Cryptography

Lecture 10

Arkady Yerukhimovich

September 30, 2024

Outline

1. Lecture ~~8~~ 9 Review
2. Chosen-ciphertext Attack (CCA) Security (Chapter 3.7)
3. Importance of CCA Security (Chapter 3.7)

Lecture 8 Review

- Proof of CPA-security for PRF+OTP
- Modes of operations

Outline

1. Lecture 8 Review
2. Chosen-ciphertext Attack (CCA) Security (Chapter 3.7)
3. Importance of CCA Security (Chapter 3.7)

Chosen-ciphertext Attack

- CPA security captures scenario where $\mathcal{A}$ may trick parties to encrypt messages on his behalf

Chosen-ciphertext Attack

- CPA security captures scenario where $\mathcal{A}$ may trick parties to encrypt messages on his behalf
- But what if $\mathcal{A}$ can also trick parties to decrypt (some) ciphertexts for him.

Chosen-ciphertext Attack

- CPA security captures scenario where $\mathcal{A}$ may trick parties to encrypt messages on his behalf
- But what if $\mathcal{A}$ can also trick parties to decrypt (some) ciphertexts for him.
  - May be enough to just get partial decryptions
  - Security against such an attack is not addressed by CPA security

Chosen-ciphertext Attack

- CPA security captures scenario where $\mathcal{A}$ may trick parties to encrypt messages on his behalf
- But what if $\mathcal{A}$ can also trick parties to decrypt (some) ciphertexts for him.
  - May be enough to just get partial decryptions
  - Security against such an attack is not addressed by CPA security
- Want undecrypted messages to remain secure

Is PRF+OTP CCA Secure?

PRF+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- $\mathsf{Dec}(k,c)$: Parse c as (r, c'), compute $m = F_k(r) \oplus c'$

Is this CCA Secure?

Is PRF+OTP CCA Secure?

PRF+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k, m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- $\mathsf{Dec}(k, c)$: Parse c as (r, c'), compute $m = F_k(r) \oplus c'$

The Attack:

- $\mathcal{A}$ receives ciphertext $c = (r^*, F_k(r^*) \oplus m)$

Is PRF+OTP CCA Secure?

PRF+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k, m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- $\mathsf{Dec}(k, c)$: Parse c as (r, c'), compute $m = F_k(r) \oplus c'$

The Attack:

- $\mathcal{A}$ receives ciphertext $c = (r^*, F_k(r^*) \oplus m)$
- $\mathcal{A}$ constructs forged ciphertext $\overline{c} = (r^*, 0^n)$, and queries $\mathsf{Dec}_k(\overline{c})$

Is PRF+OTP CCA Secure?

PRF+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k, m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- $\mathsf{Dec}(k, c)$: Parse c as (r, c'), compute $m = F_k(r) \oplus c'$

The Attack:

- $\mathcal{A}$ receives ciphertext $c = (r^*, F_k(r^*) \oplus m)$
- $\mathcal{A}$ constructs forged ciphertext $\overline{c} = (r^*, 0^n)$, and queries $\mathsf{Dec}_k(\overline{c})$
- $\mathsf{Dec}_k(\overline{c})$ returns $\overline{m} = F_k(r^*) \oplus 0^n = F_k(r^*)$

Is PRF+OTP CCA Secure?

PRF+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- $\mathsf{Dec}(k,c)$: Parse c as (r, c'), compute $m = F_k(r) \oplus c'$

The Attack:

- $\mathcal{A}$ receives ciphertext $c = (r^*, F_k(r^*) \oplus m)$
- $\mathcal{A}$ constructs forged ciphertext $\overline{c} = (r^*, 0^n)$, and queries $\mathsf{Dec}_k(\overline{c})$
- $\mathsf{Dec}_k(\overline{c})$ returns $\overline{m} = F_k(r^*) \oplus 0^n = F_k(r^*)$
- $\mathcal{A}$ can now use $F_k(r^*)$ to decrypt c

Is PRF+OTP CCA Secure?

