# ECE/COMPSCI 356 Computer Network Architecture

# Lecture 25: SSL/TLS – Symmetric Crypto

Neil Gong

neil.gong@duke.edu

Slides credit: Dan Boneh, Doug Tygar, Dawn Song, David Wagner, Wenliang Du

# SSL/TLS

- SSL: Secure Sockets Layer
  - deprecated
- TLS: Transport Layer Security
  - builds on SSL
- HTTPS rely on SSL/TLS
- Based on cryptography

# Overview

- Cryptography: secure communication over insecure communication channels
- Three goals
  - Confidentiality
  - Integrity
  - Authenticity

# Brief History of Crypto

- 2,000 years ago
  - Caesar Cypher: shifting each letter forward by a fixed amount
  - Encode and decode by hand
- During World War I/II
  - Mechanical era: a mechanical device for encrypting messages
- After World War II
  - Modern cryptography: rely on mathematics and electronic computers
- Post-quantum crypto

# Modern Cryptography

- Symmetric-key cryptography
  - The same secret key is used by both endpoints of a communication
- Public-key cryptography
  - Two endpoints use different keys

# Roadmap

Symmetric-key cryptography

- –Confidentiality
  - •Block cipher
  - •Stream cipher
- –Integrity & authenticity
  - •HMAC

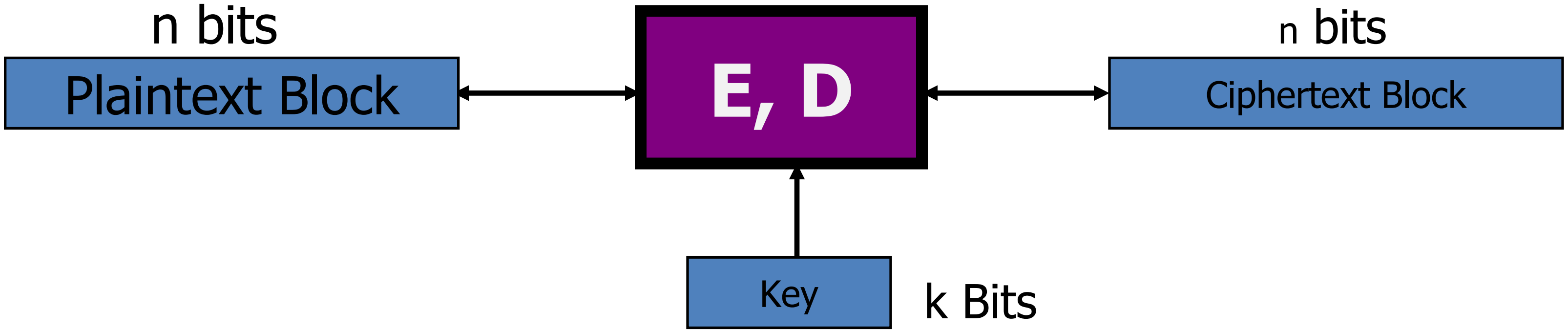
# Plaintext and ciphertext

- Plaintext is the message before encryption

- Ciphertext is the encrypted message

# Block Cipher

- Encrypt/Decrypt messages in fixed size blocks using the same secret key
  - k-bit secret key
  - n-bit plaintext/ciphertext



# Examples of Block Cipher

- DES - Data Encryption Standard (1977)
  - Works on 64 bit block with 56 bit keys
  - Developed by IBM (Lucifer) improved by NSA
  - Brute force attack feasible in 1997
- AES – Advanced Encryption Standard (2001)
  - Block size 128 bits
  - Key can be 128, 192, or 256 bits

# Stream cipher

- For a message with a length k
- Generate a key with length k
  - This is often a pseudo-random bit stream generated from a master secret key
- Encryption and decryption are simple
- Examples
  - One-time pad
  - RC4 (insecure)
  - ChaCha20-Poly1305

# One-time Pad

- K: random n-bit key
- P: n-bit message (plaintext)
- C: n-bit ciphertext
- Encryption: C = P xor K
- Decryption: P = C xor K
- A key can only be used once
- Impractical!

