Chủ đề 5:
Modes of Operation và Padding Scheme

PGS.TS. Trần Minh Triết

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Nội dung

- **Các kiểu thao tác (Modes of Operation)**
- **Các kiểu chèn bổ sung thông tin (Padding Scheme)**

Các kiểu thao tác (Modes of Operation)

- Trong mã hóa, thường dữ liệu được chia thành từng đoạn (block) có kích thước cố định (ví dụ như 64 hay 128 bit).
- Để mã hóa các thông điệp dài (có thể chia thành nhiều block), có thể sử dụng các kiểu thao tác khác nhau (**modes of operation**) khác nhau

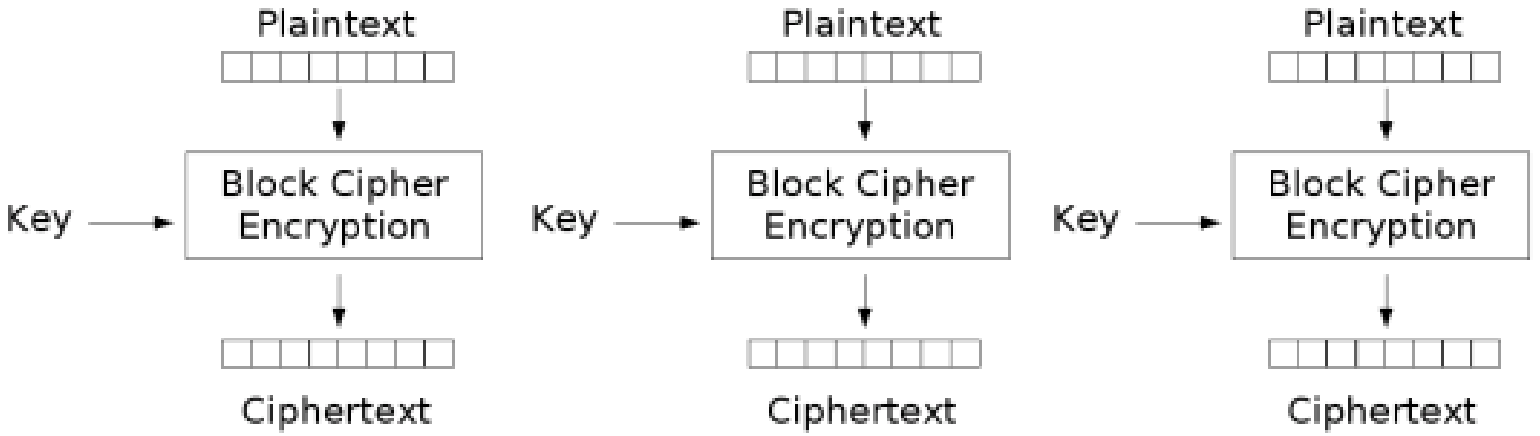
Các kiểu thao tác (Modes of Operation)

- Các kiểu thao tác đầu tiên được đề nghị (ECB, CBC, OFB, CFB) đảm bảo tính bí mật (**confidentiality**), không giúp đảm bảo tính toàn vẹn thông tin (**message integrity**)
- Các kiểu thao tác được thiết kế cho phép (**CCM**, **EAX** và **OCB**) vừa đảm bảo tính bí mật, vừa đảm bảo xác định tính toàn vẹn thông tin.
- Một số kiểu thao tác được xây dựng để mã hóa sector trên đĩa:
  - Tweakable narrow-block encryption –LRW
  - Wide-block encryption -CMC và EME

Electronic codebook (ECB)

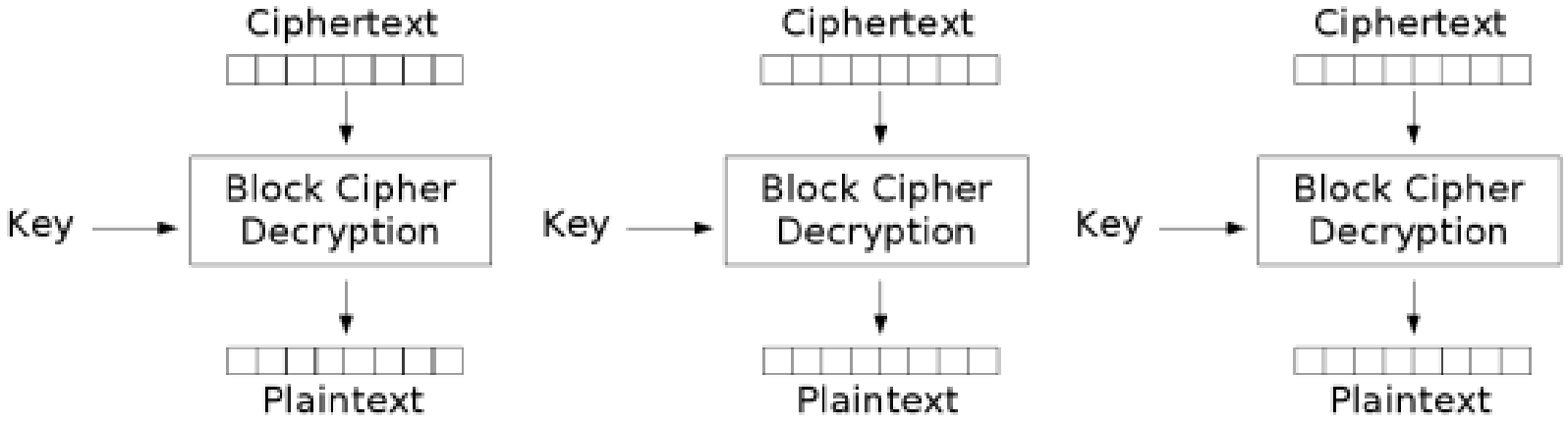
- Kiểu mã hóa đơn giản nhất là electronic codebook (ECB)
- Thông điệp cần mã hóa được chia thành từng đoạn, mỗi đoạn được mã hóa độc lập nhau.
- Hạn chế: các khối có cùng nội dung, sau khi mã hoá xong cũng tạo thành các khối kết quả giống hệt nhau ➔ Không che giấu được các “mẫu” dữ liệu (data pattern).
- Không khuyến khích sử dụng ECB trong các giao thức mã hóa

