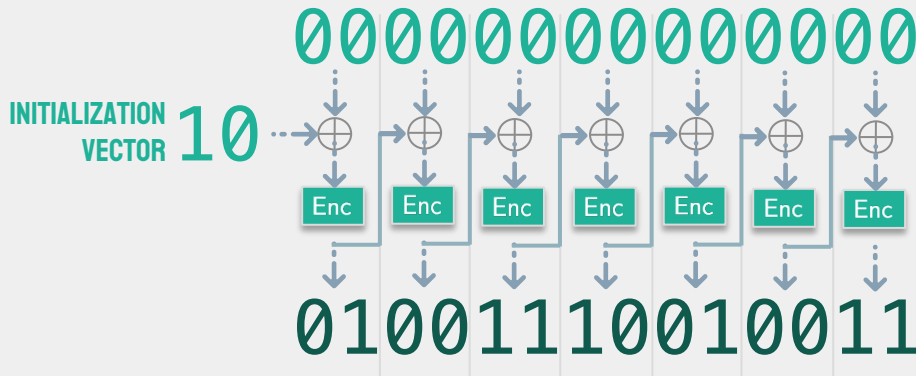


WHAT IF YOU COULD ENCRYPT A
PLAINTEXT TWICE BUT GET TWO
DIFFERENT CIPHERTEXTS?

CIPHER BLOCK CHAINING (CBC)



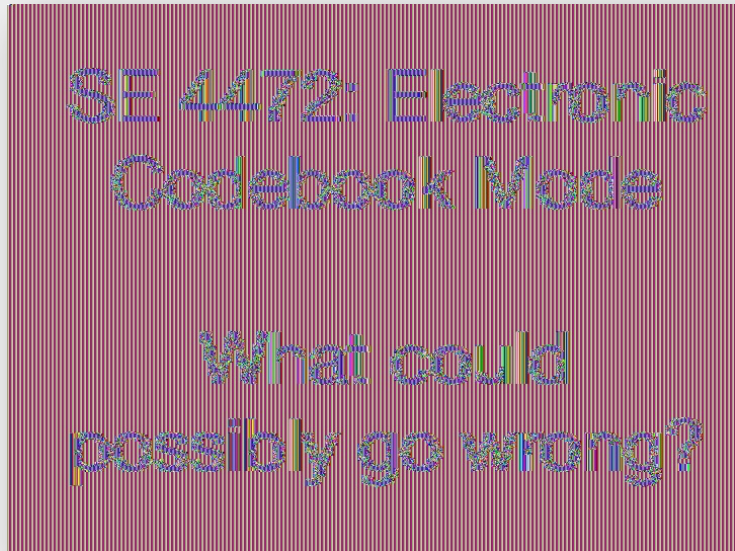
CIPHER BLOCK CHAINING (CBC)



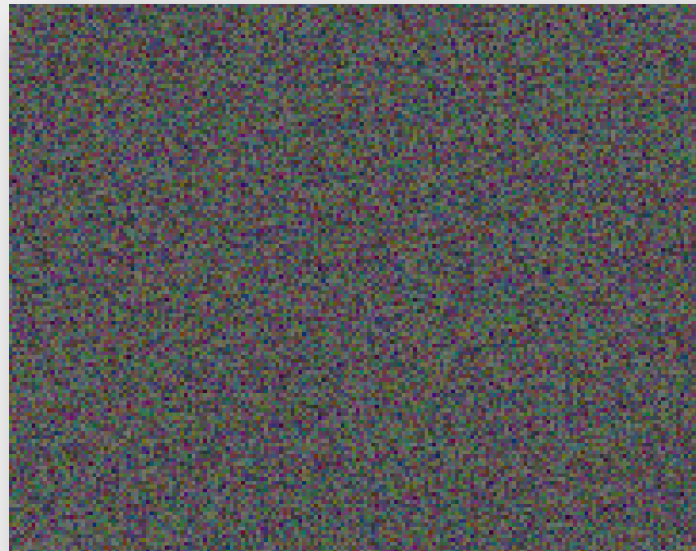
APPROACH

XOR EACH PLAINTEXT BLOCK WITH PREVIOUS CIPHERTEXT BLOCK

CIPHER BLOCK CHAINING (CBC)