PRF+OTP Encryption

- Gen(1^n): $k \leftarrow \{0,1\}^n$
- Enc(k, m): Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- Dec(k, c): Parse c as (r, c'), compute $m = F_k(r) \oplus c'$

The Attack:

- $\mathcal{A}$ receives ciphertext $c = (r^*, F_k(r^*) \oplus m)$
- $\mathcal{A}$ constructs forged ciphertext $\overline{c} = (r^*, 0^n)$, and queries $\underline{\mathrm{Dec}_k(\overline{c})}$
- $\mathrm{Dec}_k(\overline{c})$ returns $\overline{m} = F_k(r^*) \oplus 0^n = F_k(r^*)$
- $\mathcal{A}$ can now use $F_k(r^*)$ to decrypt c

Takeaway

PRF+OTP is not CCA-Secure

Defining CCA-Secure Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

Defining CCA-Secure Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$

Defining CCA-Secure Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.

Defining CCA-Secure Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$

Defining CCA-Secure Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0, 1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)}$ outputs a guess bit b' ($\mathcal{A}$ may not query $\mathsf{Dec}_k(c)$)

Defining CCA-Secure Encryption

Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\text{PrivK}^{cca}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \text{Gen}(1^n)$
- $\mathcal{A}^{\text{Enc}_k(\cdot),\text{Dec}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \text{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\text{Enc}_k(\cdot),\text{Dec}_k(\cdot)}$ outputs a guess bit b' ($\mathcal{A}$ may not query $\text{Dec}_k(c)$)
- We say that $\text{PrivK}^{cca}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Defining CCA-Secure Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)}$ outputs a guess bit b' ($\mathcal{A}$ may not query $\mathsf{Dec}_k(c)$)
- We say that $\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

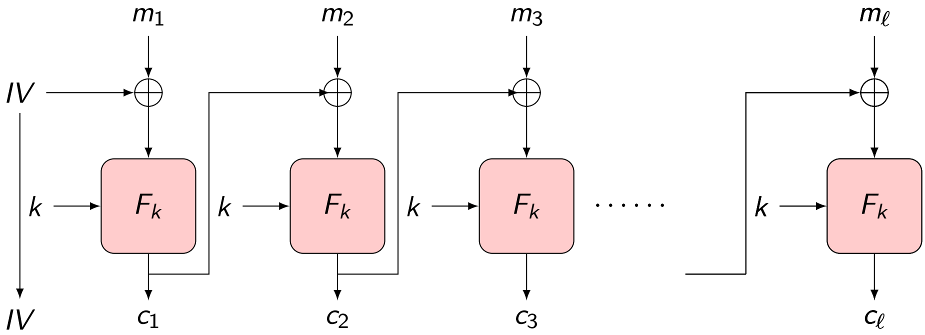
Definition: An encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$ is CCA-secure if for all PPT $\mathcal{A}$ it holds that

$$\Pr[\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi}(n) = 1] \leq 1/2 + \mathsf{negl}(n)$$

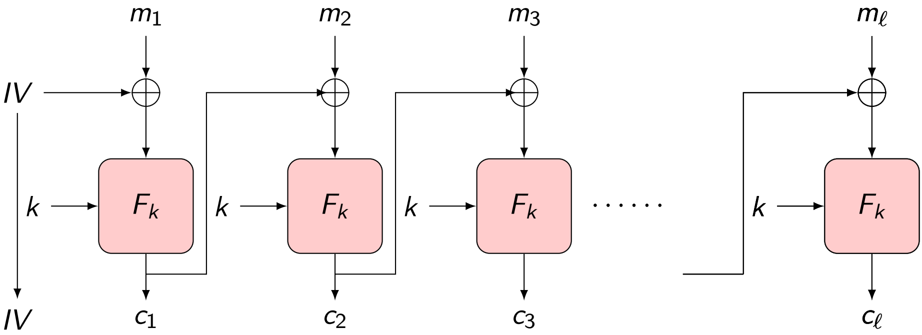
Outline

1. Lecture 8 Review
2. Chosen-ciphertext Attack (CCA) Security (Chapter 3.7)
3. Importance of CCA Security (Chapter 3.7)

Padding Oracle Attack on CBC Mode

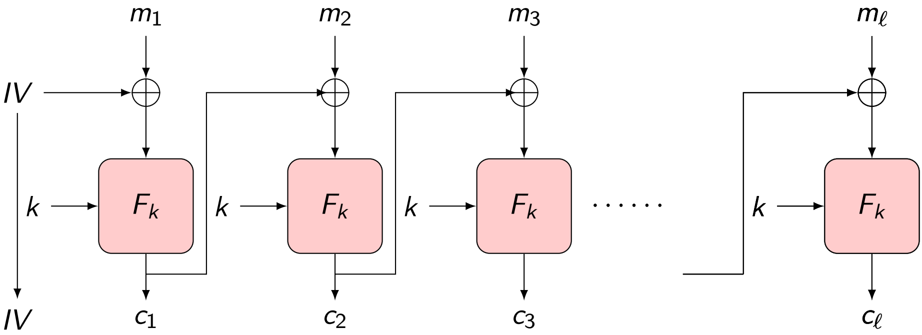


Padding Oracle Attack on CBC Mode



- This assumes that $|m|$ is a multiple of block-length L.