Electronic codebook (ECB)



Electronic Codebook (ECB) mode encryption

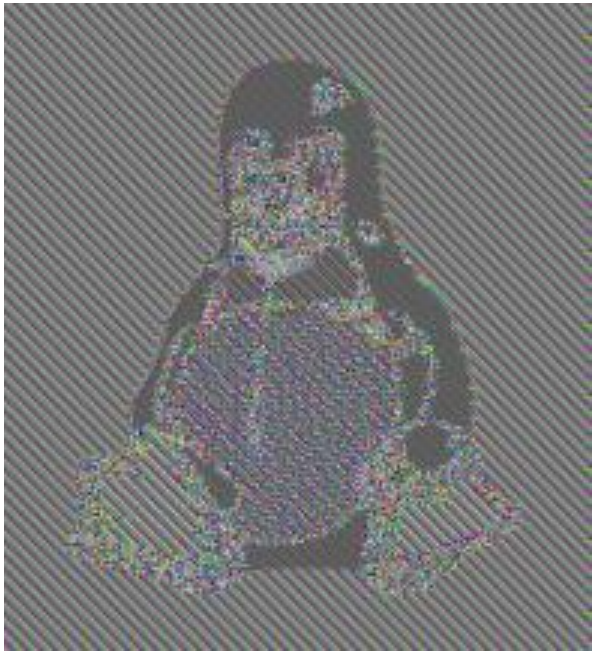
Electronic codebook (ECB)



Electronic Codebook (ECB) mode decryption

Electronic codebook (ECB)

Ảnh gốc



Mã hóa theo kiểu ECB

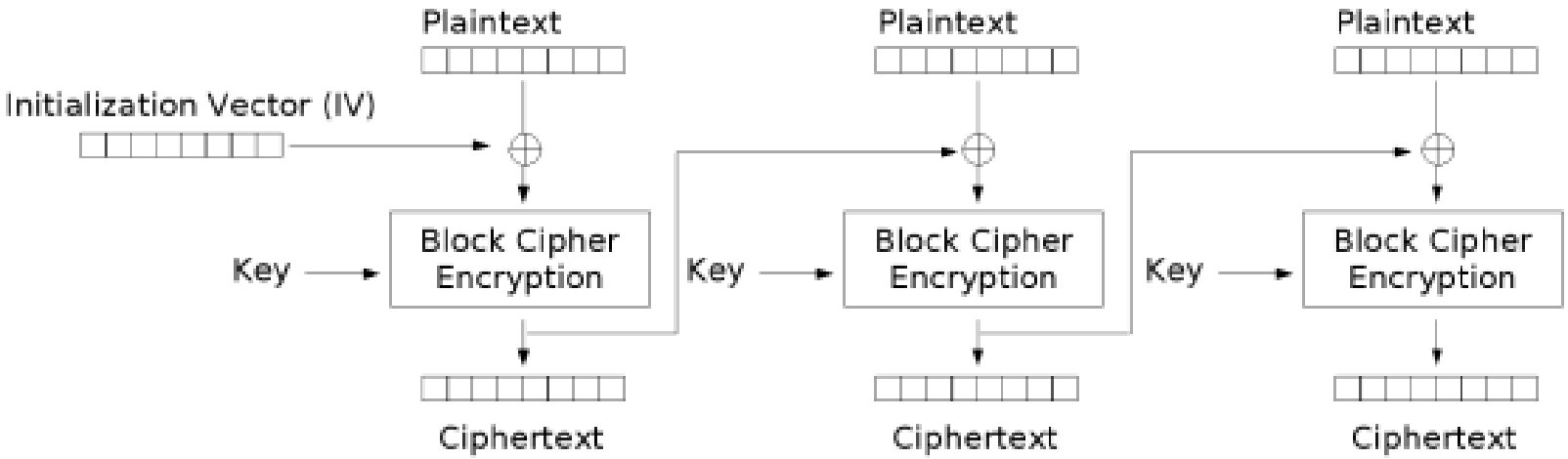
Mã hóa theo các kiểu khác

ECB có thể làm cho giao thức kém an toàn để bảo vệ tính toàn vẹn thông tin (ví dụ như đối với kiểu tấn công replay attacks)

Cipher-block chaining (CBC)

- Trong kiểu mã hóa **cipher-block chaining** (CBC):
  - Mỗi khối plaintext được XOR với khối ciphertext trước khi được mã hóa.
  - Như vậy, mỗi khối ciphertext phụ thuộc vào tất cả các khối plaintext xuất hiện từ đầu đến thời điểm đó
  - Để đảm bảo tính duy nhất của mỗi thông điệp được mã hóa, ta sử dụng thêm vector khởi tạo (initialization vector)

