Cryptography

Lecture 10

Arkady Yerukhimovich

September 30, 2024

Outline

Lecture 8 Review

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

Outline

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

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

Is this CCA Secure?

Is PRF+OTP CCA Secure?

PRF+OTP Encryption

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

- $\text{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\text{Enc}(k, m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- $\text{Dec}(k, c)$: 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 $\text{Dec}_k(\overline{c})$

Is PRF+OTP CCA Secure?

PRF+OTP Encryption

- $\text{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\text{Enc}(k, m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- $\text{Dec}(k, c)$: 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 $\text{Dec}_k(\overline{c})$
- $\text{Dec}_k(\overline{c})$ returns $\overline{m} = F_k(r^*) \oplus 0^n = F_k(r^*)$

Is PRF+OTP CCA Secure?

PRF+OTP Encryption

- $\text{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\text{Enc}(k, m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- $\text{Dec}(k, c)$: 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 $\text{Dec}_k(\overline{c})$
- $\text{Dec}_k(\overline{c})$ 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

- $\text{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\text{Enc}(k, m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- $\text{Dec}(k, c)$: 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 $\text{Dec}_k(\overline{c})$
- $\text{Dec}_k(\overline{c})$ 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 $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

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 $k \leftarrow \text{Gen}(1^n)$

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 $|m_0| = |m_1|$.

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 $\mathcal{A}$

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 $\text{Dec}_k(c)$)

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 $b' = b$.

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:

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

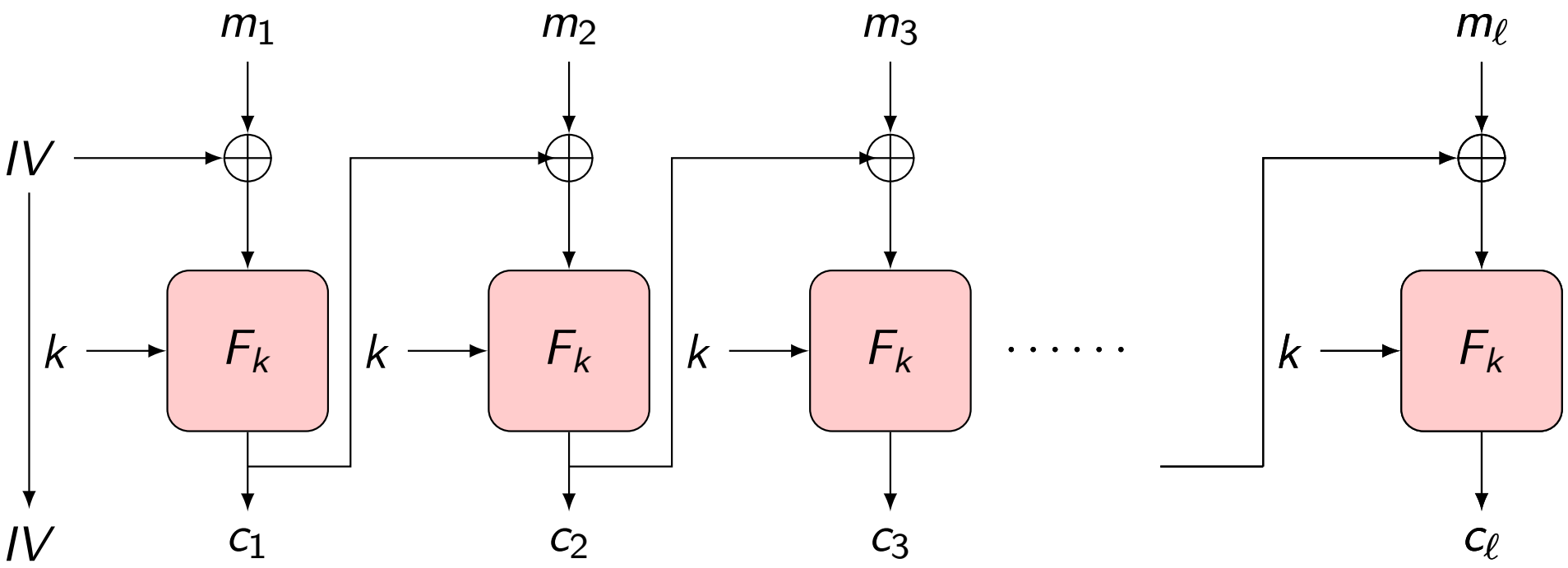
- The challenger chooses $k \leftarrow \text{Gen}(1^n)$
- $\mathcal{A}^{\text{Enc}_k(\cdot),\text{Dec}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \text{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\text{Enc}_k(\cdot),\text{Dec}_k(\cdot)}$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