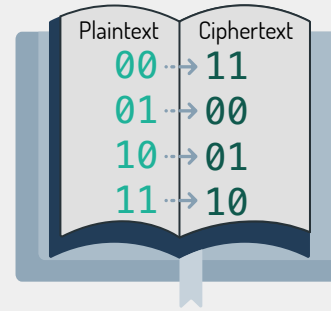
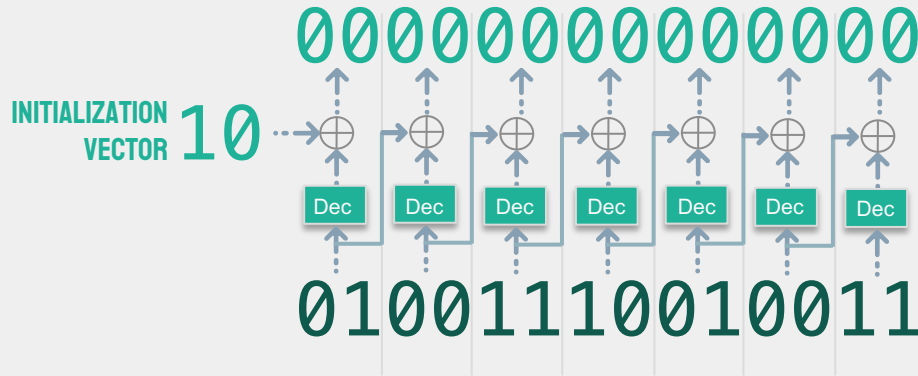


ECB MODE



CBC MODE

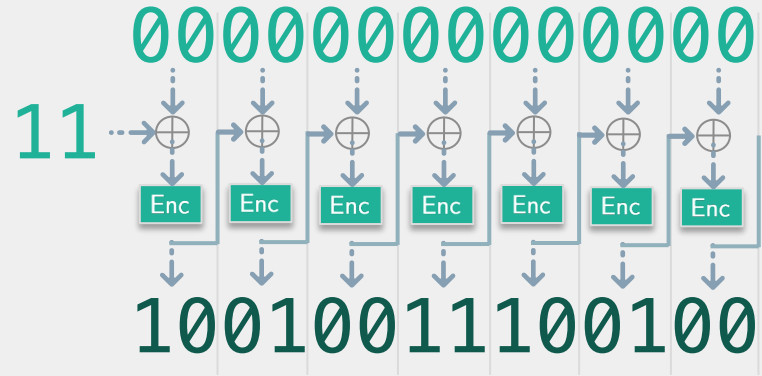
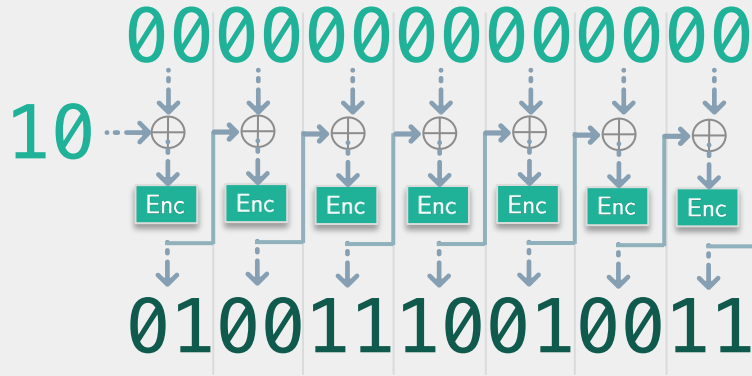
CIPHER BLOCK CHAINING (CBC)



DECRYPTION STILL FEEDS THE PREVIOUS CIPHERTEXT FORWARD

INITIALIZATION VECTOR IS NEEDED TO DECRYPT

CIPHER BLOCK CHAINING (CBC)