Cipher-block chaining (CBC)

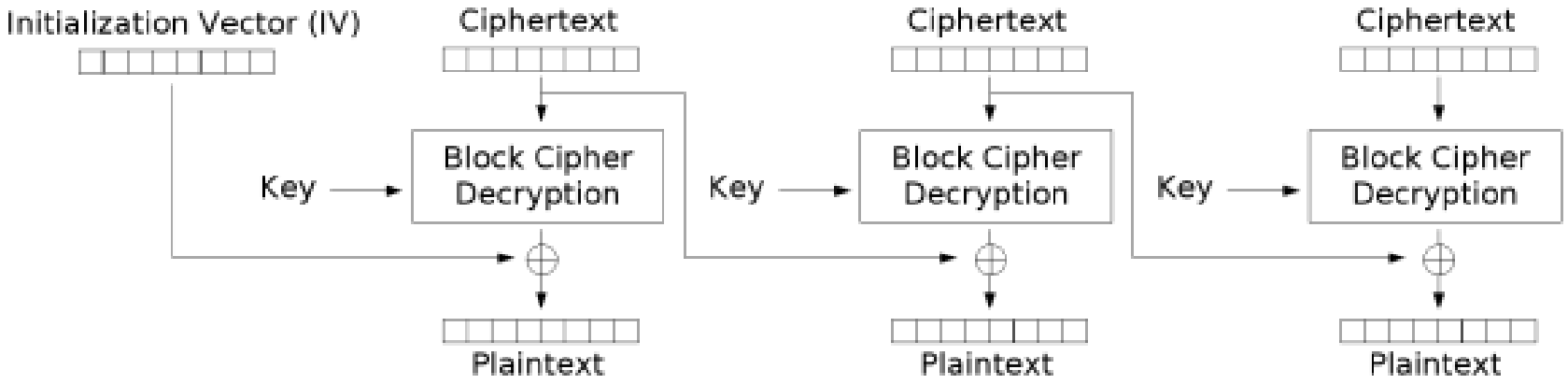


Cipher Block Chaining (CBC) mode encryption

$$C_0 = \mathbf{IV}$$

$$C_i = E_K(P_i \oplus C_{i-1})$$

Cipher-block chaining (CBC)



Cipher Block Chaining (CBC) mode decryption

$$C_0 = \mathbf{IV}$$

$$P_i = D_K(C_i) \oplus C_{i-1}$$

Cipher-block chaining (CBC)

- CBC là kiểu mã hóa thường được sử dụng nhất
- Hạn chế: xử lý tuần tự, không thể song song hóa
  - có thể chọn giải pháp counter mode để xử lý song song

Propagating cipher-block chaining (PCBC)

- Kiểu mã hóa **propagating cipher-block chaining** được thiết kế cho phép sự ảnh hưởng lan truyền nhiều hơn trong kiểu CBC.

$$P_0 = \text{IV}, C_0 = 0, C_i = E_K (P_i \oplus P_{i-1} \oplus C_{i-1})$$

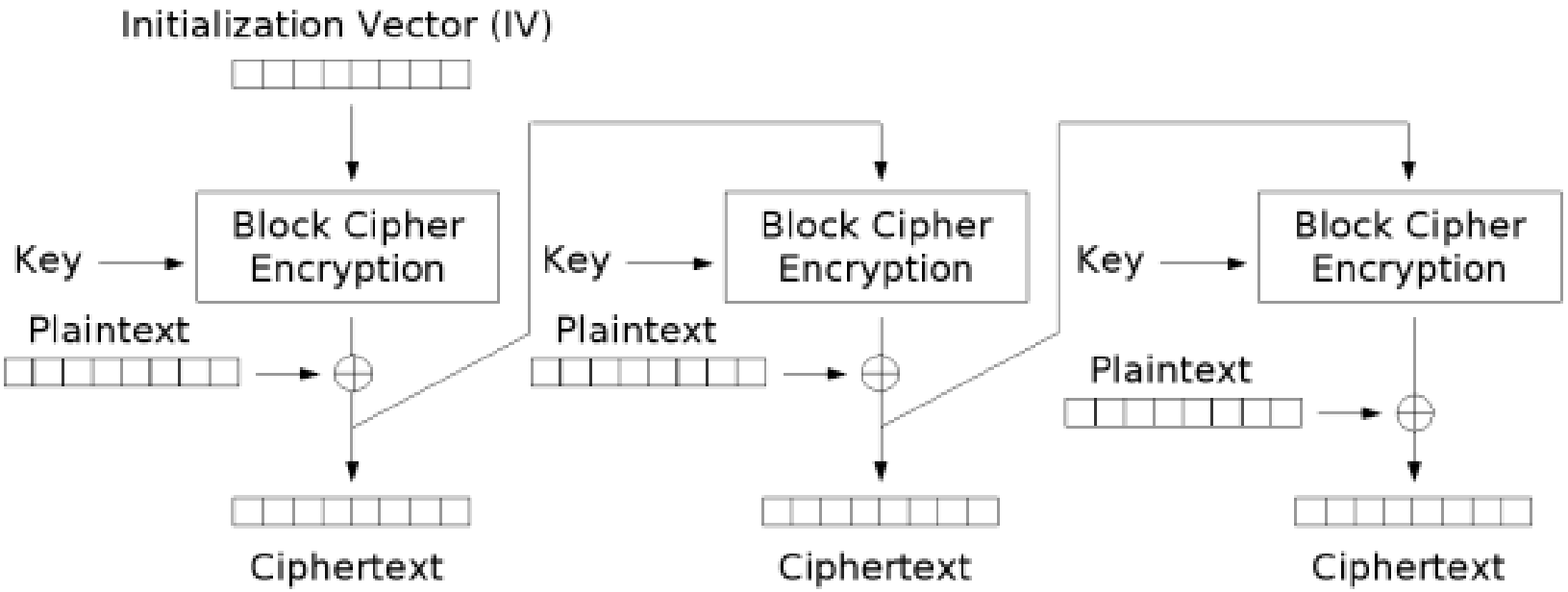
$$P_0 = \text{IV}, C_0 = 0, P_i = D_K (C_i) \oplus P_{i-1} \oplus C_{i-1}$$