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

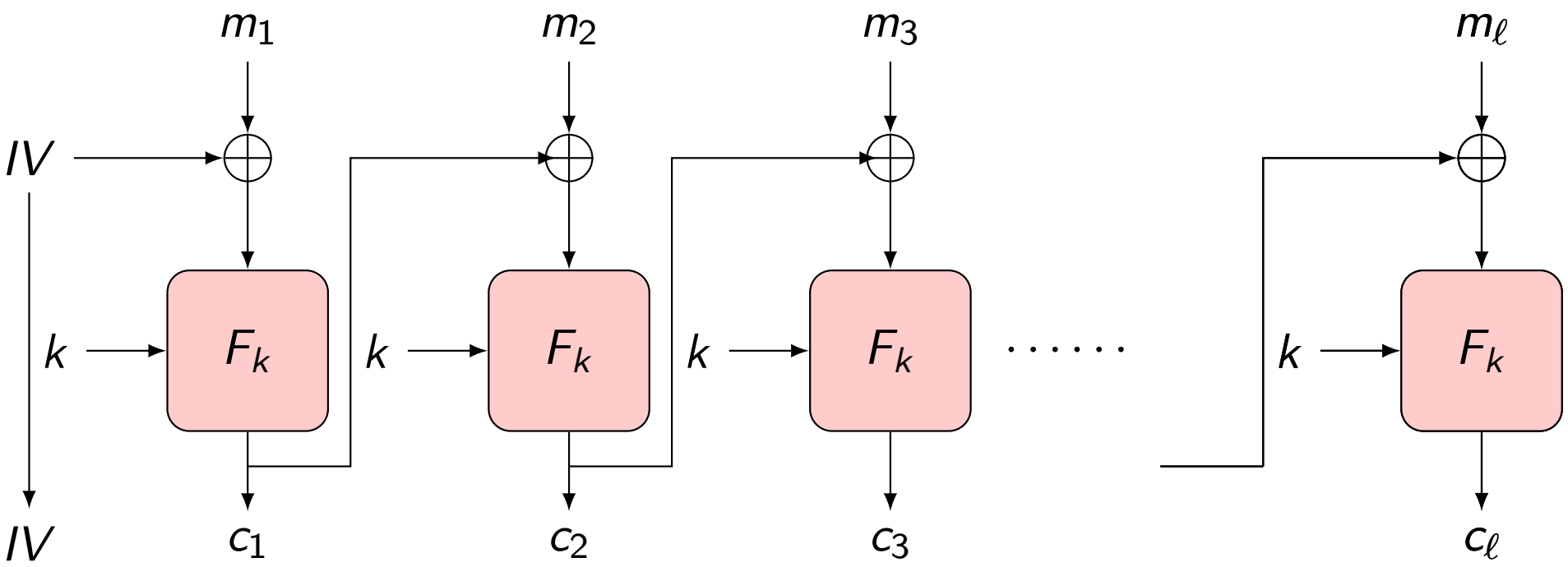
$$\Pr[\text{PrivK}^{cca}_{\mathcal{A},\Pi}(n) = 1] \leq 1/2 + \text{negl}(n)$$