RANDOMIZED ENCRYPTION

DIFFERENT INITIALIZATION VECTORS PRODUCE
DIFFERENT CIPHERTEXTS ON THE SAME PLAINTEXT

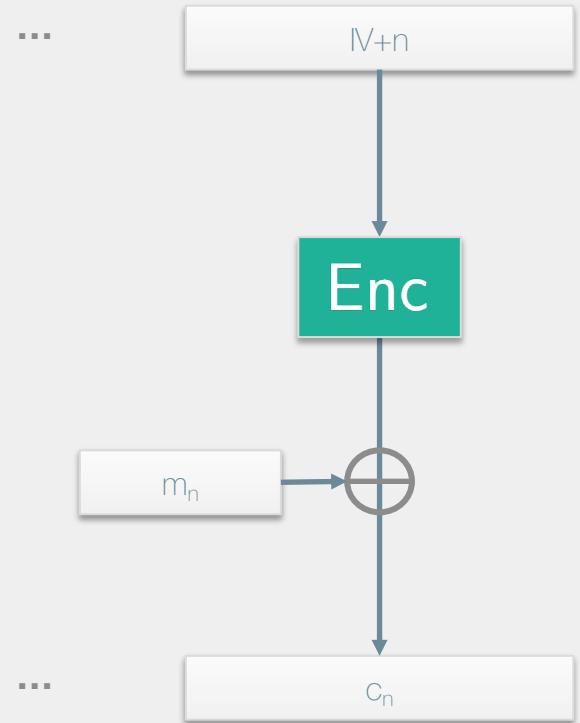
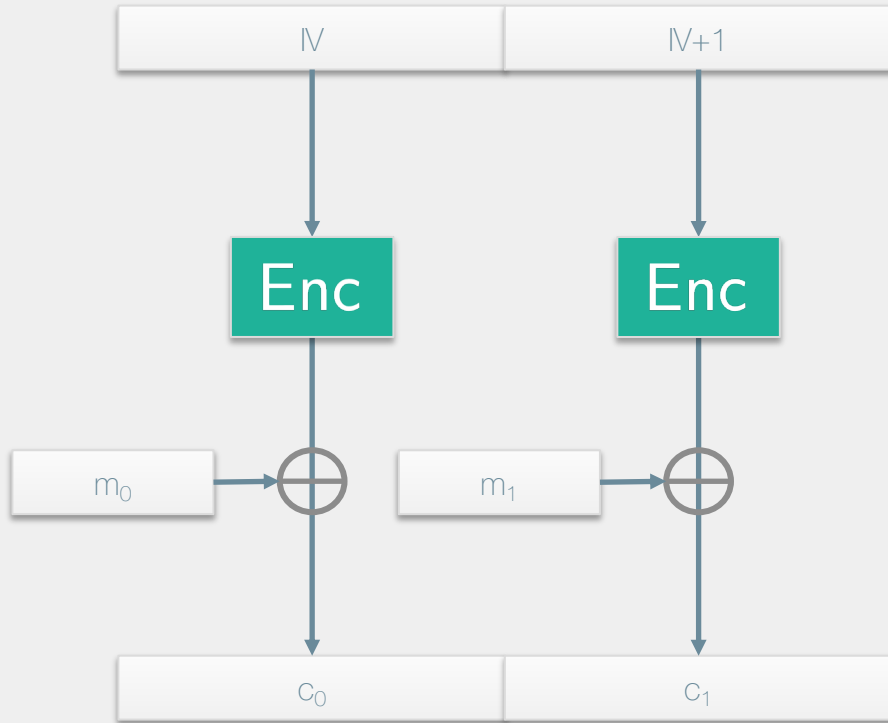
CIPHER BLOCK CHAINING (CBC)



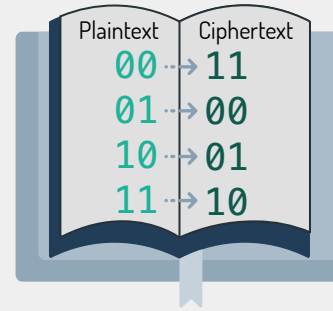
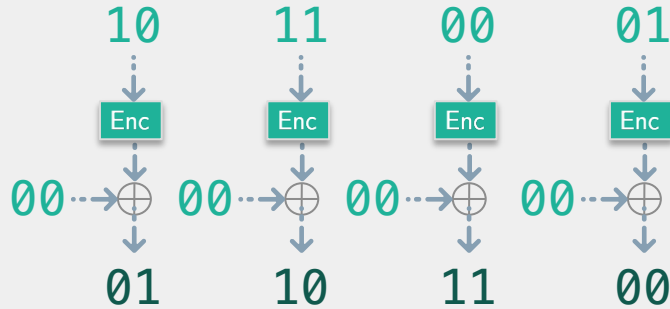
The initialization vector is:

- Not secret
- Sent along with the ciphertext
- Must only ever be used once
- Must not be predictable