- PCBC thường được dùng chủ yếu trong Kerberos và WASTE (ngoài ra thì ít thông dụng !)

Cipher feedback (CFB)

- Bản chất:
  - Plaintext KHÔNG được mã hóa bằng chính thuật toán đang xét
  - Plaintext được mã hóa bằng cách XOR với một chuỗi được tạo ra bằng thuật toán mã hóa.
  - Biến Block Cipher thành stream cipher

Cipher feedback (CFB)

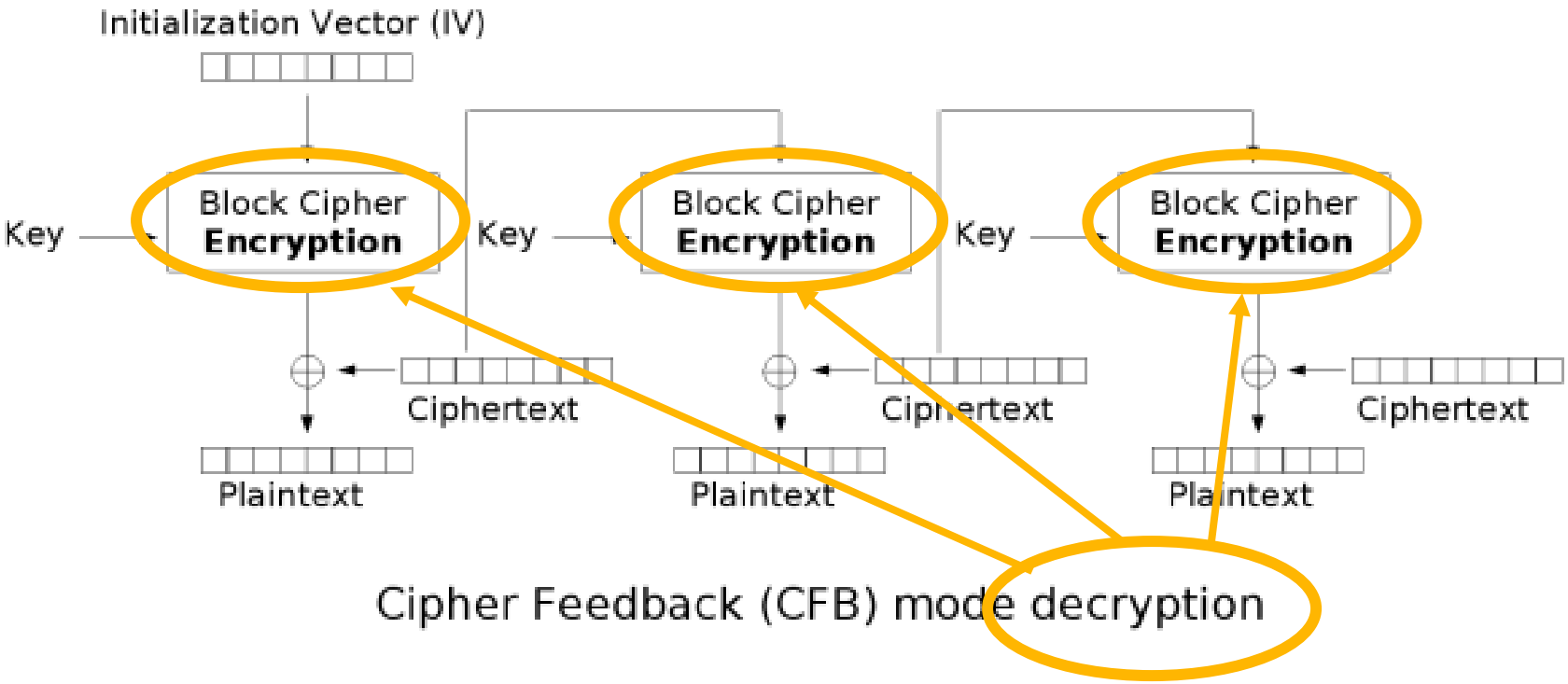