Outline

Padding Oracle Attack on CBC Mode

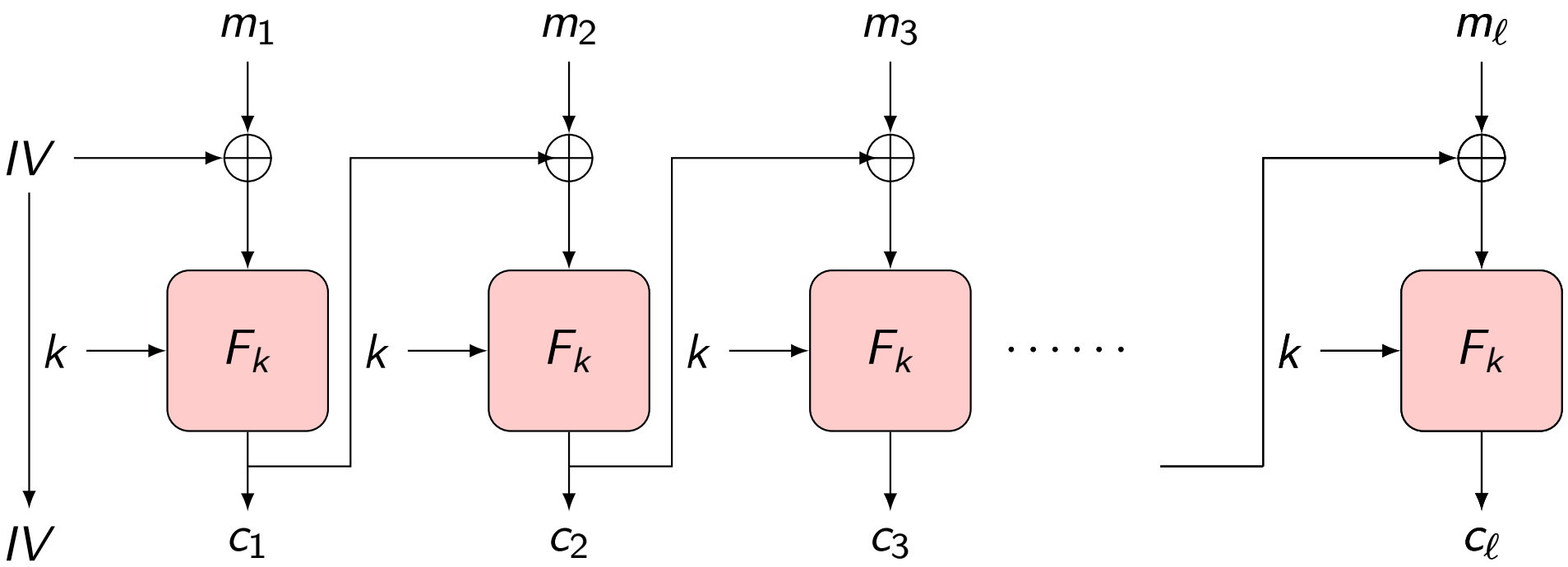


Padding Oracle Attack on CBC Mode



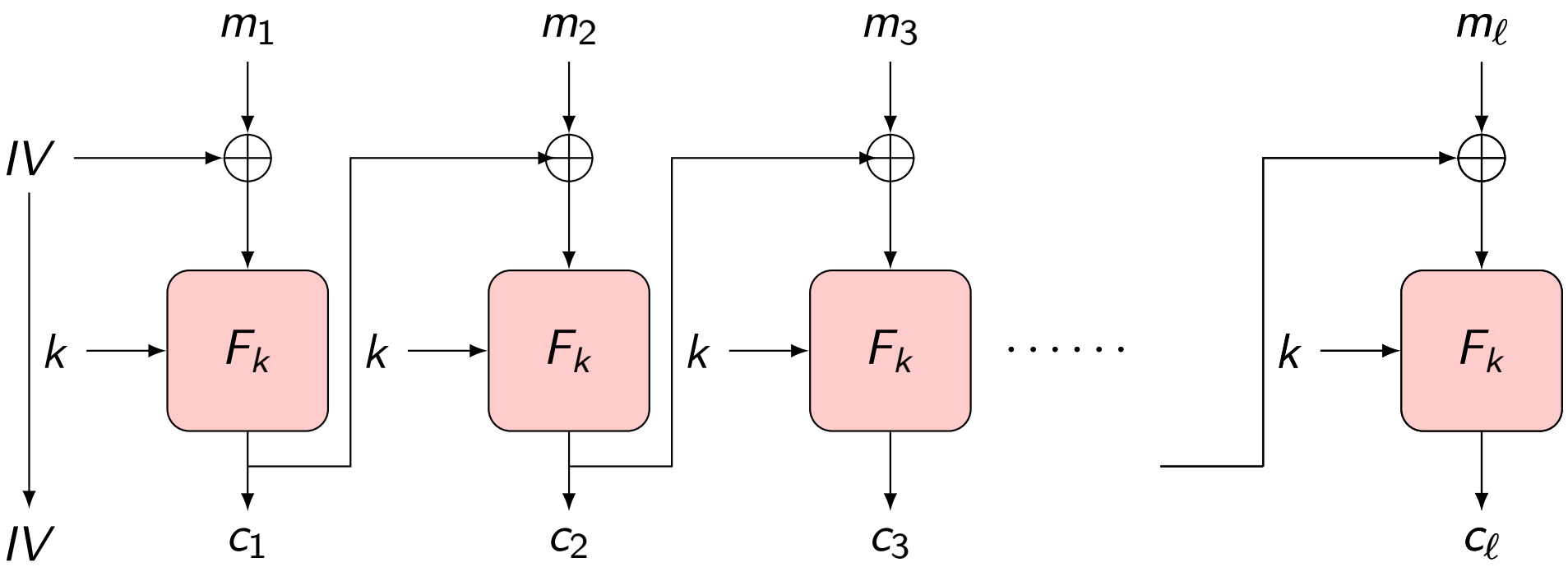
- This assumes that $|m|$ is a multiple of block-length L.

