CS5321 Network Security
Week1: Crypto Basics (1)

Daisuke MASHIMA

http://www.mashima.us/daisuke/index.html

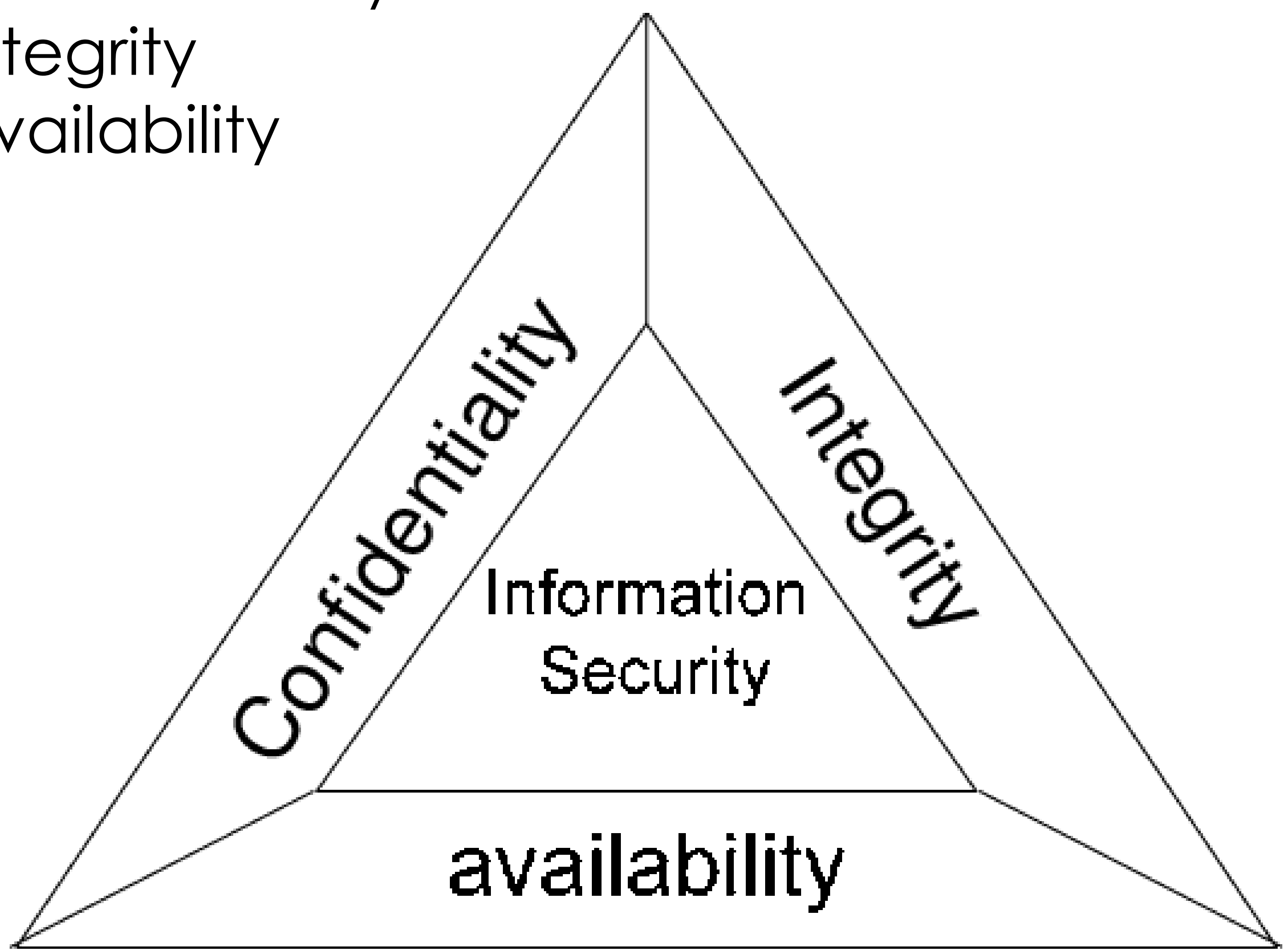
2022/23 Sem 2

Reference

Introduction to Modern Cryptography Second Edition
by Jonathan Katz and Yehuda Lindell

CIA Triad

3 Principles in information security
- **C**onfidentiality
- **I**ntegrity
- **A**vailability



Secrecy, Confidentiality, Privacy, Anonymity

- Often considered synonymous, but are slightly different
- **Secrecy**
  - Keep data hidden from unintended receivers
  - “Alice and Bob use encrypted communication links to achieve secrecy”
- **Confidentiality**
  - Keep **someone else’s data** secret
  - “Trent encrypts all user information to keep their client’s information confidential in case of a file server compromise”
- **Privacy**
  - Keep **data about a person** secret
  - “To protect Alice’s privacy, company XYZ did not disclose any personal information”
- **Anonymity**
  - Keep **identity** of a protocol participant secret
  - “To hide her identity to the web server, Alice uses The Onion Router (TOR) to communicate”

Integrity, Authentication

- Sometimes used interchangeably, but they have different connotations
- **Data integrity**
  - Ensure data is "correct" (i.e., correct syntax & unchanged)
  - Prevents unauthorized or improper changes
  - "Trent always verifies the integrity of his database after restoring a backup, to ensure that no incorrect records exist"
- **Entity authentication or identification**
  - Verify the identity of another protocol participant
  - "Alice authenticates Bob each time they establish a secure connection"
- **Data authentication**
  - Ensure that data originates from claimed sender
  - "For every message Bob sends, Alice authenticates it to ensure that it originates from Bob"

Difference between Integrity and Authentication

- Integrity is often a property of **local or stored data**
  - For example, we want to ensure integrity for a database stored on disk, which emphasizes that we want to prevent unauthorized changes
  - Integrity emphasizes that data has not been changed
- Authentication used in **network context**, where entities communicate across a network
  - Two communicating hosts want to achieve data authentication to ensure data was not changed by network
- Data authentication emphasizes that data was created by a specific sender
  - Implies **integrity of data**
  - Implies that **identity of sender** is verified