# Modes of Operation or Encryption Modes

- Block ciphers encrypt fixed size blocks
  - eg. DES encrypts 64-bit blocks with 56-bit key
- Need to en/decrypt arbitrary amounts of data
- Cover a wide variety of applications
- Can be used with any block cipher

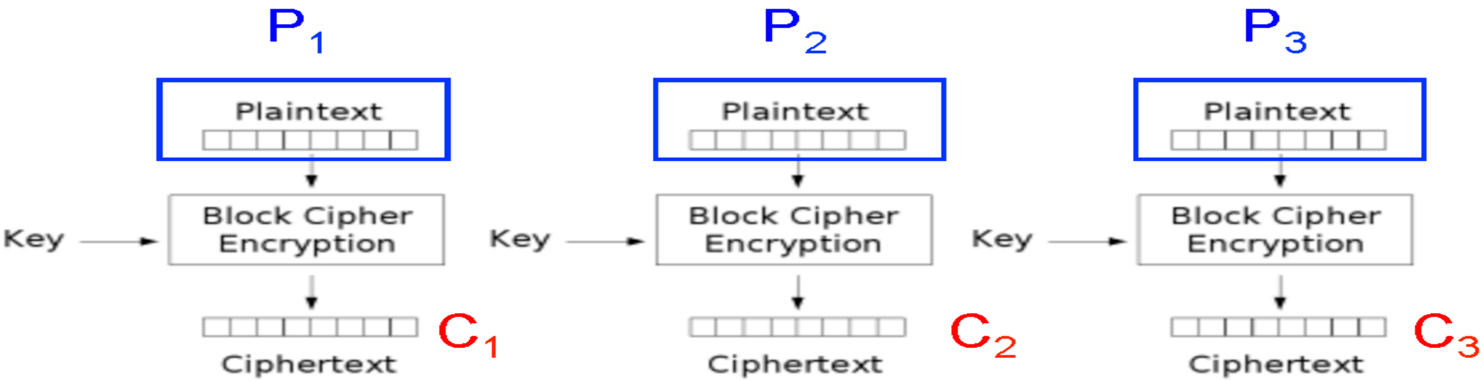
# Modes of Operation or Encryption Modes

- Examples include:
  - Electronic Codebook (ECB)
  - Cipher Block Chaining (CBC)
  - Cipher Feedback (CFB)
  - Output Feedback (OFB)
  - Counter (CTR)

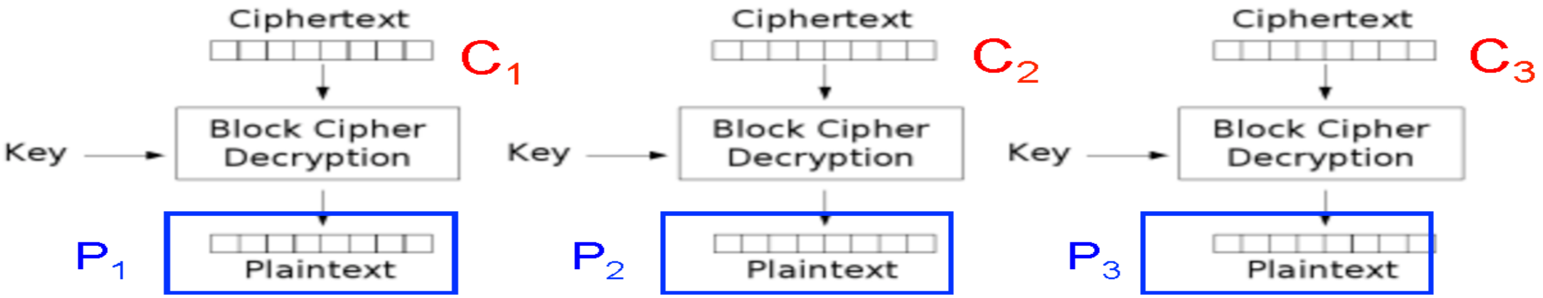
# Electronic Code Book (ECB)