Padding Oracle Attack on CBC Mode



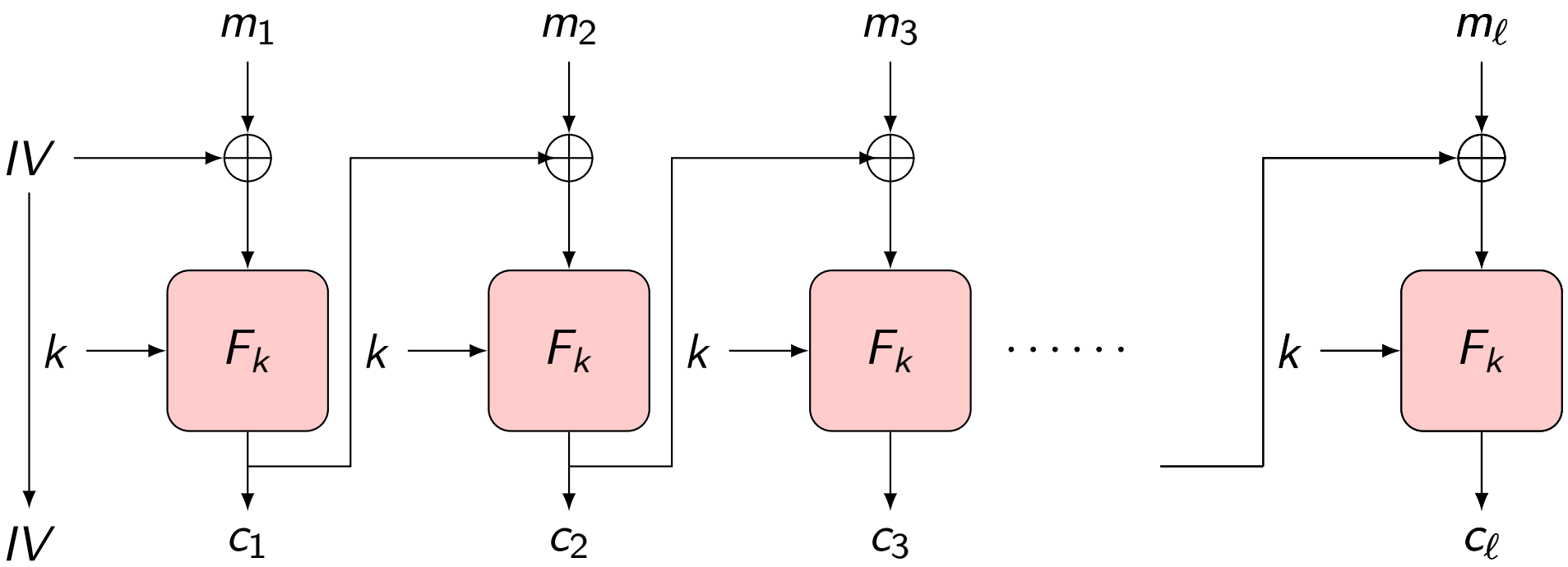
- This assumes that $|m|$ is a multiple of block-length L.
- If it is not, standard approach is to pad m to a multiple of L
  - Need to be able to tell what is part of m and what is padding