Signature, Non-repudiation

- Signature: non-repudiation of origin
  - Binds data to an identity
  - The signer cannot deny having created the signature
  - “Alice’s signature provides non-repudiation, preventing her from denying receipt of the document”

Difference between Authentication and Signature

- Authentication enables the receiver to verify origin, but receiver cannot convince a third party of origin
- Signature enables the receiver to verify origin, and receiver can convince third party of origin as well
- Signature provides authentication + public verifiability

Other Properties

- **Authorization**
  - Allowing another entity to perform an action
- **Auditability**
  - Enable forensic activities after intrusions
  - Prevent attacker from erasing or altering logging information
- **Availability**
  - Provide access to resource despite attacks
  - Denial-of-Service (DoS) attacks attempt to prevent availability

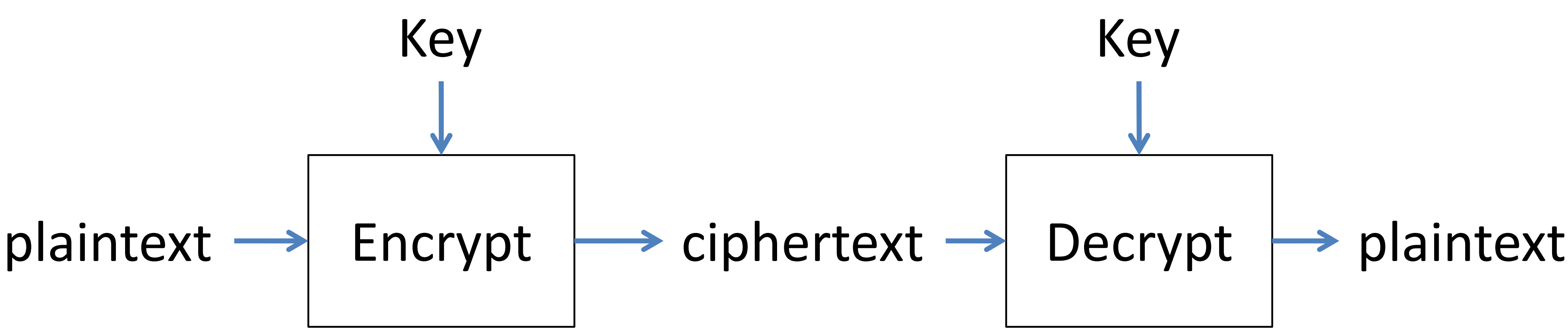
Basic Cryptographic Primitives

- Symmetric (shared-key, same-key)
  - Block cipher (pseudo-random permutation PRP)
  - Stream cipher (pseudo-random generators PRG)
  - Message authentication code (MAC)
  - Authenticated encryption
- Asymmetric (public-private key)
  - Public-key encryption
  - Digital signature
  - Diffie-Hellman key agreement
- Others (unkeyed symmetric)
  - One-way function
  - Cryptographic hash function

Symmetric Encryption Primitives

- Often called shared key crypto or secret key crypto

- Encryption key = decryption key
- Encryption: E_K(plaintext) = ciphertext
- Decryption: D_K(ciphertext) = plaintext
- We write $\{\text{plaintext}\}_K$ for E_K(plaintext)



Stream Ciphers

- One-time pad
  - Use unique random keystream for each message

    e.g., $c_1 = k_1 \oplus p_1$, $c_2 = k_2 \oplus p_2$, …
  - Is this secure?
  - Is this secure if we re-use keystream?

c_i: ciphertext
p_i: plaintext
k_i: key
$\oplus$: XOR

(src: http://cryptomuseum.com/crypto/otp/index.htm)

Stream Ciphers

- Stream ciphers use pseudo-random generator (PRG) to generate keystream from seed
  - Encryption: use shared key k and initialization vector IV for the seed.

    $$\text{ciphertext} = \text{plaintext} \oplus \text{PRG}(k, IV)$$

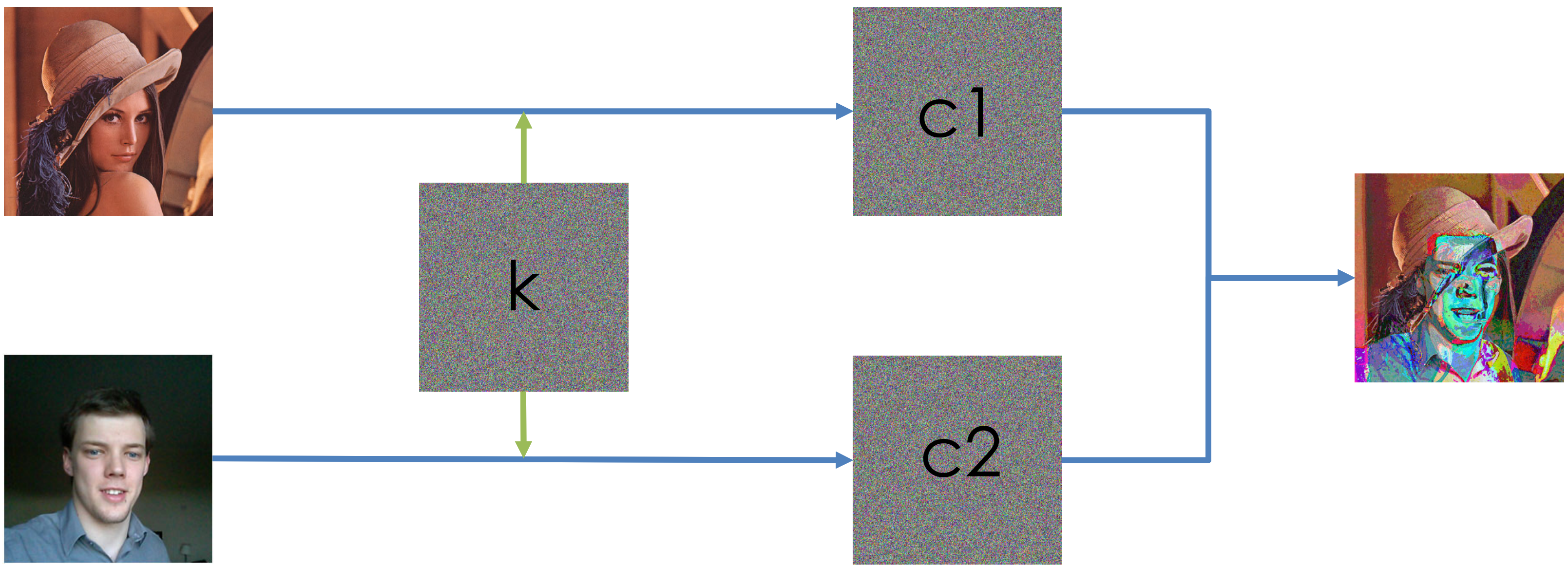
  - Send IV, ciphertext
- Example: RC4, AES in CTR mode