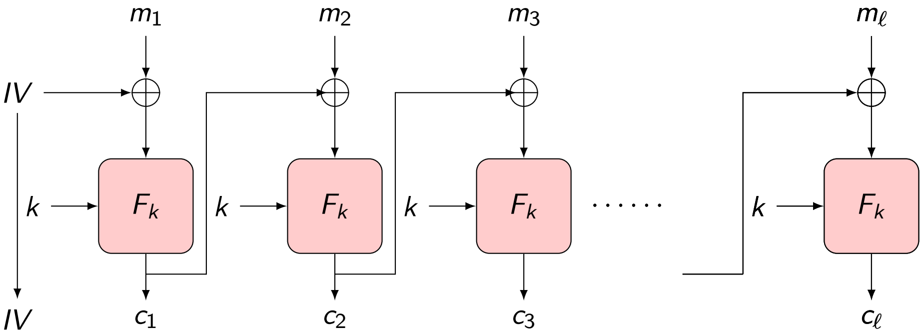
Padding Oracle Attack on CBC Mode



- This assumes that $|m|$ is a multiple of block-length L.
- If it is not, standard approach is to pad m to a multiple of L
  - Need to be able to tell what is part of m and what is padding

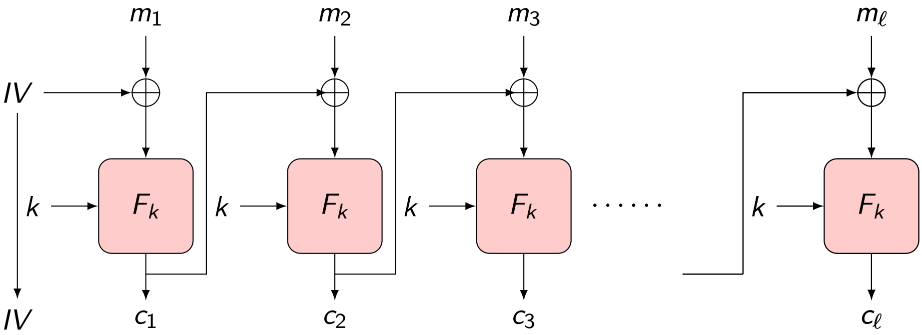
m_1 m_2

Padding Oracle Attack on CBC Mode



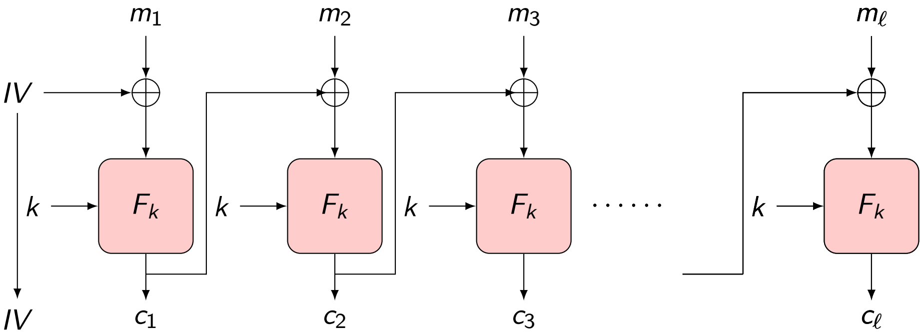
- This assumes that $|m|$ is a multiple of block-length L.
- If it is not, standard approach is to pad m to a multiple of L
  - Need to be able to tell what is part of m and what is padding
  - Add 1 to L bytes to end of m to pad to next multiple of L.

