Crypto VIII: Two Attacks on Encryption

Cryptographic Hardware for Embedded Systems
ECE 3170

Fall 2025

Assoc. Prof. Vincent John Mooney III
Georgia Institute of Technology

Reading Assignment

- Please read Chapter 3 of the optional course textbook by Katz and Lindell
- NOTE that you are responsible for everything that is explained in lecture!!!

Recall from Cryptography Part I Lecture

Con. 5.38

1) Ciphertext only attack
   - Cryptanalyst has the ciphertext $\{C_i\}$ of a number of messages
     - $C_1 = E_k(P_1)$, $C_2 = E_k(P_2)$, …
2) Known plaintext attack
   - Cryptanalyst has a number of plaintext, ciphertext pairs
     - $(P_i, C_i) \mid C_i = E_k(P_i)$
   - May also have additional ciphertext without associated plaintext
3) Chosen Plaintext Attack (CPA)
   - Cryptanalyst can obtain ciphertext for chosen plaintext
   - Given P_i, $C_i = E_k(P_i)$ can be found
4) Chosen Ciphertext Attack (CCA)
   - Cryptanalyst can obtain plaintext for (some) chosen ciphertext
   - Given C_i, $P_i \mid C_i = E_k(P_i)$ can be found for some (or all) cases

Notation from Katz and Lindell

- {X} is a set of elements of type X
- m is a message in plaintext
  - m is composed of smaller blocks m_i suitable for individual encryption steps
  - $m = \{m_i\}$
- c_i is ciphertext corresponding to message block m_i
- c is ciphertext corresponding to message m
- Enc_k is encryption with key k
  - $c \leftarrow Enc_k(m)$
- Dec_k is decryption with key k
  - $m \leftarrow Dec_k(c)$
- MAC_k is generation of a message authentication code t with key k
  - $t \leftarrow Mac_k(m)$ or, alternatively, $t \leftarrow Mac_k(c)$
- $\langle a,b \rangle$ is a concatenation of a followed by b

F_k

CONSTRUCTION 3.30 (page 83 in Ch. 3 of K & L)

- F_k is a pseudorandom function which varies with a key k
  - Note: we will not cover elliptic curves in this course, but F_k can be implemented by such curves (this is known as elliptic curve cryptography)
- A uniformly random n-bit key is selected and provided to the sender and receiver (but not to the adversary, of course)
- Enc_k : given an n-bit message m, choose a uniformly random n-bit number r
  - $c := \langle r, F_k(r) \oplus m \rangle$

$$c = \langle r, F_k(r) \oplus m \rangle$$

- Dec_k : given length $2n$ ciphertext $c = \langle r,s \rangle$
  - $m := F_k(r) \oplus s = F_k^{-1}(c)$

$$F_k(r) \oplus s = \overset{0}{\cancel{F_k(r) \oplus F_k(r)}} \oplus m = m$$

Chosen Ciphertext Attack (CCA)

- Katz and Lindell define CCA indistinguishability in Section 3.7.1 (page 96 of the second edition of their book) as follows
- Generate a uniformly random key k of length n
- Adversary A is given oracle access to Enc_k and Dec_k but is not allowed to query the actual challenge ciphertext

polynomial

- A chooses two messages m_0 and m_1
- $b \in \{0,1\}$ is chosen and is hidden from A

$Enc_k(m_b)$

- $c \leftarrow Enc_k(m_b)$ is given to A
- Test: given c, can A distinguish which case was encrypted?
- For example, consider m_0 = a plaintext of all zeros and m_1 = a plaintext of all ones

The Adversary Wins

$C = \langle r, s \rangle$ $\boxed{M_0 = 000\cdots0}$

$M_1 = 111\cdots1$

$C = \langle r, s' \rangle$

- Approach:
  - take s and flip the most significant bit, resulting in s'
  - decrypt r, s' $F_K(r) \oplus s' = 10000\cdots0$
  - if the answer of decryption is a 1 followed by all zeros, the original message was all zeros $= 01111\cdots1$
  - if the answer of decryption is a 0 followed by all ones, the original message was all ones