- What could go wrong?
  - What if the system reuse the IV?
  - What if the ciphertext is modified during transmission?

Stream Cipher Vulnerabilities

- Keystream reuse attack
  - Enormous security vulnerability if same keystream used to encrypt to different messages
  - $c1 = p1 \oplus k$, $c2 = p2 \oplus k$
  - $c1 \oplus c2 = p1 \oplus p2$ (which is easy to analyze, because the unknown key is removed!)
  - $c1 = p1 \oplus PRG(K, IV)$, where IV=initialization vector, make sure IV is never used twice!

c_i: ciphertext
p_i: plaintext
$\oplus$: XOR



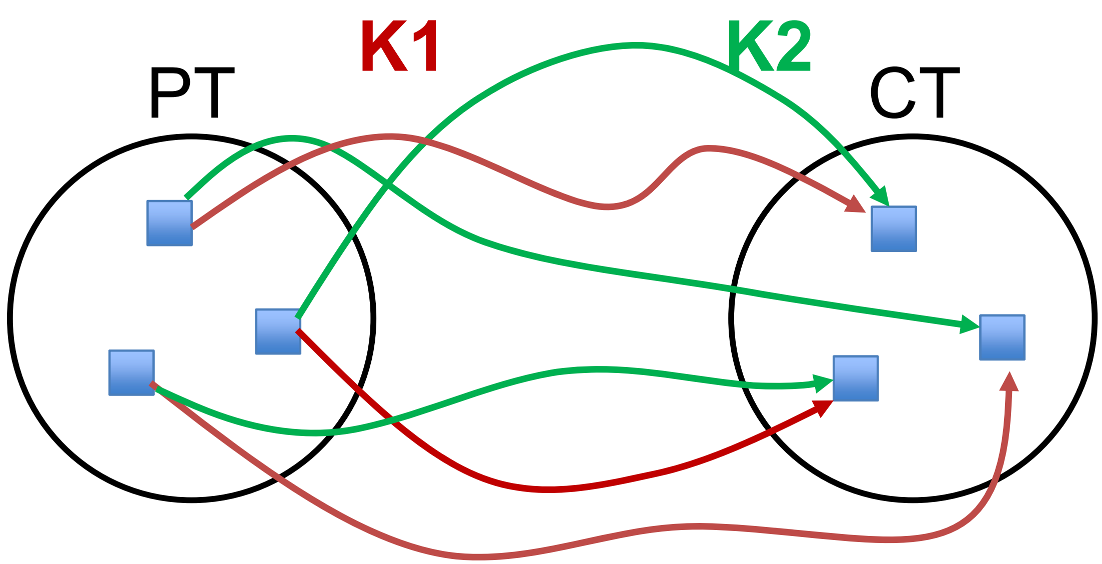
https://incoherency.co.uk/blog/stories/otp-key-reuse.html

Stream Cipher Vulnerabilities

- Ciphertext modification attack
  - Alteration of ciphertext will alter corresponding values in plaintext after decryption
  - Example, encrypt a single bit: $c = p \oplus k$, for p=1, k=0, thus c=1
  - If attacker changes c to 0 during transmission, decrypted value is changed to 0! $p = c \oplus k$, if c=0, then p=0.
  - To defend, need to ensure authenticity of ciphertext

Block Ciphers

- Block cipher is a **pseudo-random permutation (PRP)**, each key defines a one-to-one mapping of input block to output block
- Encrypt each block separately
- Examples: DES, RC5, AES



key-size should be at least 128 bit

Advanced Encryption Standard: AES

- AES is successor to (US-selected) DES
- Officially adopted for US government work, but voluntarily adopted by private sector
- Winning cipher was Rijndael (pronounced Rhine-doll)
  – Belgian designers: Joan Daemen & Vincent Rijmen
- Adopted by NIST in Nov 2001
- {128, 192, 256}-bit key size
- High-speed cipher
  – Software: 28 cycles/byte
  – Intel’s AES-NI: 3.5 cycles/byte

Modes of Operation

- Block cipher **modes of operation**
  - ECB: Electronic code book
  - CBC: Cipher block chaining
  - CFB: Cipher feedback
  - OFB: Output feedback
  - CTR: Counter mode