Padding Oracle Attack on CBC Mode



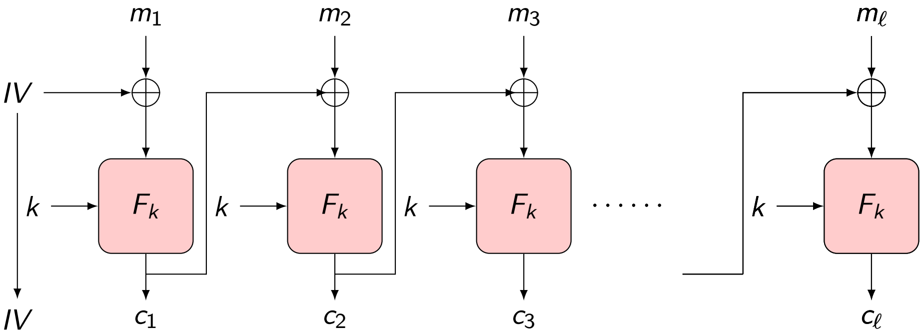
- This assumes that $|m|$ is a multiple of block-length L.
- If it is not, standard approach is to pad m to a multiple of L
  - Need to be able to tell what is part of m and what is padding
  - Add 1 to L bytes to end of m to pad to next multiple of L.
  - To identify padding, pad value indicates number of Bytes of padding

Padding Oracle Attack on CBC Mode



- This assumes that $|m|$ is a multiple of block-length L.
- If it is not, standard approach is to pad m to a multiple of L
  - Need to be able to tell what is part of m and what is padding
  - Add 1 to L bytes to end of m to pad to next multiple of L.
  - To identify padding, pad value indicates number of Bytes of padding
  - Example: $m' = m||0x2||0x2$ if need 2 Bytes of padding

Padding Oracle Attack on CBC Mode

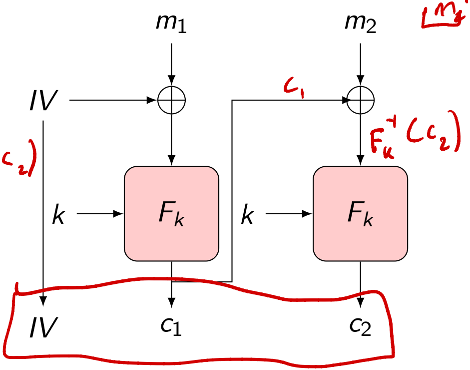


- This assumes that $|m|$ is a multiple of block-length L.
- If it is not, standard approach is to pad m to a multiple of L
  - Need to be able to tell what is part of m and what is padding
  - Add 1 to L bytes to end of m to pad to next multiple of L.
  - To identify padding, pad value indicates number of Bytes of padding
  - Example: $m' = m||0x2||0x2$ if need 2 Bytes of padding
- Decryption can then remove padding and return m
  - If padding incorrect, return "bad padding" error

Padding Oracle Attack on CBC Mode

$m_2 = c_1 \oplus F_k^{-1}(c_2)$

$m_2 \oplus \delta = (c_1 \oplus \delta) \oplus F_k^{-1}(c_2)$

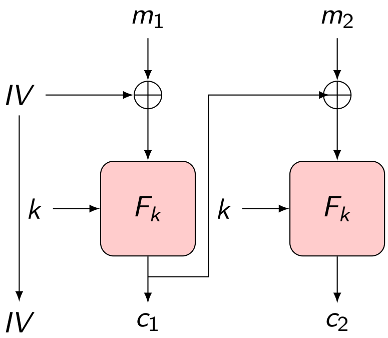


- Consider encryption of a 2-block message m

Quiz

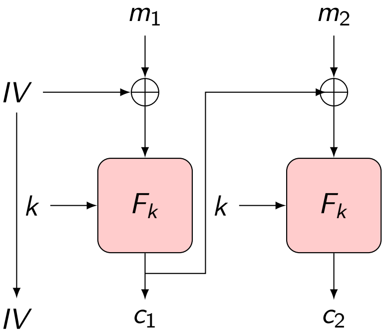
You will now develop an attack on this mode of operations.

Padding Oracle Attack on CBC Mode



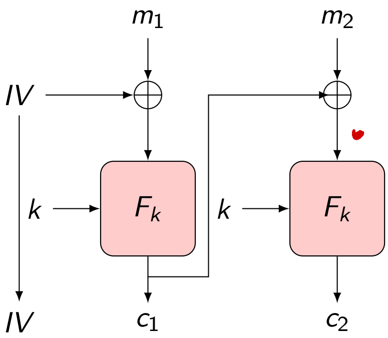
- Consider encryption of a 2-block message m

Padding Oracle Attack on CBC Mode



- Consider encryption of a 2-block message m
- Note that $m_2 = F_k^{-1}(c_2) \oplus c_1$