$s = 1010111\cdots01$

$s' = 0010111\cdots01$

Would this attack work with

$c_0 = DES_k(m_0)$

$c_1 = DES_k(m_1)$

What if we encrypt m_0

$c_0 = Enc(m_0)$

Takeaway

- Any encryption scheme which allows ciphertexts to “manipulated” in any controlled manner or way cannot be CCA-secure
- It is better if encryption schemes have the property that if the adversary tries to modify a given ciphertext, the results decrypts to a plaintext having no relationship to the original plaintext
  - Is enough to have no detectable relationship, i.e., which can be detected by a sequence of steps including an algorithm written in computer code

RECALL: Cipher Block Chaining

- Use results of previous block encryption
- Typical use is based on exclusive-or (XOR)
  - For encryption where $i > 1$ (i.e., after the first block), $C_i = E_k(P_i \oplus C_{i-1})$
  - For decryption (except for the first block, i.e., $i \neq 1$), $P_i = C_{i-1} \oplus D_k(C_i)$

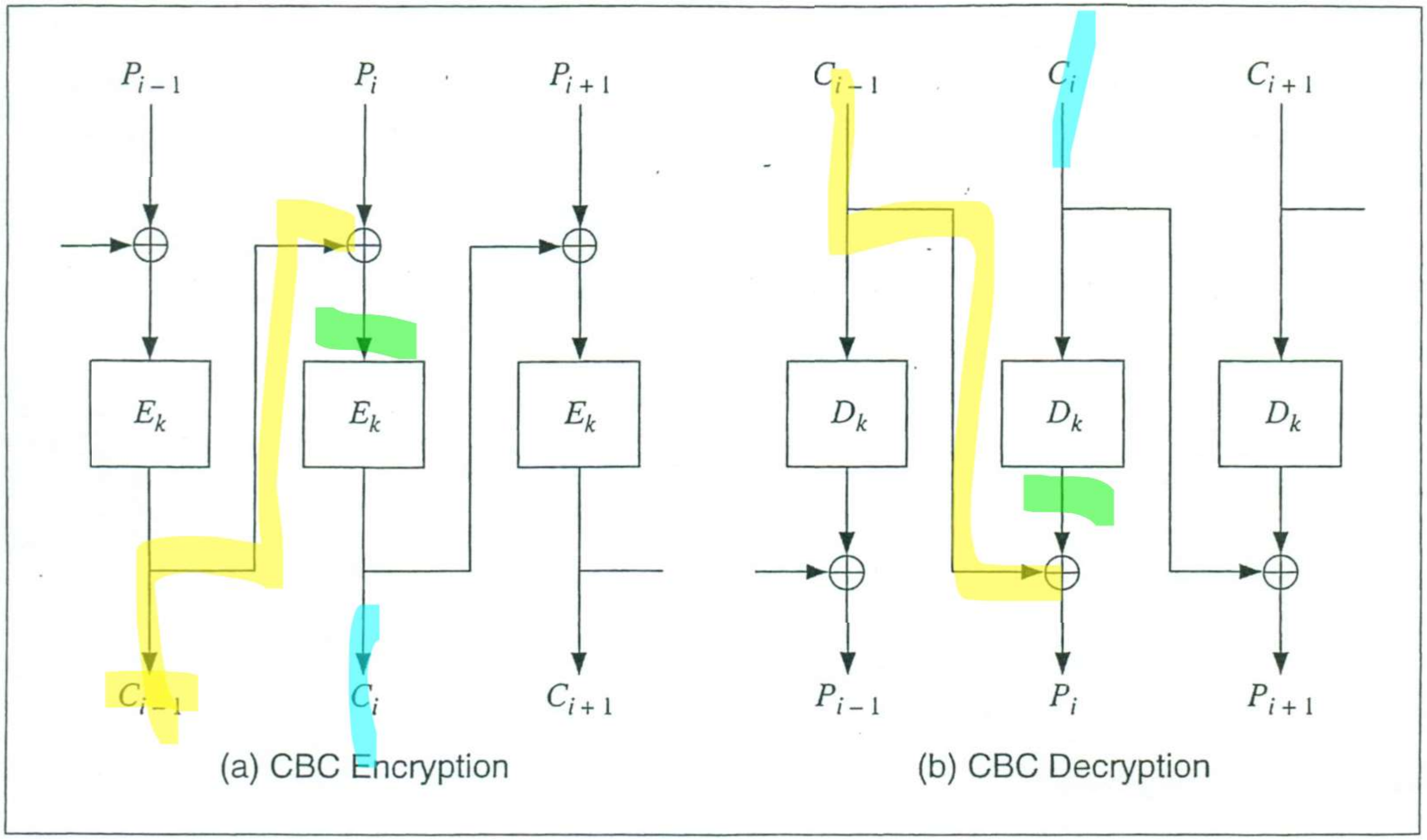


Figure 9.3 Cipher block chaining mode.