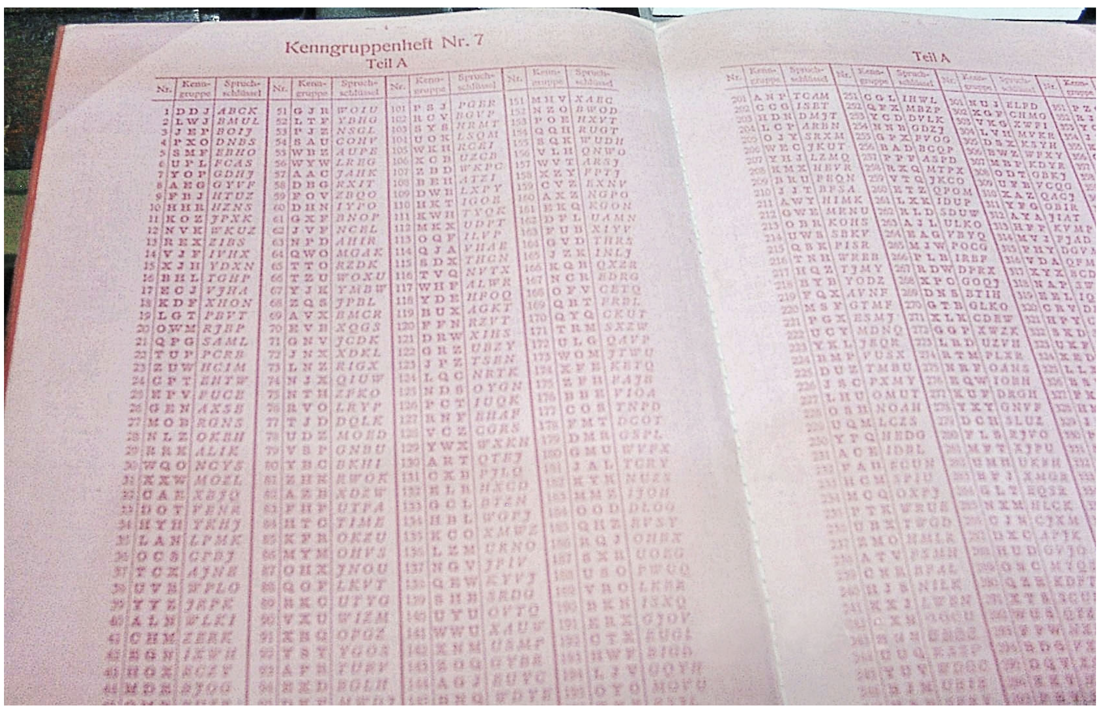
block cipher in a stream cipher mode

ECB: Electronic code book

- Natural approach for encryption: given a message M, split M up into blocks of size b bits (where b = input size of block cipher), $M_1, \ldots, M_n$
  - Last block includes "padding"
- $E_K(M) = E_K(M_1) \mid\mid E_K(M_2) \mid\mid \ldots \mid\mid E_K(M_n)$