Padding Oracle Attack on CBC Mode



- Consider encryption of a 2-block message m
- Note that $m_2 = F_k^{-1}(c_2) \oplus c_1$
  - If we change c_1 to $c_1' = c_1 \oplus \delta$ without changing c_2 then, we change m_2 to $m_2' = m_2 \oplus \delta$

Padding Oracle Attack

Observation: We know that m_2 ends in $(0xb)$ repeated b times

Padding Oracle Attack

Observation: We know that m_2 ends in $(0xb)$ repeated b times

Step 1: Learn size of padding

Padding Oracle Attack

Observation: We know that m_2 ends in $(0xb)$ repeated b times

Step 1: Learn size of padding

- Change 1st Byte of c_1 (thus, also m_2) and see if error occurs
    - Error only occurs if $|pad| = L$

Padding Oracle Attack

Observation: We know that m_2 ends in $(0xb)$ repeated b times

Step 1: Learn size of padding

- Change 1st Byte of c_1 (thus, also m_2) and see if error occurs
  - Error only occurs if $|pad| = L$
- Change Bytes $2, \ldots, L$ until first time we get error, this is first Byte of padding

Padding Oracle Attack

Observation: We know that m_2 ends in $(0xb)$ repeated b times

Step 2: Using knowledge of $b = |pad|$, decrypt m

Padding Oracle Attack

Observation: We know that m_2 ends in $(0xb)$ repeated b times

Step 2: Using knowledge of $b = |pad|$, decrypt m

- Change c_1 (and thus also m_2) by δ_i defined as

$$\begin{aligned}\delta_i \quad = \quad & \overbrace{0x00||\cdots||0x00}^{L-(b+1)\text{ Bytes}}||0x(i)||\overbrace{0x(b+1)||\cdots||0x(b+1)}^{b\text{ Bytes}} \oplus \\ & \underbrace{0x00||\cdots||0x00||0x00}_{L-b\text{ Bytes}}||\underbrace{0xb||\cdots||0xb}_{b\text{ Bytes}}\end{aligned}$$

m' $\underbrace{b}$ $\underbrace{b}$ $\underbrace{b}$

$\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$

$b+1$ $b+1$ $b+1$ $b+1$

Padding Oracle Attack

Observation: We know that m_2 ends in $(0xb)$ repeated b times

Step 2: Using knowledge of $b = |pad|$, decrypt m

- Change c_1 (and thus also m_2) by δ_i defined as

$$\begin{aligned}\delta_i \;=\;& \overbrace{0x00||\cdots||0x00}^{L-(b+1)\text{ Bytes}}||0x(i)||\overbrace{0x(b+1)||\cdots||0x(b+1)}^{b\text{ Bytes}} \oplus \\ & \underbrace{0x00||\cdots||0x00||0x00}_{L-b\text{ Bytes}}||\underbrace{0xb||\cdots||0xb}_{b\text{ Bytes}}\end{aligned}$$

- $m_2 \oplus \delta_i = \overbrace{m_2^1||\cdots||m_2^{L-(b+1)}}^{L-(b+1)\text{ Bytes}}||(0x(i) \oplus m_2^{L-b})||\overbrace{0x(b+1)||\cdots}^{b\text{ Bytes}}$

Padding Oracle Attack

Observation: We know that m_2 ends in $(0xb)$ repeated b times

Step 2: Using knowledge of $b = |pad|$, decrypt m

- Change c_1 (and thus also m_2) by δ_i defined as

$$\delta_i = \overbrace{0x00||\cdots||0x00}^{L-(b+1) \text{ Bytes}}||0x(i)||\overbrace{0x(b+1)||\cdots||0x(b+1)}^{b \text{ Bytes}} \oplus$$
$$\underbrace{0x00||\cdots||0x00||0x00}_{L-b \text{ Bytes}}||\underbrace{0xb||\cdots||0xb}_{b \text{ Bytes}}$$

- $m_2 \oplus \delta_i = \overbrace{m_2^1||\cdots||m_2^{L-(b+1)}}^{L-(b+1) \text{ Bytes}}||(0x(i) \oplus m_2^{L-b})||\overbrace{0x(b+1)||\cdots}^{b \text{ Bytes}}$
- Will only decrypt correctly if $0xi \oplus m_2^{L-b} = 0x(b+1)$