Cipher Feedback (CFB) mode encryption

$$C_0 = \mathbf{IV}$$

$$C_i = P_i \oplus E_K(C_{i-1})$$

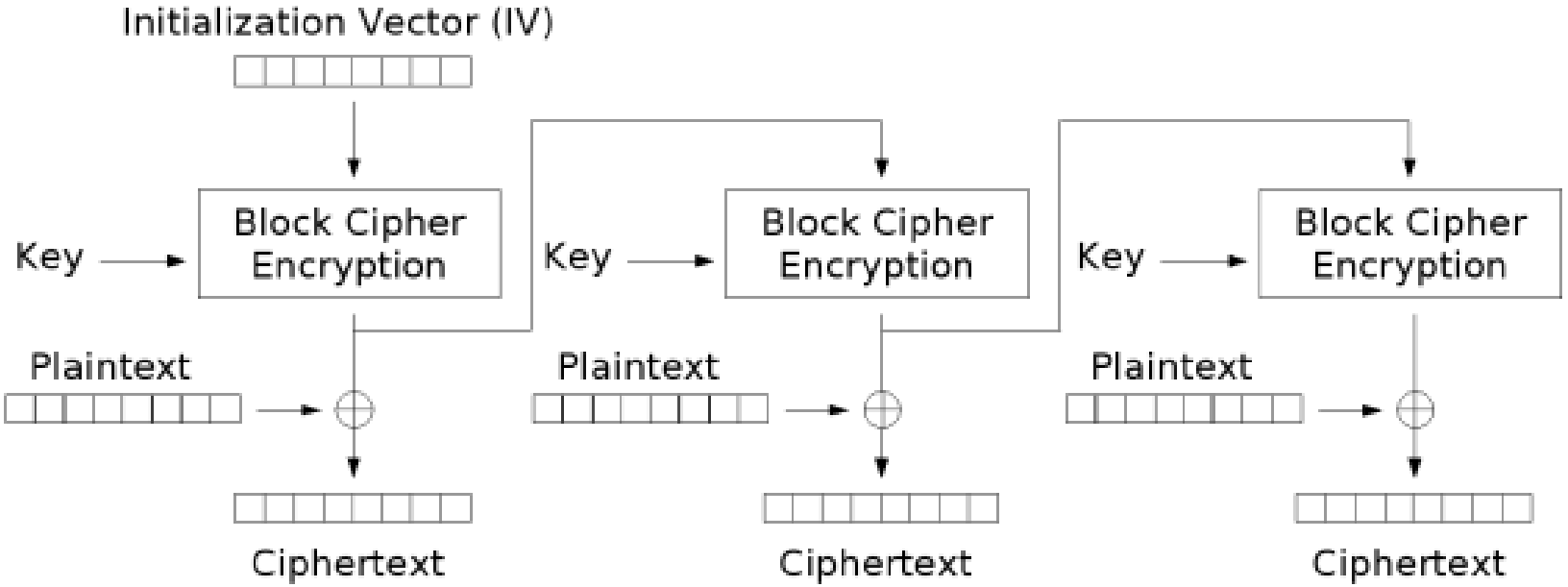
Cipher feedback (CFB)



Output feedback (OFB)

- Bản chất:
  - Plaintext KHÔNG được mã hóa bằng chính thuật toán đang xét
  - Plaintext được mã hóa bằng cách XOR với một chuỗi được tạo ra bằng thuật toán mã hóa.
  - Biến Block Cipher thành stream cipher

Output feedback (OFB)