- Message is broken into independent blocks
- Each block is encoded independently of the other blocks

$$C_i = EK(Pi)$$




Electronic Codebook (ECB) mode encryption




Electronic Codebook (ECB) mode decryption

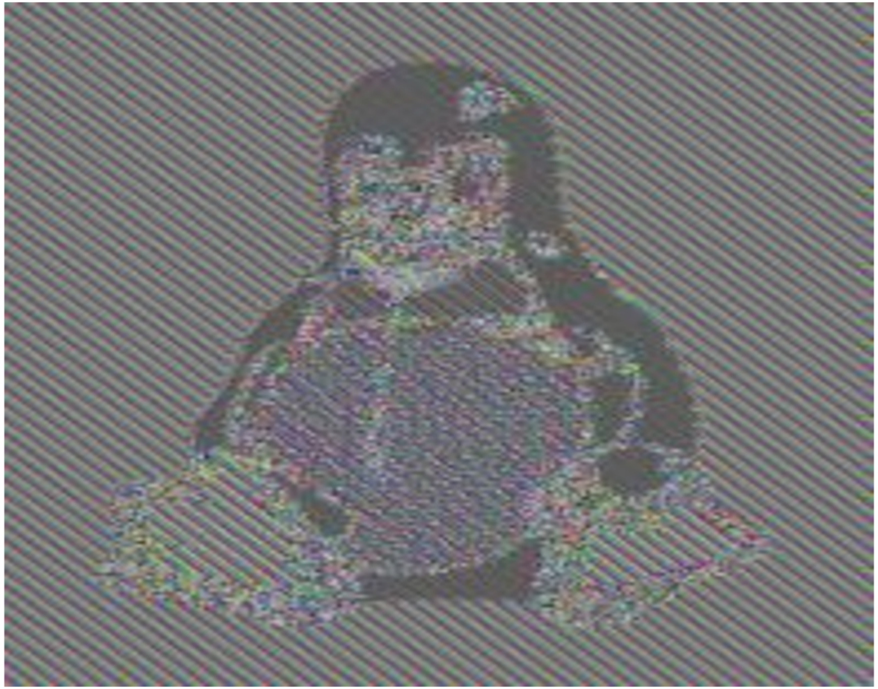
# Padding

- No guarantee that the size of the last block matches the cipher’s block size.
- Last block of the plaintext needs **padding** i.e. before encryption, extra data needs to be added to the last block of the plaintext, so its size equals to the cipher’s block size.
- Padding schemes need to clearly mark where the padding starts, so decryption can remove the padded data.
- Commonly used padding scheme is PKCS#5
  - PKCS: Public-Key Cryptography Standards

# Advantages and Limitations of ECB

- Message repetitions may show in ciphertext
  - If aligned with message block
  - Particularly with data such graphics
  - Or with messages that change very little