Insecure against the Padding Oracle Attack

- In the previous attack on the earlier slides, the adversary was given access to Enc_k and Dec_k but is not allowed to query the actual challenge ciphertext
  - Such access to Enc_k and Dec_k unlikely to happen in practice
- Here we consider an attack based on much less information
  - The adversary is informed if a modified ciphertext decrypts correctly
  - Such information is frequently easy to obtain
    - Retransmission request
    - Session termination
- "The attack has been shown to work in practice on various deployed protocols; we give one concrete example at the end of this section." (Page 98 of Katz and Lindell 2nd Edition, Section 3.7.2)

Set Up

L = 8 → 64 bits

- Cipher Block Chaining (CBC) mode, block length *L* (measured in bytes)
- Message *m* has some number of bytes but must be a multiple of *L*

m = 128 bits = 16B = 8x2B

m = 96 bits

- PKCS #5 padding
  - Let *b* be the number of bytes appended to *m*
    - Do not allow *b* = 0 in order to avoid ambiguous padding
    - If *m* is already a multiple of *L*, then add *L* bytes of padding
  - Append to the end of *m* a string containing *b* repeated *b* times
    - E.g., using hexadecimal format for each byte, if *b* = 1 then append 0x01
    - if *b* = 4 then append 0x04040404
- The padded message is then encrypted and sent

0x0808080808080808

Decryption

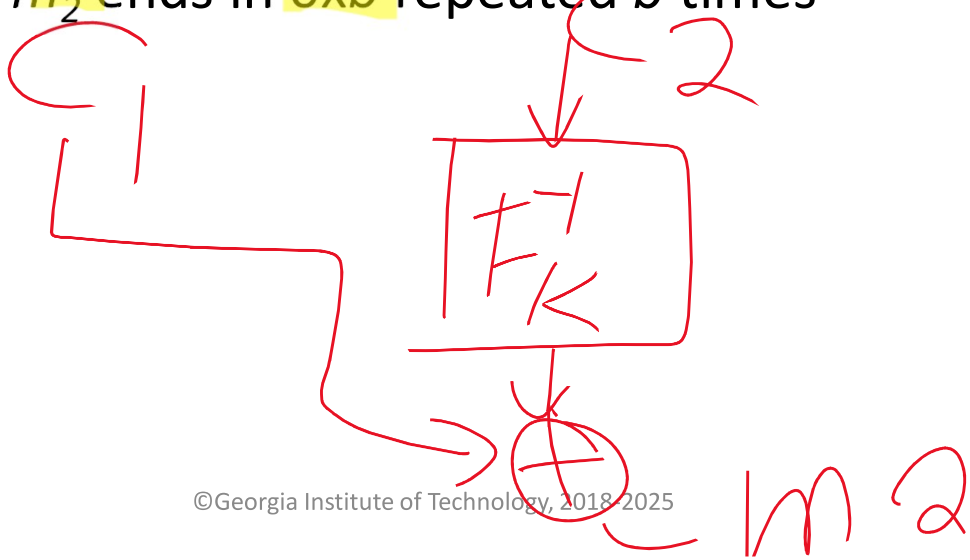
$L = 8$

- Padded data is decrypted using Construction 3.30 in CBC mode
- After decryption, the message is checked for correct padding
  - Simply read the last byte
  - The value *b* of the last byte should be repeated *b* times
- If the padding is found to be correct, it is stripped from the message
- Otherwise, a standard procedure is to return a “bad padding” error
  - E.g., in Java, `javax.crypto.BadPaddingException`
- Such an error message provides an adversary with a partial decryption oracle