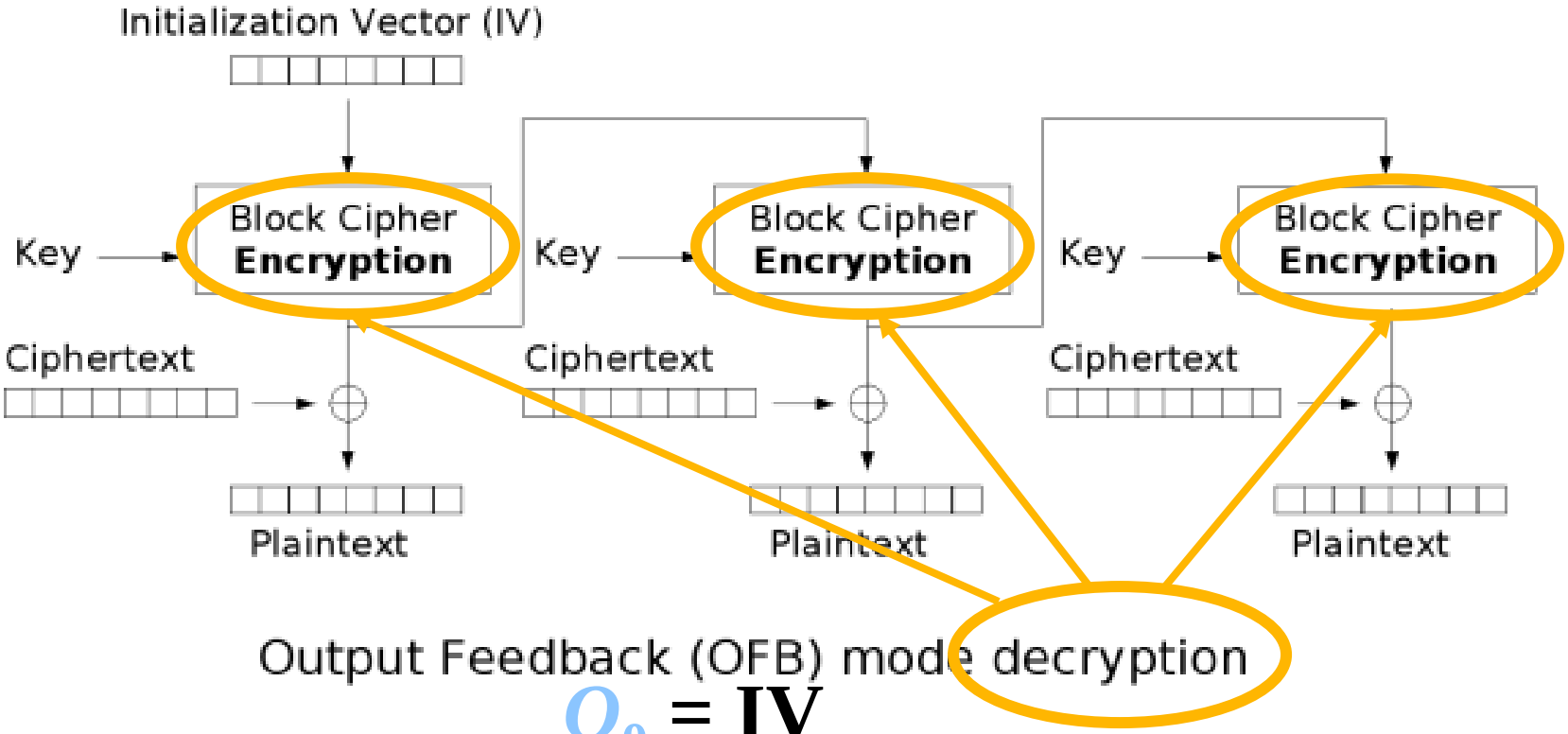
Output Feedback (OFB) mode encryption

$$O_0 = \mathbf{IV}$$

$$O_i = E_K(O_{i-1})$$

$$C_i = P_i \oplus O_i$$

Output feedback (OFB)



$$O_0 = \mathbf{IV}$$

$$O_i = E_K(O_{i-1})$$

$$P_i = C_i \oplus O_i$$

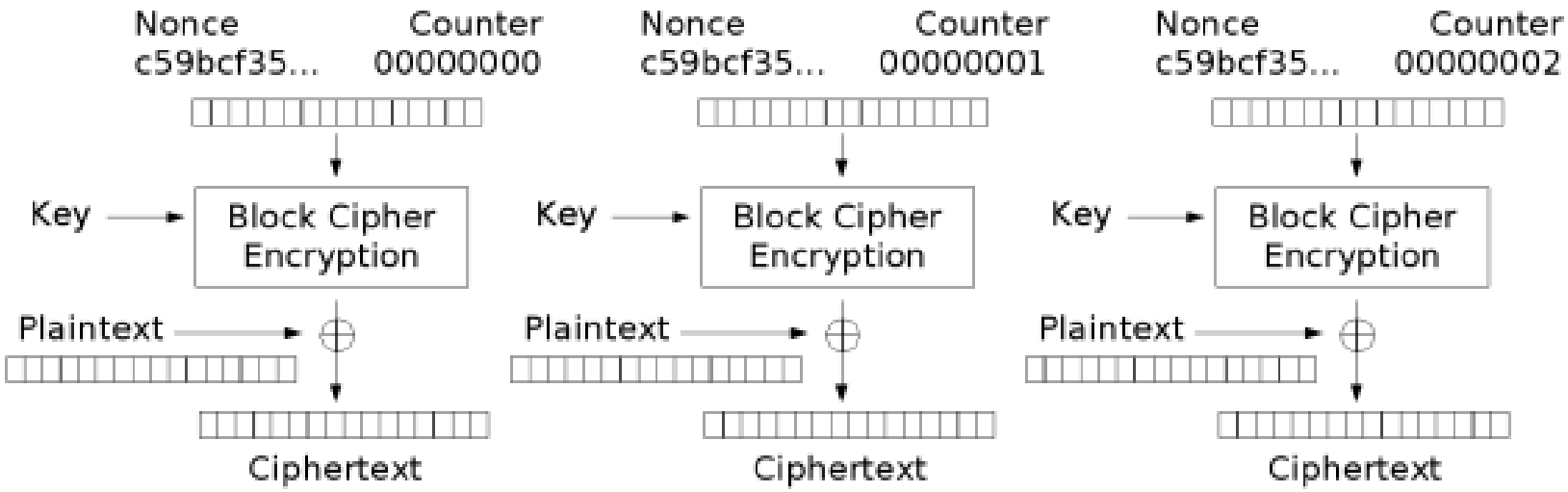
Counter (CTR)

- Kiểu CTR còn gọi là Segmented Integer Counter (SIC)
- Tương tự OFB, kiểu Counter cũng biến block cipher thành stream cipher.
  - Tạo ra block keystream tiếp theo bằng cách mã hóa giá trị kế tiếp của "counter".
- Counter có thể là bất kỳ hàm nào sinh ra dãy số không có giá trị lặp lại sau một khoảng thời gian đủ lâu

Counter (CTR)