Original image

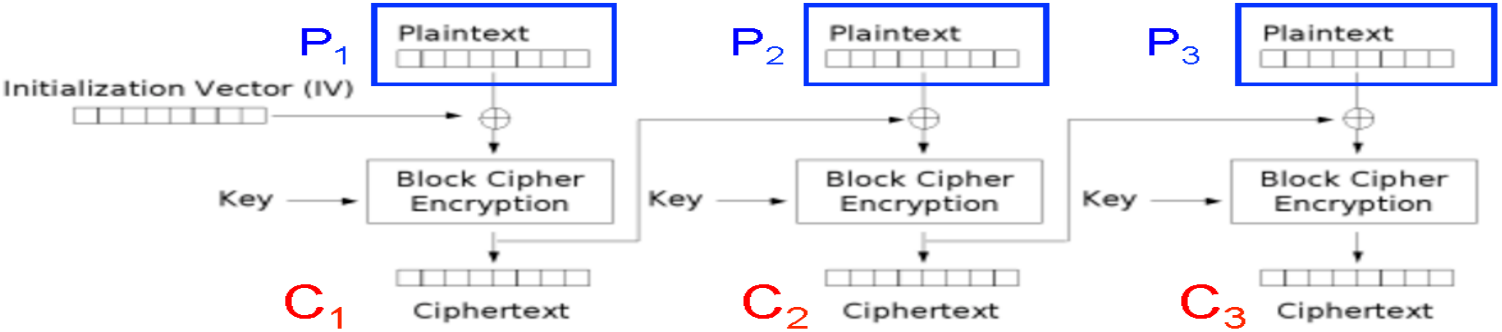


Encrypted with ECB

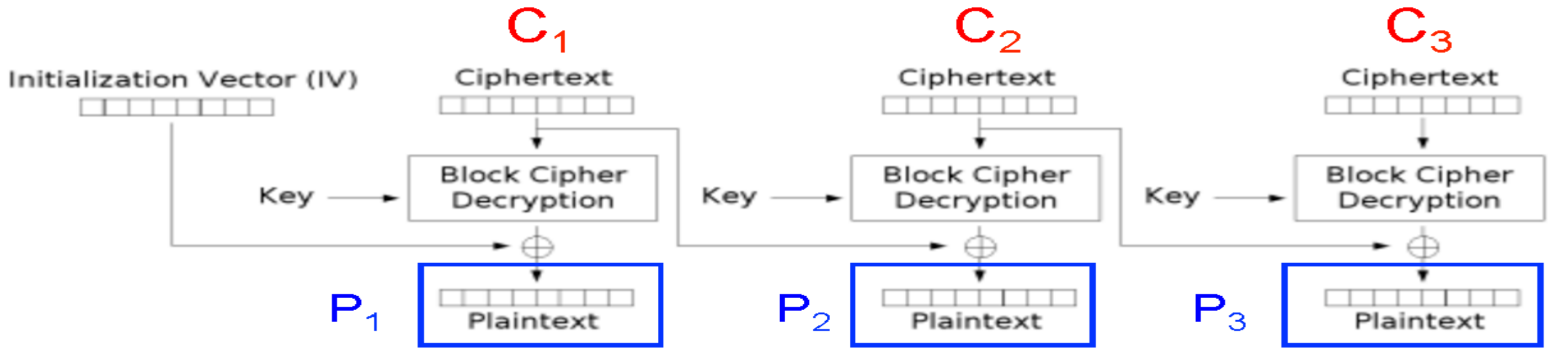
# Cipher Block Chaining (CBC)

- Message broken into blocks
- Blocks “chained” in encryption
- Initial Vector ($IV$) to start process

$$C_i = E_K(P_i \; xor \; C_{i-1})$$
$$C_{-1} = IV$$




Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

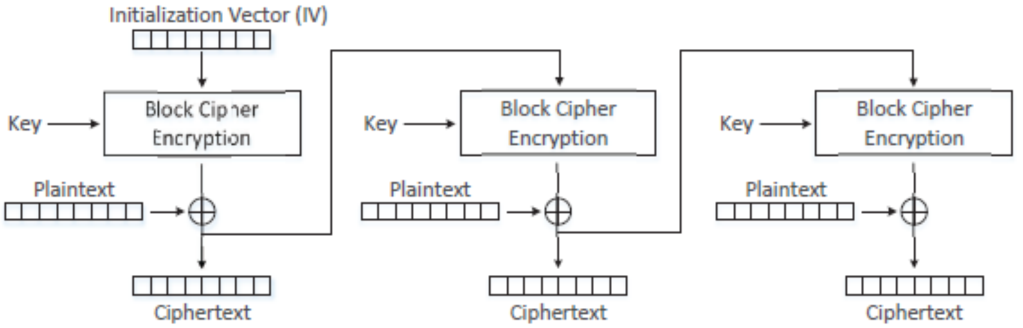
# Advantages and Limitations of CBC

- Ciphertext block depends on **all** blocks before it
- Change to a block affects all following blocks
- Need **Initialization Vector** (IV)
  - Random numbers
  - Must be known to sender & receiver