COUNTER MODE (CTR)



CIPHER BLOCK CHAINING (CBC)

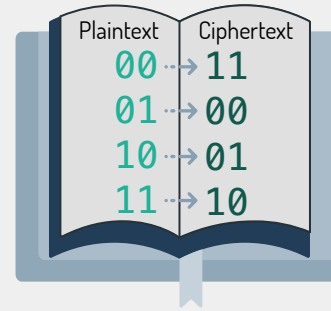
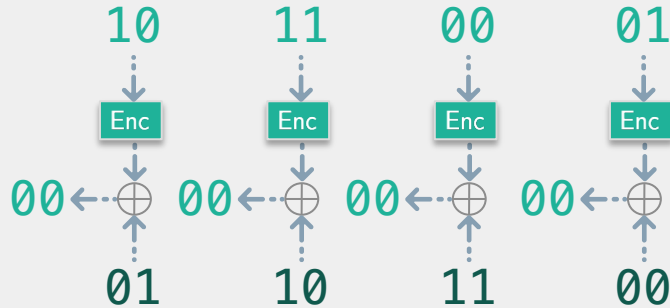


PLAINTEXT: 00000000 IV: 10

APPROACH

ENCRYPT A COUNTER. XOR RESULT WITH PLAINTEXT

CIPHER BLOCK CHAINING (CBC)



CIPHERTEXT: 01101100 IV: 10

DECRYPTION

NOTICE IT USES THE ENCRYPTION FUNCTION!

CIPHER BLOCK CHAINING (CBC)



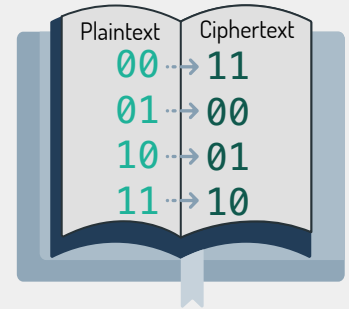
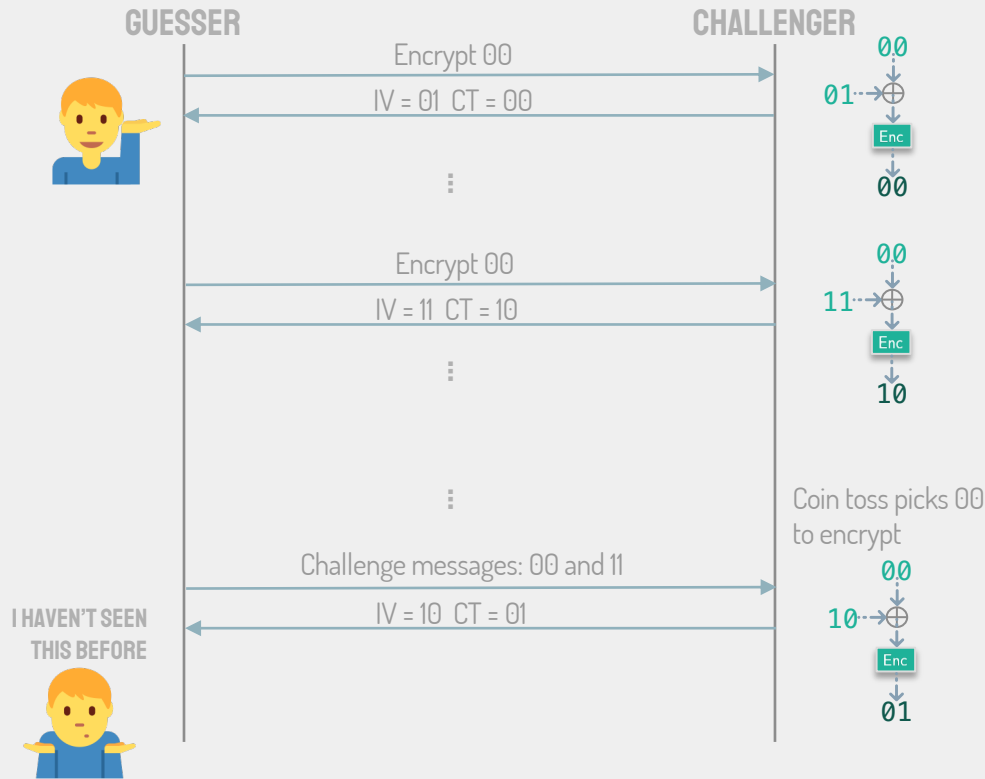
Pros:

- Not vulnerable to padding oracle attacks
- Simpler implementation
- Parallelizable, pre-processable, random access

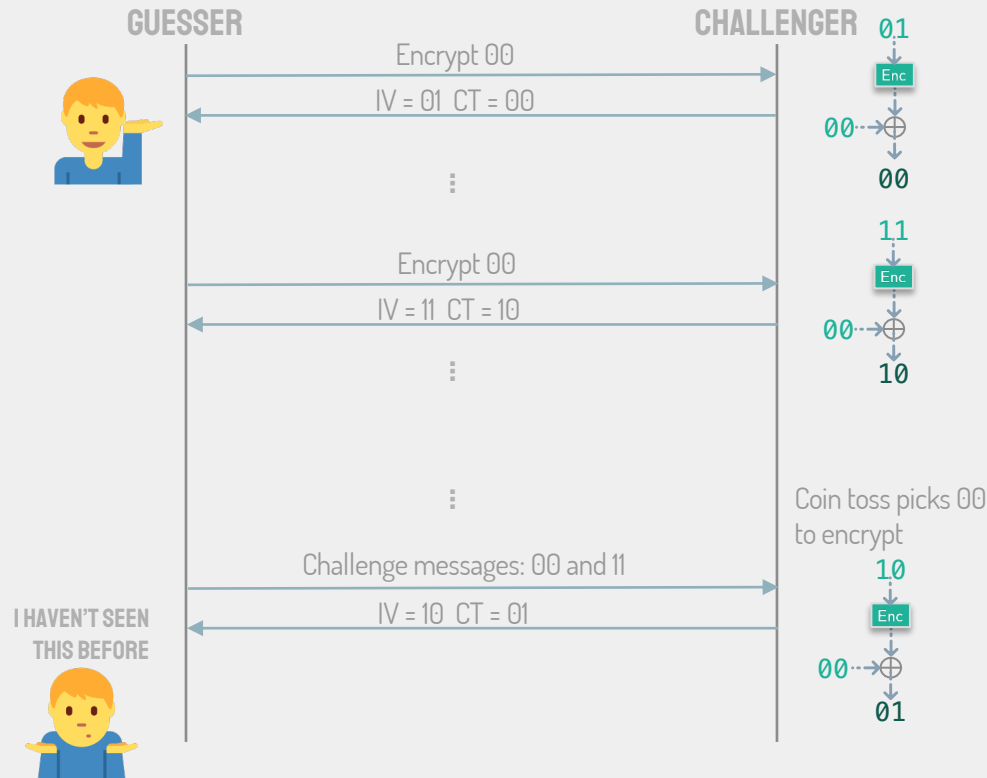
Cons:

- Not safe for small block lengths (<128-bits)

CBC MODE IN THE CPA GAME



CTR MODE IN THE CPA GAME



DEFEATING CHOSEN PLAINTEXT ATTACKS



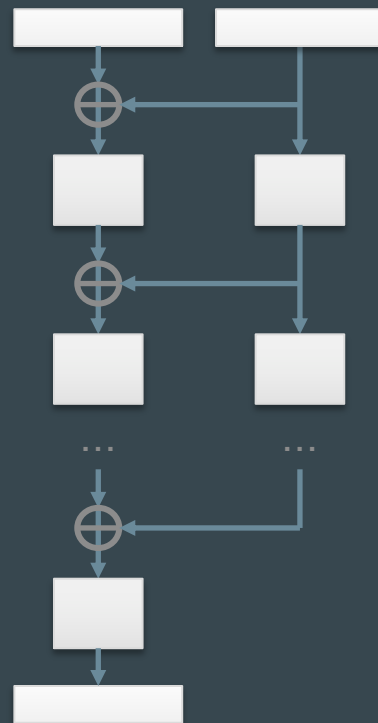
**RANDOMIZED ENCRYPTION IS HOW
WE CAN DEFEAT THE CHOSEN
PLAINTEXT ATTACK**

EXERCISE

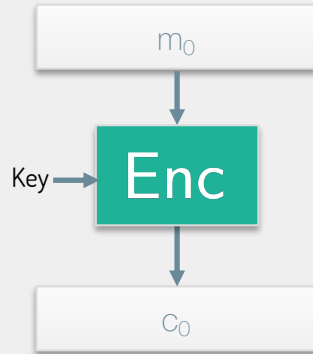
**PROVE CBC MODE IS NOT IND-CPA SECURE IF
YOU CAN PREDICT THE IV/COUNTER**

Hint: Imagine the challenger incremented the IV by 1 with each new encryption. How could you exploit this fact to counteract the contribution of the IV?