The Padding Oracle Attack on a ~~3~~ 2-block Message

w/ IV

- Attacker observes IV, c_1, c_2 where IV is the Initialization Vector
- Let the correct message decrypted be m_1, m_2
- Note that $m_2 = F_k^{-1}(c_2) \oplus c_1$ where key k is not known to the attacker
- Further note that m_2 ends in 0xb repeated b times



$c_2 = 8$ Bytes

Attack Technique: Send IV, c_1', c_2 for Decryption

- Let c_1' be identical to c_1 except for the ~~final~~ most sig byte
- Consider IV, c_1', c_2 : decryption will result in m_1', m_2'
  - Will have $m_2' = F_k^{-1}(c_2) \oplus c_1'$
  - Recall $m_2 = F_k^{-1}(c_2) \oplus c_1$
  - $\Rightarrow$ m_2' and m_2 differ only in the final byte
- Note that the value of m_1' has no discernable relationship to m_1, but this will not matter for the attack to succeed
- Similarly, if c_1' is identical to c_1 except for byte i, then m_2' and m_2 differ only in the i^{th} byte
- In general, if $c_1' = c_1 \oplus \Delta$, then $m_2' = m_2 \oplus \Delta$

c_1' c_2 F_k^{-1} F_k^{-1} m_1' m_2'

First Step: Discover the Padding Length

- Let c_1' be identical to c_1 except for the most significant byte of the total number of L bytes
- Send IV, c_1', c_2 : if there is a padding error, the message has length L bytes
- Otherwise now let c_1'' be identical to c_1 except for the second most significant byte

  if $L = 8$

- Send IV, c_1'', c_2 : if there is a padding error, the message has length L-1 $= 7$
- Otherwise now let c_1''' be identical to c_1 except for the third most significant byte
- Send IV, c_1'', c_2 : if there is a padding error, the message has length L-2
- Otherwise…
- Continuing in this fashion, the padding length is discovered

Once have padded bits length

Comment

- Note that we now have some of the plaintext of the final message
- Recall Cryptography Part I lecture:

2) Known plaintext attack
   - Cryptanalyst has a number of plaintext, ciphertext pairs
     - $(P_i, C_i) \mid C_i = E_k(P_i)$
   - May also have additional ciphertext without associated plaintext

Next Step: Discover the Final Message Byte

least significant

- We have currently that m_2 = … B1 B0 0xb … 0xb
  - Where message bytes … B1 B0 are not yet known to the attacker
  - We aim now to discover the final message byte B0
- Recall that if $c_1' = c_1 \oplus \Delta$, then $m_2' = m_2 \oplus \Delta$
- Define Δ_i = 0x00 … 0x00 0xi 0x(b+1) … 0x(b+1)

  $\oplus$ 0x00 … 0x00 0x00 0xb … 0xb
  - where $0 \le i < 2^8$
- Send $IV, c_1 \oplus \Delta_i, c_2 \Rightarrow m_2'$ = … B1 0x(B0⊕i) 0x(b+1) … 0x(b+1)
- Whenever 0x(B0⊕i) = 0x(b+1) will not have a padding error anymore

B0⊕i = 0x06

6 6 6 6 6

©Georgia Institute of Technology, 2018-2025 19

$\Delta = \boxed{1000}0 \cdots 0$

64 bits, 63 zeros

$\Delta = 0x80\ 00\ 00\ 00\ 00\ 00\ 00\ 00 \cdots$

$b = 0x05 = 00000101$

$$\begin{array}{r} b+1 = 0x06 = \underline{00000110} \\ 11 \end{array}$$

$i = 12 = x0C$

$\Delta = 0x00000C0303030303$

$$0x05 \oplus 0x03$$

$$= 00000101 \oplus 00000011$$

$$= 00000110$$

$$= 0x06$$

RECALL: Cipher Block Chaining Summary

- Use results of previous block encryption
- Typical use is based on exclusive-or (XOR)
  - For encryption where $i > 1$ (i.e., after the first block), $C_i = E_k(P_i \oplus C_{i-1})$
  - For decryption (except for the first block, i.e., $i \neq 1$), $P_i = C_{i-1} \oplus D_k(C_i)$

