Name: Madhav Verma
Roll: CrS2406
Subject: Cryptographic & Security Implementations

Question: Why is ECB mode insecure? Which mode should be used instead and why?

Why ECB mode is insecure

ECB encrypts each 16 byte block independently with the same key. Formally, for block i,

$$C_i = E_K(P_i).$$

There is no randomness and no chaining across blocks. This leads to several failures.

Equality leakage and loss of semantic security If two plaintext blocks are equal then their ciphertext blocks are equal:

$$P_i = P_j \Rightarrow C_i = C_j.$$

An observer who only sees ciphertext learns where blocks repeat and how often they repeat. This leaks structure from images, records, headers, and protocol fields, so it breaks semantic security.

No IV or nonce means cross message linkage Because ECB has no initialization vector and no nonce, the first block of any message with the same prefix encrypts to the same ciphertext block. An adversary can build a codebook of common blocks and later recognize them in unrelated messages.

Malleability and splicing ECB provides confidentiality only at the single block level and no integrity at all. An attacker can cut and paste ciphertext blocks to rearrange the decrypted message in a controlled way. Bit flips within a block cause uncontrolled changes, but block level splicing is trivial.

No replay or ordering guarantees ECB gives no assurance of freshness or correct ordering. Replayed or shuffled ciphertext blocks will decrypt to replayed or shuffled plaintext blocks without detection.

Chosen plaintext distinguishers are trivial The deterministic nature of ECB lets an attacker test guesses about unknown plaintext by submitting chosen plaintext blocks and checking for equality of ciphertext blocks.

Recommended modes and why

AES GCM (preferred) Galois Counter Mode combines counter mode encryption with a polynomial authenticator (GHASH). It yields an authenticated encryption with associated data

(AEAD) scheme: it provides confidentiality and integrity in one pass, requires no padding, and is fast and parallel friendly.

- Use a unique 96 bit nonce per key. Never reuse a key nonce pair.
- Verify the authentication tag before accepting or using the plaintext.
- Include headers or metadata as associated data so tampering is detected.

AES CTR with a MAC Counter mode turns AES into a stream of keystream blocks. Encryption is

$$C_i = P_i \oplus E_K(\text{nonce} \,\|\, \text{ctr}_i).$$

Add a separate message authentication code, for example HMAC, over the nonce and the ciphertext (encrypt then MAC). CTR is simple, parallel, and needs no padding, but you must ensure nonce uniqueness and you must authenticate.

AES CBC with a MAC Cipher Block Chaining hides repeated blocks if every message uses a fresh unpredictable IV. It requires padding and must be authenticated to avoid padding oracle and other chosen ciphertext attacks. Use only for legacy compatibility. In new designs prefer GCM or CTR with a MAC.

Nonce misuse resistance If accidental nonce reuse is a realistic risk, choose AES GCM SIV or AES SIV. These remain secure under nonce reuse by deriving the keystream from a synthetic IV that depends on the message, at some performance cost.