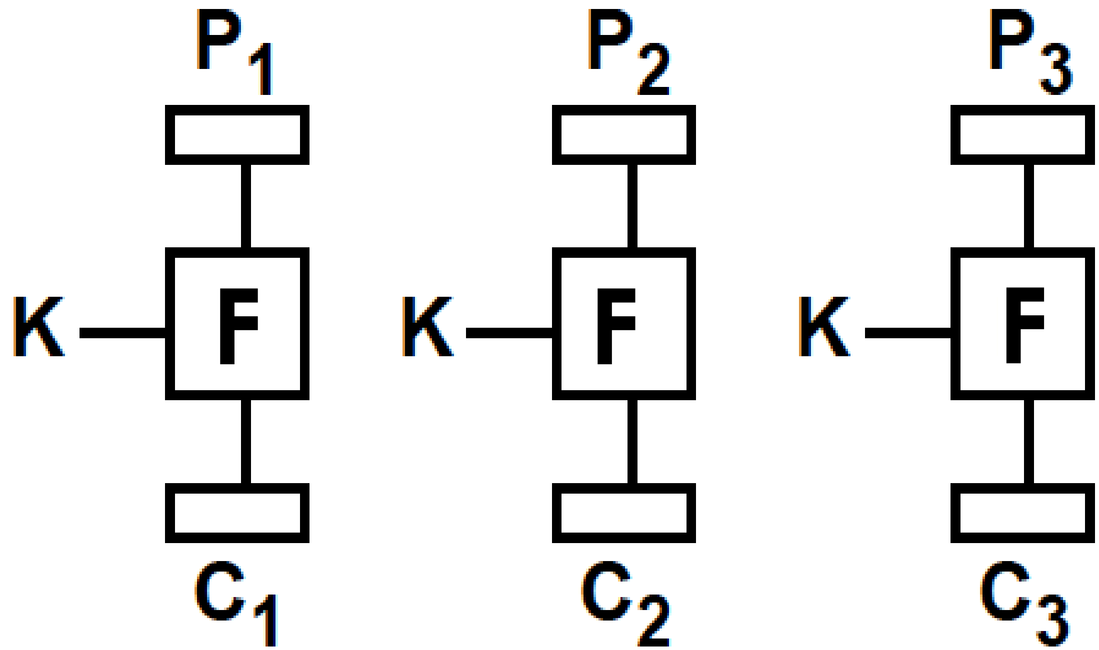
|| : concatenation

- This approach is called **Electronic Code Book mode (ECB mode)**



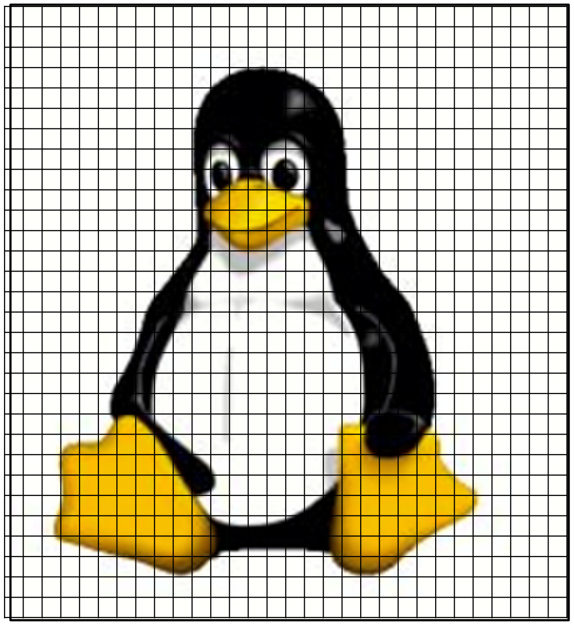
ECB Mode

- $C_j = F_K(P_j) = E_K(P_j)$
- $P_j = F^{-1}_K(C_j) = D_K(C_j)$

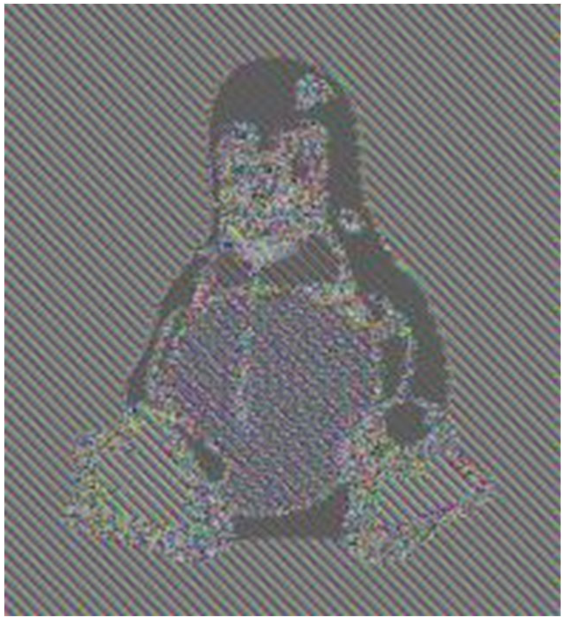


- Advantages
  - Simple to compute
- Disadvantages
  - Same plaintext always corresponds to same ciphertext
  - Traffic analysis yields which ciphertext blocks are equal → know which plaintext blocks are equal
  - Adversary may be able to guess part of plaintext, can decrypt parts of a message if same ciphertext block occurs

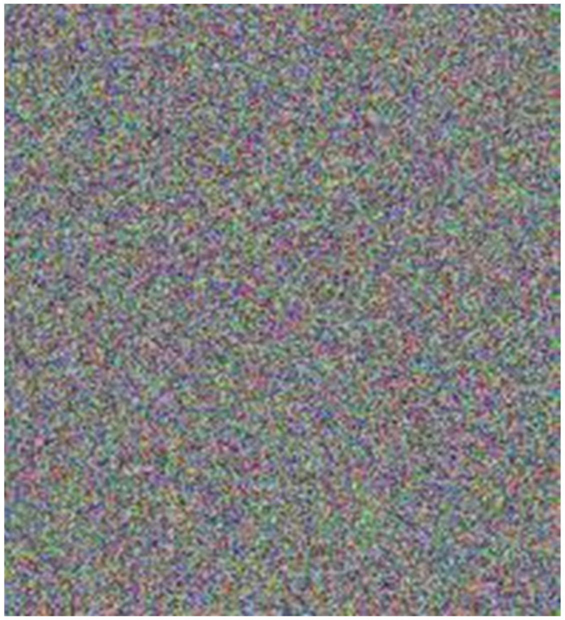
Problems of ECB Mode



plaintext



ciphertext of ECB mode



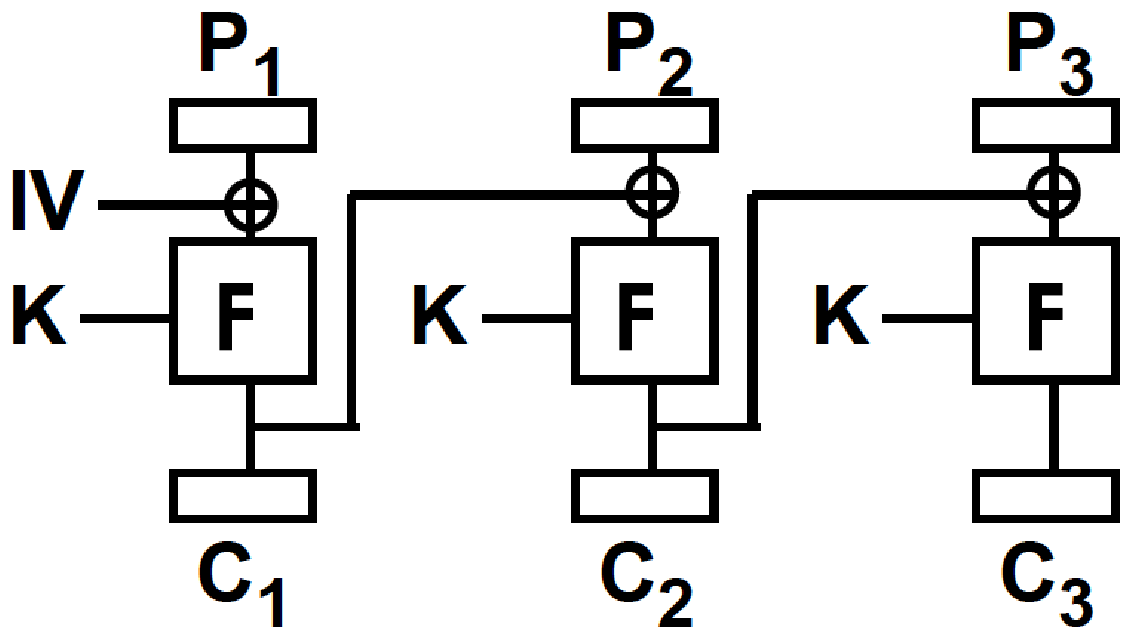
We want ciphertext like this!