Padding Oracle Attack on CBC Mode



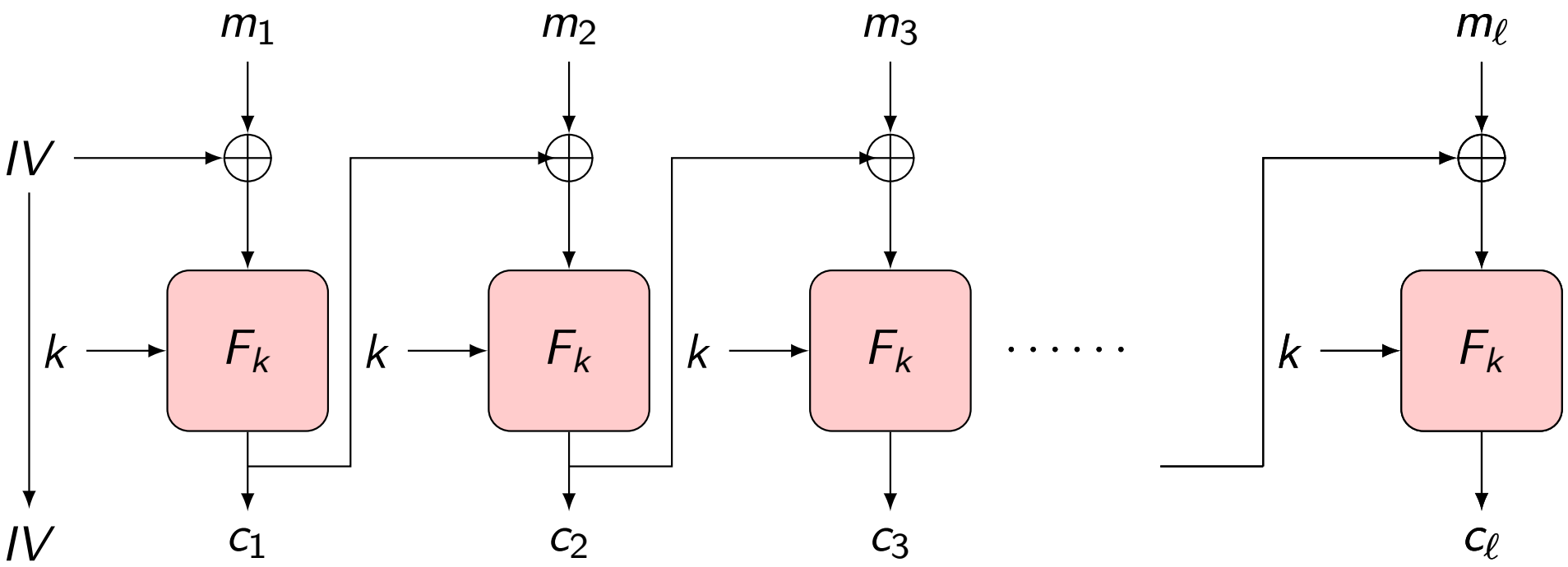
- This assumes that $|m|$ is a multiple of block-length L.
- If it is not, standard approach is to pad m to a multiple of L
  - Need to be able to tell what is part of m and what is padding
  - Add 1 to L bytes to end of m to pad to next multiple of L.