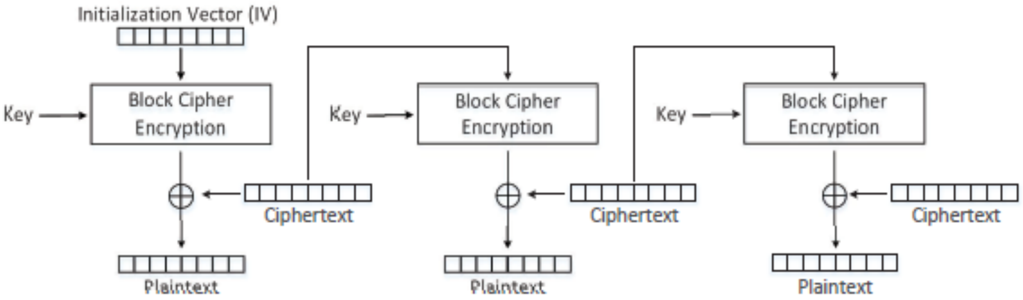
Original image

Encrypted with CBC

# Cipher Feedback (CFB) Mode



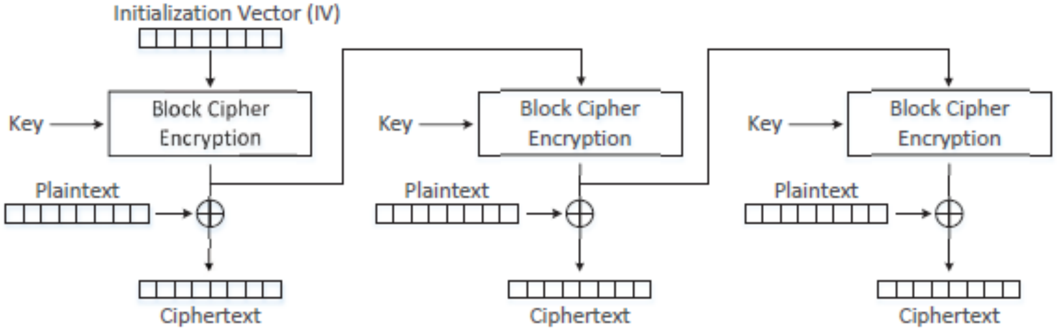

(a) Cipher Feedback (CFB) mode encryption



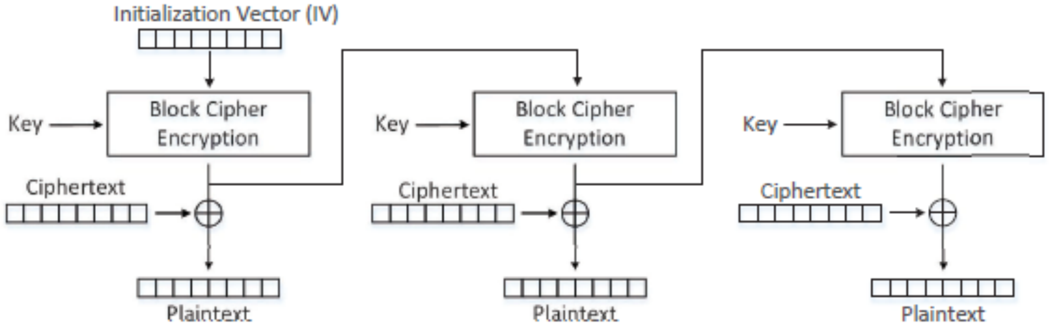

(b) Cipher Feedback (CFB) mode decryption

- Padding not required for the last block

# Output Feedback (OFB) Mode



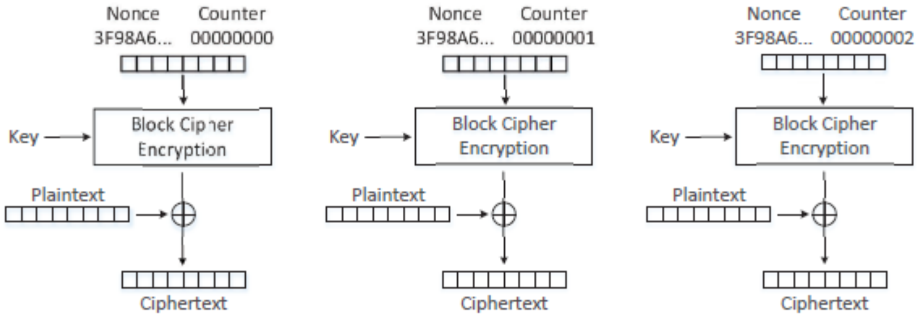

(a) Output Feedback (OFB) mode encryption



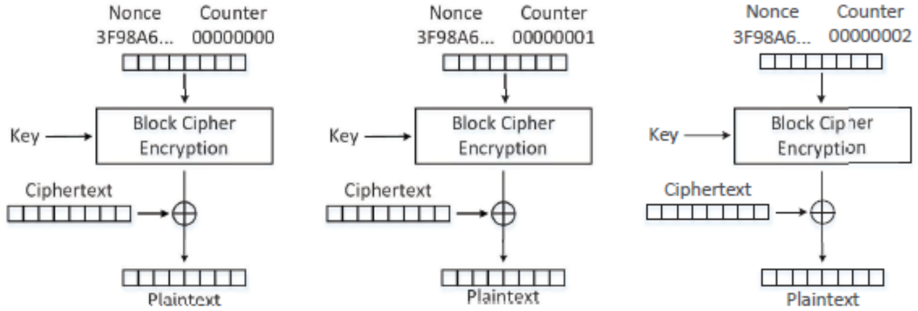

(b) Output Feedback (OFB) mode decryption

- Does not need padding

# Counter (CTR) Mode




(a) Counter (CTR) mode encryption




(b) Counter (CTR) mode decryption

- It uses a nonce + counter to generate the key streams
- This nonce serves the same role as the IV does to the other encryption modes.

# Initial Vector

- Initial vectors have the following requirements:
  - IV should not repeat (uniqueness).
  - IV should not be predictable.
  - SSL/TLS: send to the other endpoint in plaintext

# Hash functions

- Properties
  - Variable input size
  - Fixed output size (e.g., 512 bits)
  - Efficient to compute
  - Pseudo-random (mixes up input well)

# Cryptographic hash functions

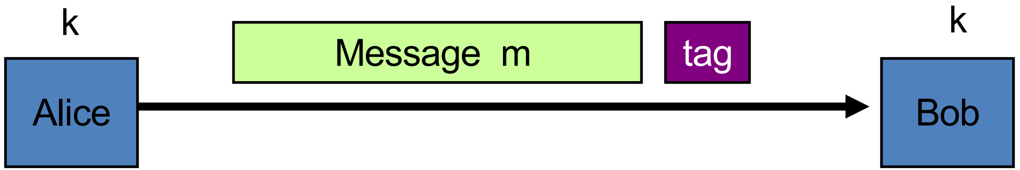