Desired Properties

- **Semantic security**: if adversary had to guess value of a single plaintext bit, can guess at best at random
  - i.e., indistinguishability in CPA sense
  - Even if we keep encrypting a 1-bit message with skewed distribution (e.g., a fire alarm message, which almost always carries the same plaintext bit), attacker cannot guess value of plaintext given ciphertext

Cipher Block Chaining (CBC)

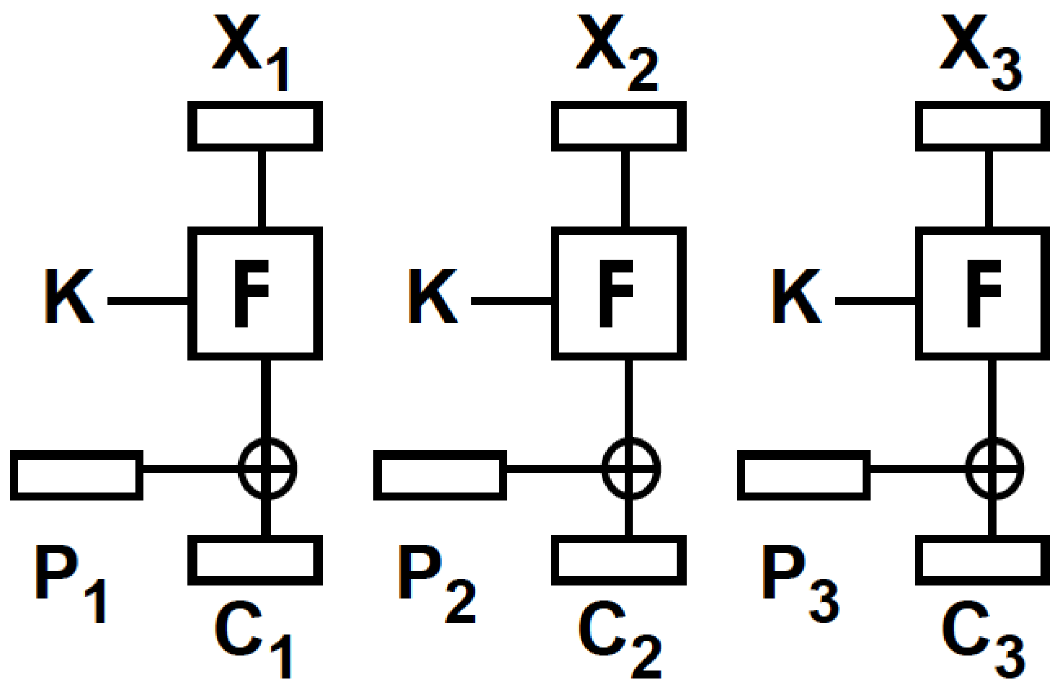
- $C_j = F_K(P_j \oplus C_{j-1})$
  $= E_K(P_j \oplus C_{j-1})$
- $P_j = F^{-1}_K(C_j) \oplus C_{j-1}$
  $= D_K(C_j) \oplus C_{j-1}$
- C_0 = IV called initialization vector



- Advantages
  - Semantic security
- Disadvantages
  - Cannot be parallelized

Counter Mode (CTR)

- X_1 = IV called initialization vector
- $X_j = X_1 + j - 1$
- $C_j = F_K(X_j) \oplus P_j$
  $= E_K(X_j) \oplus P_j$
- $P_j = F_K(X_j) \oplus C_j$
  $= E_K(X_j) \oplus C_j$



- Advantages
  - Semantic security
  - Can be parallelized

How to use block ciphers matters!

- CBC/CTR modes of encryption provide semantic security when they are ***correctly used***
- Example the ***wrong*** use of IV:
  - For CTR and CBC, **reuse** of IV breaks the encryption
  - For CBC, **predictable** IV breaks the encryption (e.g., SSL 2.0)
    - *Predictable IV does not break security of CTR mode. But we need to make sure counter value (X_i) is never reused.*
- Padding can be used to break CBC encryption (a.k.a. padding oracle attack)
  - If the decryption module tells whether padding is valid or not, an attacker can utilize that information for revealing the whole plain text.

Example of Risk of Predictable IV

- Let's assume the plain text space is "true" or "false" (e.g., history of a certain disease).
- Eve, an adversary who has capability to read DB entries, knows that Alice will be uploading the above plaintext to a DB.
- The DB encrypts it with CBC mode.
  - DB uses a secret key K for all users.
  - When encryption, DB uses IV_A, which is IV for Alice's data
  - DB stores E(K, P XOR IV_A), where P is plaintext from Alice
- Here, Eve, knowing IV_A and her own IV IV_E, can construct plaintext, "true" XOR IV_A XOR IV_E
  - DB, using IV_E, encrypts it as E(K, ("true" XOR IV_A XOR IV_E) XOR IV_E) = E(K, ("true" XOR IV_A)
- Finally, Eve can compare the above ciphertext against Alice's DB entry to know if she answered "true" or not.