Padding Oracle Attack on CBC Mode



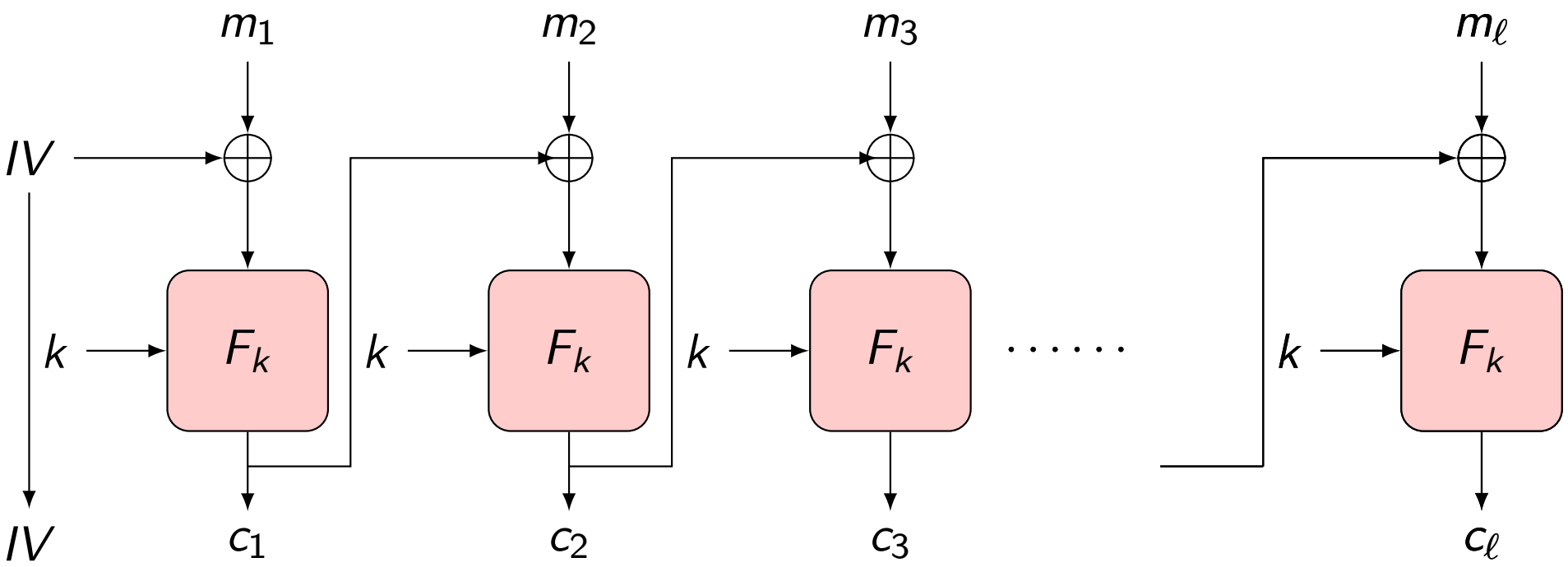
- This assumes that $|m|$ is a multiple of block-length L.
- If it is not, standard approach is to pad m to a multiple of L
  - Need to be able to tell what is part of m and what is padding
  - Add 1 to L bytes to end of m to pad to next multiple of L.
  - To identify padding, pad value indicates number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

Padding Oracle Attack on CBC Mode



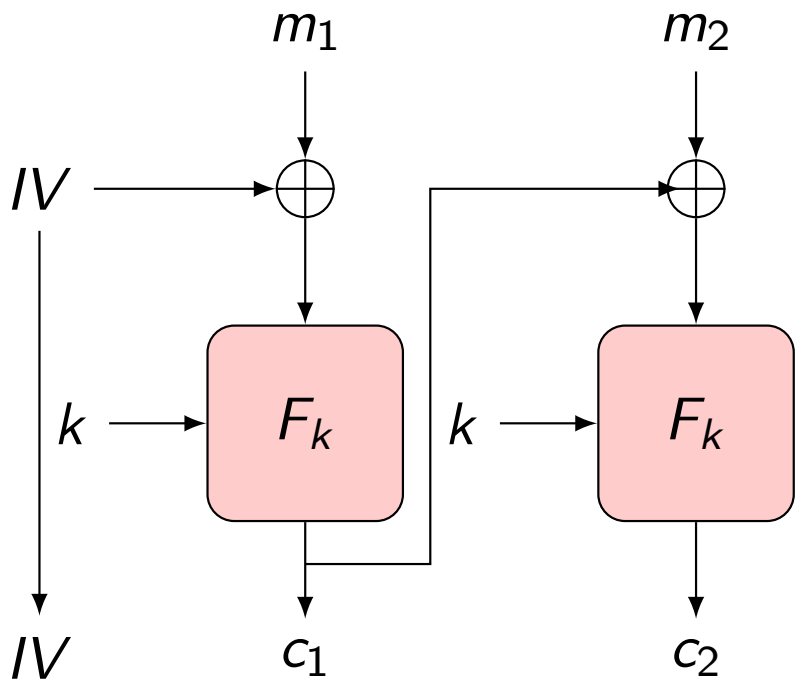
- This assumes that $|m|$ is a multiple of block-length L.
- If it is not, standard approach is to pad m to a multiple of L
  - Need to be able to tell what is part of m and what is padding
  - Add 1 to L bytes to end of m to pad to next multiple of L.
  - To identify padding, pad value indicates number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



- Consider encryption of a 2-block message m

Quiz

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