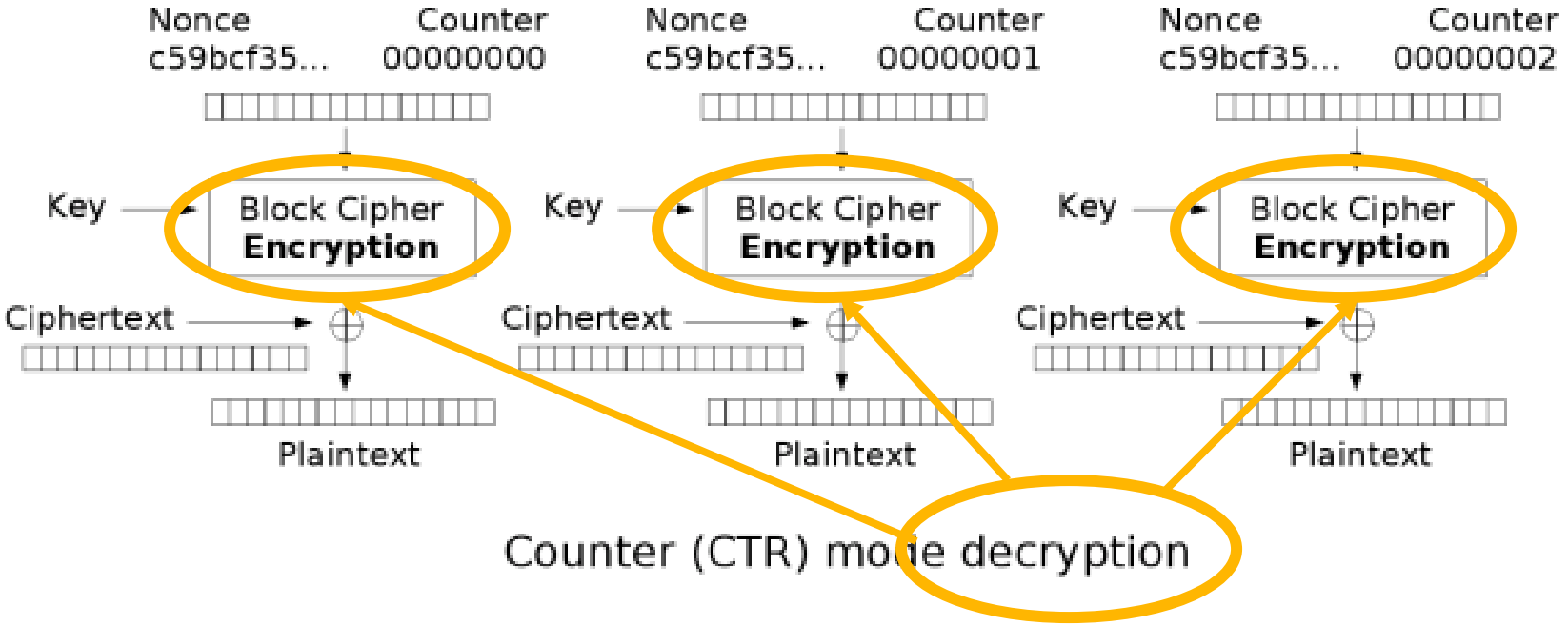
- CTR có tính chất giống OFC,
- CTR cho phép giải mã “ngẫu nhiên” bất kỳ khối cipherytext nào
- Lưu ý: vai trò của đoạn dữ liệu nonce giống như initialization vector (IV)
- IV/nonce và giá trị counter có thể được nối với nhau, cộng hay XOR để tạo thành 1 dãy bit đặc trưng duy nhất ứng với mỗi giá trị counter cụ thể

Counter (CTR)



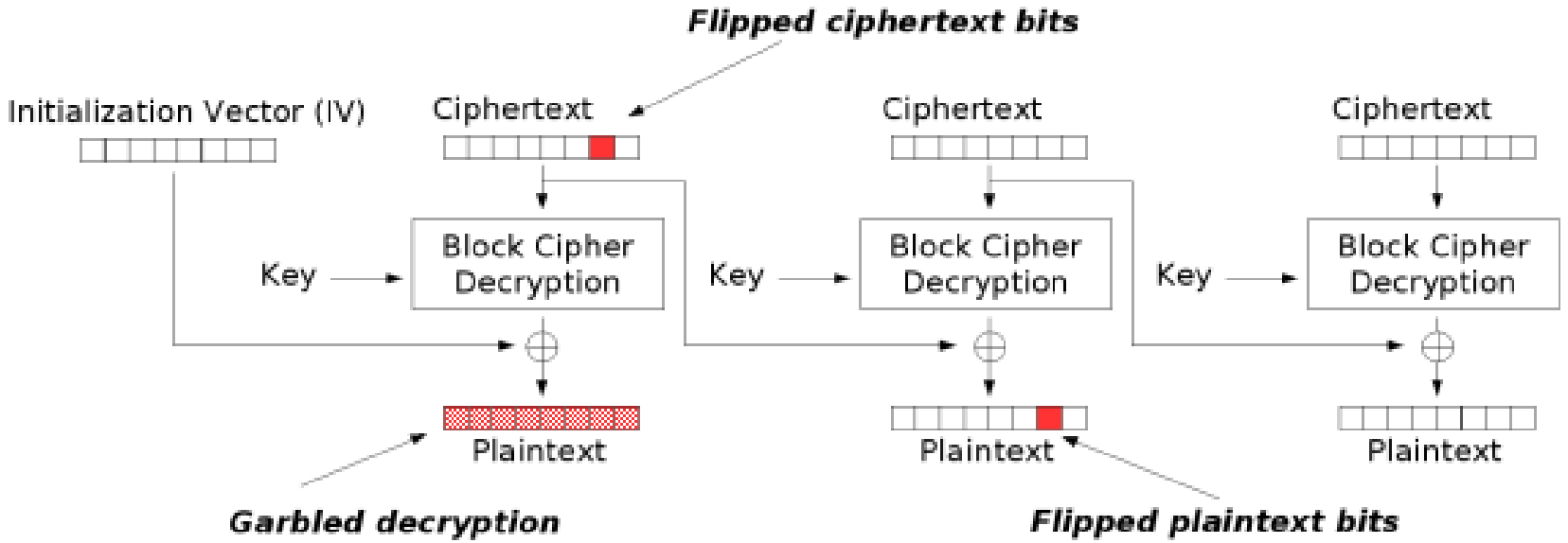
Counter (CTR) mode encryption

Counter (CTR)