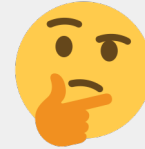
ADVANCED ENCRYPTION STANDARD (AES)



AES ENCRYPTION



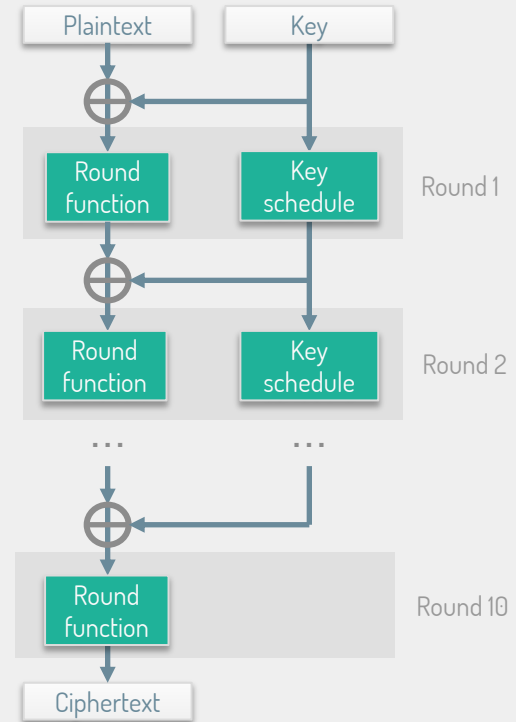
WHAT'S
INSIDE?



AES ENCRYPTION

TOP-LEVEL STRUCTURE

- 128-bit plaintext
- 128-bit ciphertext
- 128-, 192, or 256-bit key
- 10, 12, or 14 identical “rounds”
- Key schedule creates different “sub-keys” for each round



AES ROUND FUNCTION

Input (16 bytes)

b ₀	b ₄	b ₈	b ₁₂
b ₁	b ₅	b ₉	b ₁₃
b ₂	b ₆	b ₁₀	b ₁₄
b ₃	b ₇	b ₁₁	b ₁₅



SubBytes

c ₀	c ₄	c ₈	c ₁₂
c ₁	c ₅	c ₉	c ₁₃
c ₂	c ₆	c ₁₀	c ₁₄
c ₃	c ₇	c ₁₁	c ₁₅



ShiftRows

c ₀	c ₄	c ₈	c ₁₂
c ₅	c ₉	c ₁₃	c ₁
c ₁₀	c ₁₄	c ₂	c ₆
c ₁₅	c ₃	c ₇	c ₁₁

Bytes are permuted



where:

$$c_i = b_i^{-1} + 0x63$$

Byte-wise inverse in GF(256)
plus affine transform

MixColumns

d ₀	d ₄	d ₈	d ₁₂
d ₁	d ₅	d ₉	d ₁₃
d ₂	d ₆	d ₁₀	d ₁₄
d ₃	d ₇	d ₁₁	d ₁₅

where:

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 03 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} c_0 \\ c_5 \\ c_{10} \\ c_{15} \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix}$$

Byte-wise matrix multiplication
in GF(256) of columns



AddRoundKey

e ₀	e ₄	e ₈	e ₁₂
e ₁	e ₅	e ₉	e ₁₃
e ₂	e ₆	e ₁₀	e ₁₄
e ₃	e ₇	e ₁₁	e ₁₅

$$e_i = d_i + k_i$$

Bitwise XOR with roundkey



to next round



TREAT AES LIKE A BLACK BOX

What matters to you is what lives outside:

- A random, **secret key**
- A safe, randomized **mode of operation**
- An unpredictable **initialization vector**



QUESTIONS?

Contact Prof. Essex:

aessex@uwo.ca

[@aleksessex](https://www.instagram.com/aleksessex)

See course website for slides and videos:

<https://whisperlab.org/security>

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik** and illustrations by **Stories**

Please keep this slide for attribution.