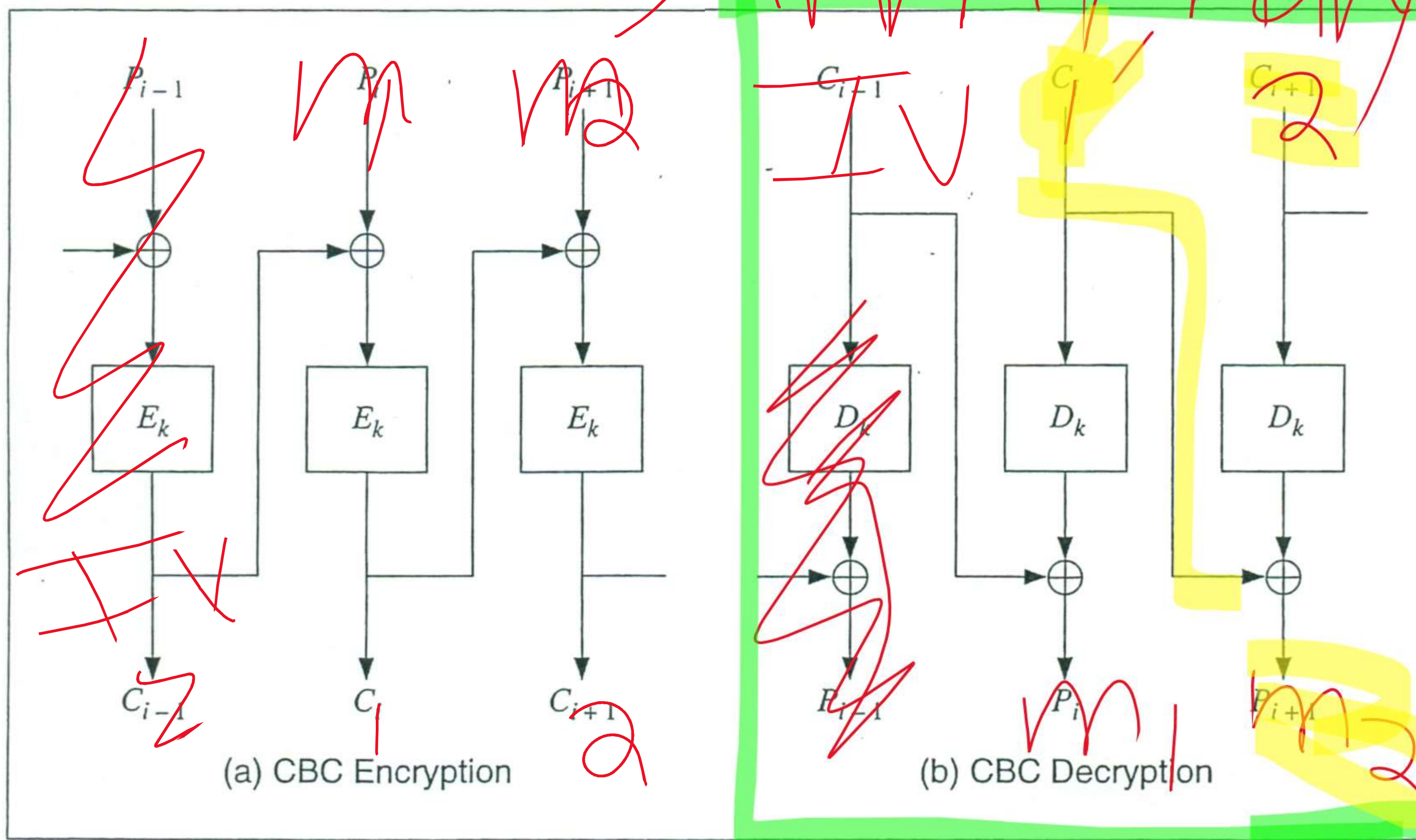


Figure 9.3 Cipher block chaining mode.

64-bit number

2^{64} poss,

128-bit numbe

2^{128}

but we find
a way to
look at a Byte
$\Rightarrow 2^8$ poss!

2^{128}

safe

$16 \times 2^8 = 16 \times 256$

$2^4 \times 2^{12} = 2^{16}$