Nonce
c59bcf35...
Counter
00000000
Nonce
c59bcf35...
Counter
00000001
Nonce
c59bcf35...
Counter
00000002
Key
Block Cipher
Encryption
Key
Block Cipher
Encryption
Key
Block Cipher
Encryption
Ciphertext
Ciphertext
Ciphertext
Plaintext
Plaintext
Plaintext
Counter (CTR) mode decryption

Sự lan truyền lỗi

- Hạn chế sự lan truyền lỗi: 1 tiêu chí để đánh giá kiểu mã hóa
- Ví dụ: Khảo sát sự lan truyền lỗi khi giải mã thông tin trong CBC



Initialization vector (IV)

- Tất cả các kiểu mã hóa (ngoại trừ ECB) đều sử dụng vector khởi tạo (*initialization vector - IV).*
- Tác dụng của IV:
  - Dummy block để việc xử lý khối đầu tiên không khác biệt so với việc xử lý các khối tiếp thao
  - Tăng tính ngẫu nhiên của quy trình mã hóa.
- IV:
  - Không cần giữ bí mật
  - Cần đảm bảo là hạn chế việc sử dụng lại cùng giá trị IV với cùng 1 khóa.

Initialization vector (IV)

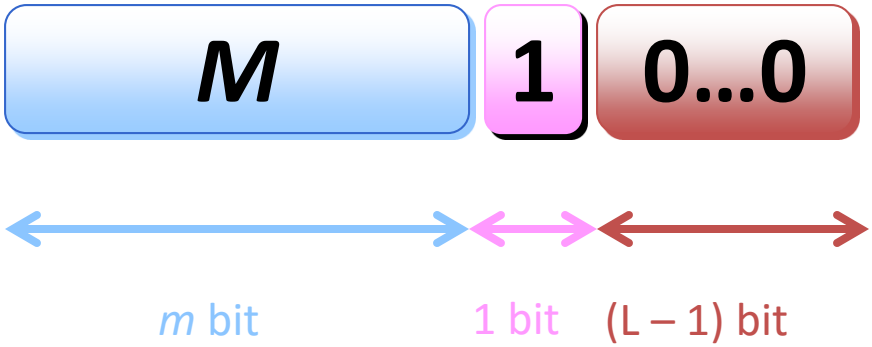
- Với CBC và CFB, sử dụng lại giá trị IV làm rò rỉ thông tin.
- Với OFB và CTR, sử dụng lại IV làm phá vỡ hoàn toàn tính an toàn của hệ thống
- IV trong CFB phải được phát sinh ngẫu nhiên và giữ bí mật cho đến khi nội dung của khối plaintext đầu tiên được sẵn sàng để mã hóa

Các kiểu chèn bổ sung thông tin

- **Padding Scheme:** bổ sung thông tin để khối dữ liệu có kích thước phù hợp cho việc mã hóa
- Yêu cầu:
  - Khối dữ liệu sau khi bổ sung có kích thước phù hợp với việc mã hóa
  - Có thể dễ dàng khôi phục chính xác dữ liệu sau khi giải mã (cắt bỏ chính xác các dữ liệu bổ sung thêm vào)
- Các phương pháp cơ bản:
  - **Bit Padding: xem RFC1321**
    - http://www.faqs.org/rfcs/rfc1321.html
  - **Byte Padding: xem RFC1319**
    - http://www.faqs.org/rfcs/rfc1319.html

Các kiểu chèn bổ sung thông tin

- **Bit Padding:**
  - Kích thước khối dữ liệu "chuẩn": *n* bit
  - Khối dữ liệu gốc *M* có kích thước *m* bit ($m \leq n$)
  - Khối dữ liệu sau khi padding

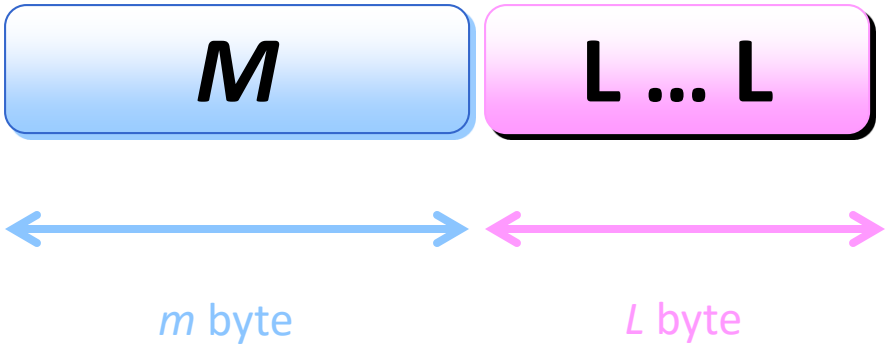


$L = n - (m \bmod n)$

Điều gì xảy ra nếu $m = n$?

Các kiểu chèn bổ sung thông tin

- **Byte Padding (PKCS5):**
  - Kích thước khối dữ liệu "chuẩn": n byte ($n < 256$)
  - Khối dữ liệu gốc M có kích thước m byte ($m \leq n$)
  - Khối dữ liệu sau khi padding



$L = n - (m \bmod n)$

Điều gì xảy ra nếu $m = n$?

Tìm hiểu thêm

- **OAEP**
- **CCM**
- **EAX**
- **OCB**