- Cryptogtaphic hash functions add conditions
- Preimage resistance
  - –Given *h*, intractable to find *y* such that $H(y)=h$
- Second preimage resistance
  - –Given *x*, intractable to find $y \neq x$ such that $H(y)=H(x)$
- Collision resistance
  - –Intractable to find *x, y* such that $y \neq x$ and $H(y)=H(x)$

# We have a cryptographic hash function crisis

- Popular hash function MD5
  - Thoroughly broken
- Government standard function SHA-1, SHA-2
  - Theoretical weaknesses
- “New” cryptographic hash function SHA-3
  - Too new to fully evaluate
  - Maybe good enough

## Message Integrity: Hashed Message Authentication Codes (HMACs)

- Goal: provide message integrity and authenticity
  - –Was my message (plaintext or cipertext) tampered during transmission?




Generate tag:

$tag \leftarrow S(k, m)$

Verify tag:

Bob receives (m’, tag)

tag’=S(k,m')

tag’=tag?

# HMAC

H: hash function.

example: SHA-256; output is 256 bits

Building a MAC out of a hash function:

– Standardized method: HMAC

opad, ipad: fixed strings

$$S(k, m) = H\Big( k \oplus opad \;||\; \mathbf{H(k \oplus ipad \;||\; m)} \Big)$$

# Summary

- Confidentiality
  - Block cipher
  - Stream cipher
  - Mode of operation
- Integrity and authenticity
  - HMAC