Padding Oracle Attack

Observation: We know that m_2 ends in $(0xb)$ repeated b times

Step 2: Using knowledge of $b = |pad|$, decrypt m

- Change c_1 (and thus also m_2) by δ_i defined as

$$\begin{aligned}\delta_i \;\; = \;\; & \overbrace{0x00||\cdots||0x00}^{L-(b+1) \text{ Bytes}}||0x(i)||\overbrace{0x(b+1)||\cdots||0x(b+1)}^{b \text{ Bytes}} \oplus \\ & \underbrace{0x00||\cdots||0x00||0x00}_{L-b \text{ Bytes}}||\underbrace{0xb||\cdots||0xb}_{b \text{ Bytes}}\end{aligned}$$

- $m_2 \oplus \delta_i = \overbrace{m_2^1||\cdots||m_2^{L-(b+1)}}^{L-(b+1) \text{ Bytes}}||(0x(i) \oplus m_2^{L-b})||\overbrace{0x(b+1)||\cdots}^{b \text{ Bytes}}$
- Will only decrypt correctly if $0xi \oplus m_2^{L-b} = 0x(b+1)$
  - Trying all 256 values for i, $\mathcal{A}$ can learn m_2^{L-b} (a Byte of m)

Padding Oracle Attack

Observation: We know that m_2 ends in $(0xb)$ repeated b times

Step 2: Using knowledge of $b = |pad|$, decrypt m

- Change c_1 (and thus also m_2) by δ_i defined as

$$\delta_i = \overbrace{0x00||\cdots||0x00}^{L-(b+1)\text{ Bytes}}||0x(i)||\overbrace{0x(b+1)||\cdots||0x(b+1)}^{b\text{ Bytes}} \oplus \underbrace{0x00||\cdots||0x00||0x00}_{L-b\text{ Bytes}}||\underbrace{0xb||\cdots||0xb}_{b\text{ Bytes}}$$

- $m_2 \oplus \delta_i = \overbrace{m_2^1||\cdots||m_2^{L-(b+1)}}^{L-(b+1)\text{ Bytes}}||(0x(i) \oplus m_2^{L-b})||\overbrace{0x(b+1)||\cdots}^{b\text{ Bytes}}$
- Will only decrypt correctly if $0xi \oplus m_2^{L-b} = 0x(b+1)$
  - Trying all 256 values for i, $\mathcal{A}$ can learn m_2^{L-b} (a Byte of m)
- Repeat attack for all Bytes of m_2 by changing to appropriate padding

Padding Oracle Attack

Observation: We know that m_2 ends in $(0xb)$ repeated b times

Step 2: Using knowledge of $b = |pad|$, decrypt m

- Change c_1 (and thus also m_2) by δ_i defined as

$$\delta_i \quad = \quad \overbrace{0x00||\cdots||0x00}^{L-(b+1)\text{ Bytes}}||0x(i)||\overbrace{0x(b+1)||\cdots||0x(b+1)}^{b\text{ Bytes}} \oplus$$
$$\underbrace{0x00||\cdots||0x00||0x00}_{L-b\text{ Bytes}}||\underbrace{0xb||\cdots||0xb}_{b\text{ Bytes}}$$

- $m_2 \oplus \delta_i = \overbrace{m_2^1||\cdots||m_2^{L-(b+1)}}^{L-(b+1)\text{ Bytes}}||(0x(i) \oplus m_2^{L-b})||\overbrace{0x(b+1)||\cdots}^{b\text{ Bytes}}$
- Will only decrypt correctly if $0xi \oplus m_2^{L-b} = 0x(b+1)$
  - Trying all 256 values for i, $\mathcal{A}$ can learn m_2^{L-b} (a Byte of m)
- Repeat attack for all Bytes of m_2 by changing to appropriate padding
- Can mount similar attack to decrypt m_1

Padding Oracle Attack

Attack outline:

1. Learn size of padding using decryption errors
2. Using knowledge of $|pad|$, decrypt m Byte-by-Byte

Padding Oracle Attack

Attack outline:

1. Learn size of padding using decryption errors
2. Using knowledge of $|pad|$, decrypt m Byte-by-Byte

Observations:

- Can view error as partial decryption
- Even very limited decryption oracle can lead to an attack

Padding Oracle Attack

Attack outline:

1. Learn size of padding using decryption errors
2. Using knowledge of $|pad|$, decrypt m Byte-by-Byte

Observations:

- Can view error as partial decryption
- Even very limited decryption oracle can lead to an attack

Warning

Be very careful with error messages in crypto constructions