Confidentiality vs. Integrity

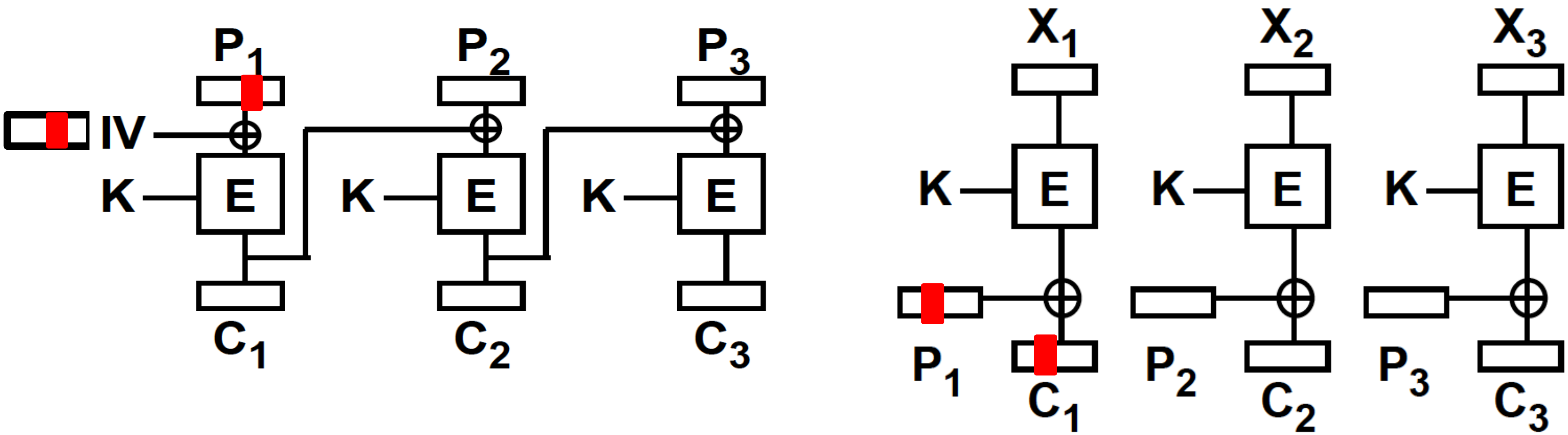
- What does “secure communication” entail?
  - So far, we aim to obtain secret communication (i.e., message confidentiality)
  - Yet, not all security concerns are related to confidentiality
  - In many cases, message integrity is equally (or more) important

Encryption vs. authentication

- "Encryption hides message contents and thus adversary cannot modify encrypted message in any meaningful way" T/F?
  - No! Common mistake.
  - Encryption ***does*** ***<u>not</u>*** ***provide authentication*** and thus one should ***<u>not</u>*** expect ***authentication*** from encryption

- Examples:
  - CTR-mode encryption
    - ciphertext bit flip => plaintext bit flip
  - CBC-mode encryption
    - IV bit flip => first message block bit flip

Lack of integrity in CBC and CTR



Message Authentication Codes

- Message authentication codes (MAC) provide a "cryptographic checksum" for ***authentication***
  - Write MAC(K, M), or MAC_K (M)
  - K is a shared symmetric key
- Use
  - A and B share symmetric key K_{AB}
  - A → B: M, t=MAC(K_{AB} , M)
- Two examples:
  - Hash-based MAC: HMAC
    - HMAC-SHA1(K, M) = SHA1((K $\oplus$ opad) || SHA1((K $\oplus$ ipad) || M))
      - ipad = 0x3636..36, opad = 0x5C5C..5C
  - Block-cipher based MAC: CBC-MAC
    - Use 0 as IV and the final block becomes MAC
    - Secure for message of fixed size

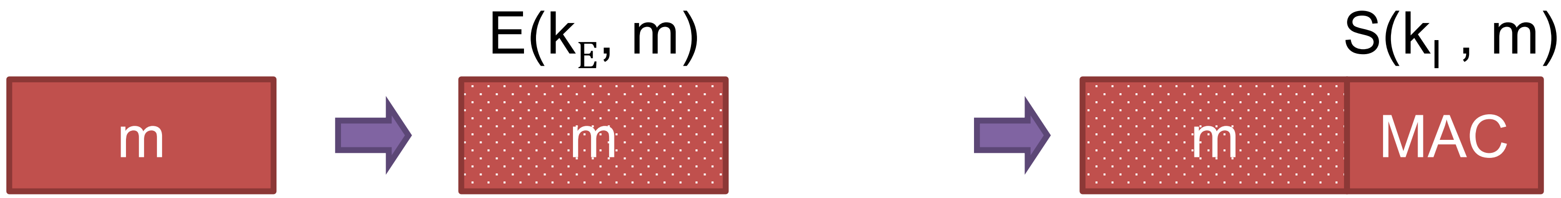
Authenticated Encryption

- Authenticated Encryption
  - **Semantic security**
  - **Ciphertext integrity**: An attacker cannot create a ciphertext that decrypts properly.

- Authenticated Encryption using Symmetric-key Encryption
  - Encrypt-and-MAC (a.k.a. Encrypt-and-Authenticate): SSH
  - MAC-then-Encrypt (a.k.a. Authenticate-then-Encrypt): SSL/TLS
  - Encrypt-then-MAC (a.k.a. Encrypt-then-Authenticate): IPSec

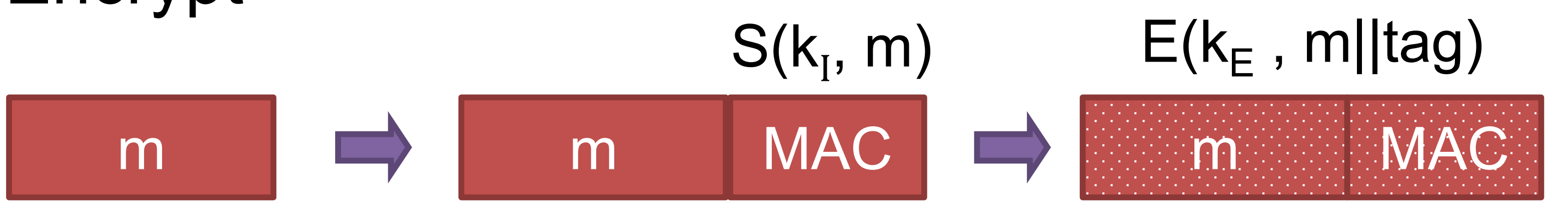
Construction of Authenticated Encryption

