AES — Conceptual Guide

How AES transforms data (no deep math)

Version	1.0
Date	2025-10-19
Owner	Security Team
Audience	Engineering & Non■specialists

1) Core idea

- AES is a block cipher: it transforms fixed-size blocks (128 bits = 16 bytes) using a secret key (128/192/256 bits).
- Same input + same key → same output; security comes from complex, key-dependent mixing per round.
- To encrypt long data, blocks are combined with a mode of operation (e.g., GCM, CTR); AES itself handles only 16■byte blocks.

2) Data structures

- State: a 4×4 byte grid holding the current 16■byte block.
- Round keys: AES expands the original key into multiple round■specific subkeys (key schedule).
- Rounds: 10 rounds (AES■128), 12 (AES■192), 14 (AES■256).

3) What each round does (conceptually)

Step	Purpose (conceptual)
SubBytes	Non■linear substitution on each byte (confusion: hide key relations).
ShiftRows	Rotate rows of the grid (permute positions to spread changes).
MixColumns	Mix bytes within each column (diffusion: spread local change across the column).
AddRoundKey	XOR the state with the round key (inject secret).

- Encryption: initial AddRoundKey → rounds 1..(r−1) with all 4 steps → final round without MixColumns.
- Decryption: inverse steps in reverse order with inverse round keys (designed to perfectly undo encryption).

4) Key schedule (conceptual)

- Derives a sequence of round keys from the original key using byte substitutions, rotations, and XOR with constants.
- Goal: each round key looks unrelated to the others; small key changes cause widespread state changes.

5) Putting it together

- Start: place 16■byte plaintext into the state; XOR with the first round key.
- Repeat: apply substitution + permutations + mixing + round■key XOR for the configured rounds.
- Finish: after the final round (no MixColumns), the 16■byte state is the ciphertext block.
- For long messages: a mode (e.g., CTR, GCM) chains blocks and introduces nonces/IVs to ensure uniqueness and integrity (GCM).
- AES by itself gives confidentiality; integrity requires an AEAD mode (e.g., GCM) or a separate MAC.

6) Practical, non■math notes

- Operations are byte■level substitutions and XOR/linear mixes — highly efficient on modern CPUs and hardware.

- Small input changes avalanche across the state after a few rounds; outputs appear random without the key.
- Security relies on using the right mode and never reusing nonces in AEAD modes.