Encrypt-and-MAC

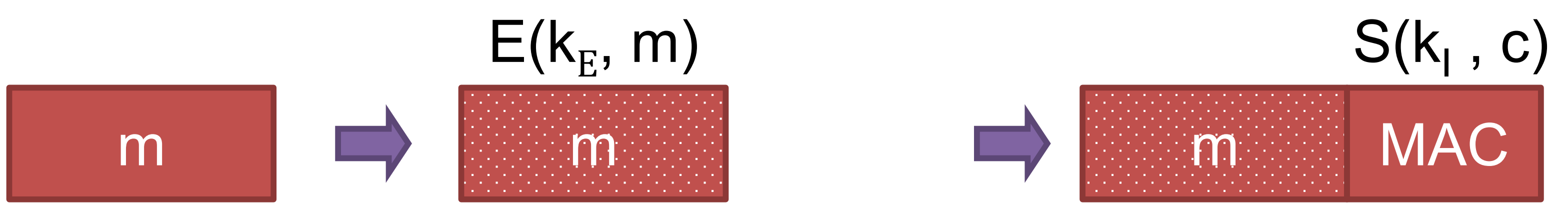


- If the same message m is encrypted twice, same MAC is seen! (not ensure semantic security)

MAC-then-Encrypt

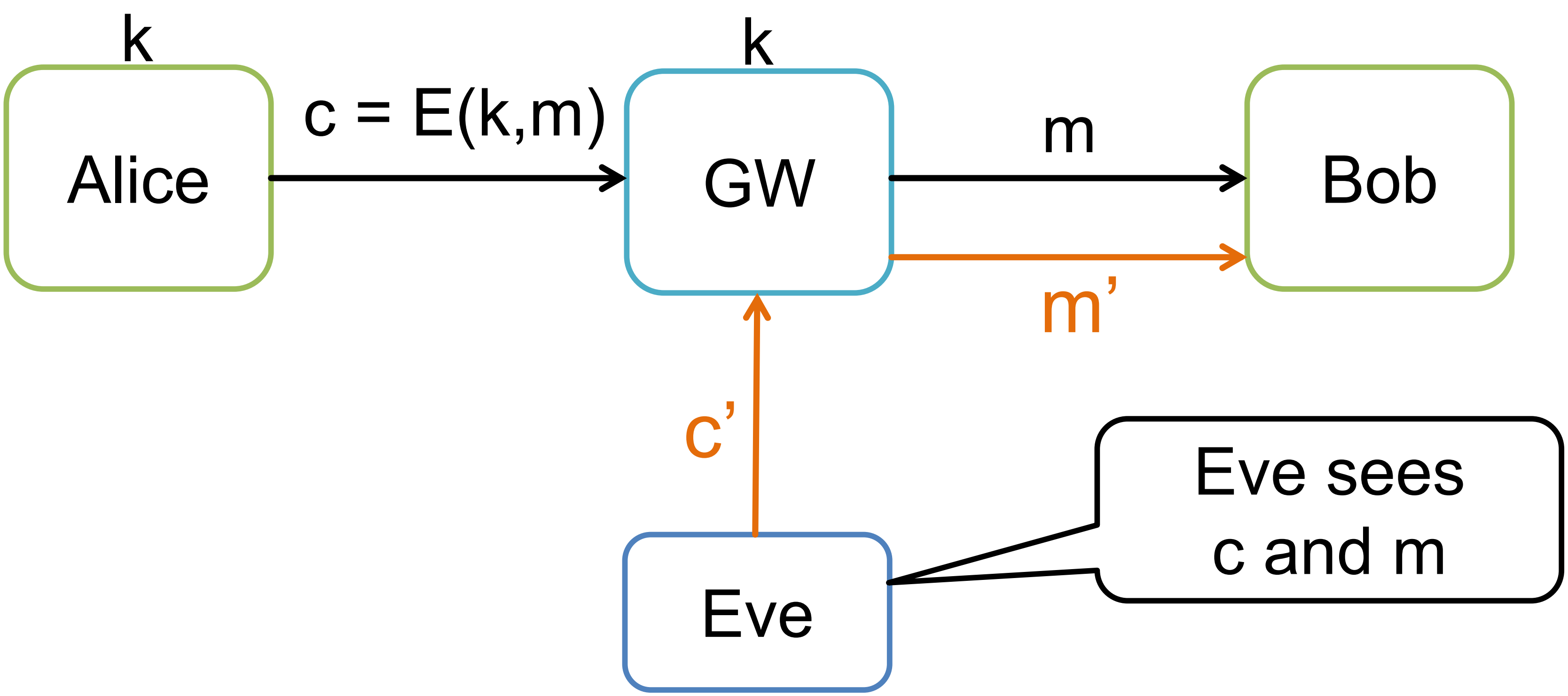


Encrypt-then-MAC



CPA Security vs CCA Security

- 2 different attacker models against semantic security:
  - Chosen plaintext attack (CPA)
    - Attacker can get ciphertexts for plaintext of his choice
    - Passive attacker
  - Chosen ciphtertext attack (CCA)
    - Additionally, attacker can get plaintext of ciphertext of his choice



MAC-then-Encrypt vs Encrypt-then-MAC

- MAC-then-encrypt
  - Secure under CPA (with appropriate selection of encryption and MAC scheme)
  - MAY be insecure under CCA
- Encrypt-then-MAC
  - Proven to be secure against both (with appropriate selection of encryption and MAC scheme)
- Although we don't dive into details, interested students can take a look at the following papers:
  - https://eprint.iacr.org/2000/025.pdf
  - https://iacr.org/archive/crypto2001/21390309.pdf
- One (practical) advantage of Encrypt-then-MAC is to avoid unnecessary decryption for garbage data received.

What's next?

- Symmetric (shared-key, private-key)
  - Block cipher (pseudo-random permutation, PRP)
  - Modes of encryption
  - Message authentication code (MAC)
  - Authenticated Encryption
- **Asymmetric (public-private key)**
  - **Diffie-Hellman key agreement**
  - **Public-key encryption**
  - **Digital signature**
- **Others (unkeyed symmetric)**
  - **Cryptographic hash function and more**

QUESTIONS?