Date:

#Symmetric

AES

- Advanced Encryption standard.
- Performs block by block Encryptions
- keyed permutation
- It works on 128 bit (16 byte) blocks & 128 bit key known as AES-128.
- we can perform that permutation in reverse using that key
- one to one correspondence ← Bijection.
- 128 bit key = large to Brute force
- There is technique to reduce security from 128 bit to 126.1 bits.
- Technically better than Brute force.
- Biclique Attack → Best single key attack against AES
- Grover's Algo (for Quantum) could theoritically cut security of symmetric cipher in half.
- So using AES 256 is recommended.

Date: Su Mo Tu We Th Fr Sa

Structure of AES.

↳ Begins with key-shcedule algo & then runs 10 rounds over a state. Starting state is plain text block represented as 4x4 matrix.

① key schedule.
② Initial key Addition
↳ AddRound key - the bytes of first round key are XOR'd with bytes of state.
③ Round - this phased is looped 10 times, for 9 main rounds & one final round.
↳ subbytes - Each byte of state is substituted for different byte according to lookup table (S-box).
↳ Shift rows - Last 3 rows of state matrix are transposed
↳ Mix Columns
↳ Add Round key.

Date:

Su Mo Tu We Th Fr Sa

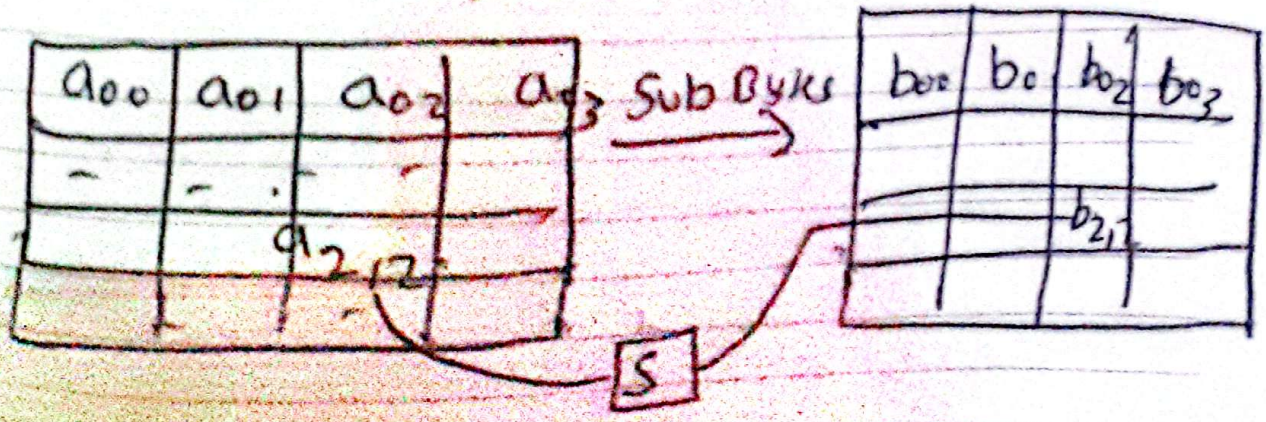
#Key Expansion Phase

↳ Takes 16 byte key & produces 11 4x4 matrices called round keys.

↳ The Initial key Addition phase has single AddRoundkey step. which XOR's our current state with current round key.

↳ AddRoundkey also occours as final step in each round its what makes it keyed permutation.

First step of each AES round is SubBytes. It involves taking each byte of state matrix & substituting it for different byte in 16x16 lookuptable (substitution or S-Box).



Date: Su Mo Tu We Th Fr Sa

AES = Not perfect. Fast lookup in S-box is shortcut for performing very non-linear functions on input bytes. This function involves taking modular inverse ø in Galois field 2^8 & applying transformation. which has been tweaked for max confusion.

$$f(x) = 06x^{fe} + 09x^{8d} + f9x^{8b} + 25x^{f7} + f4x^{ef} + 01x^{d8} + b5x^{b8} + 8fx^{7c} + 63.$$

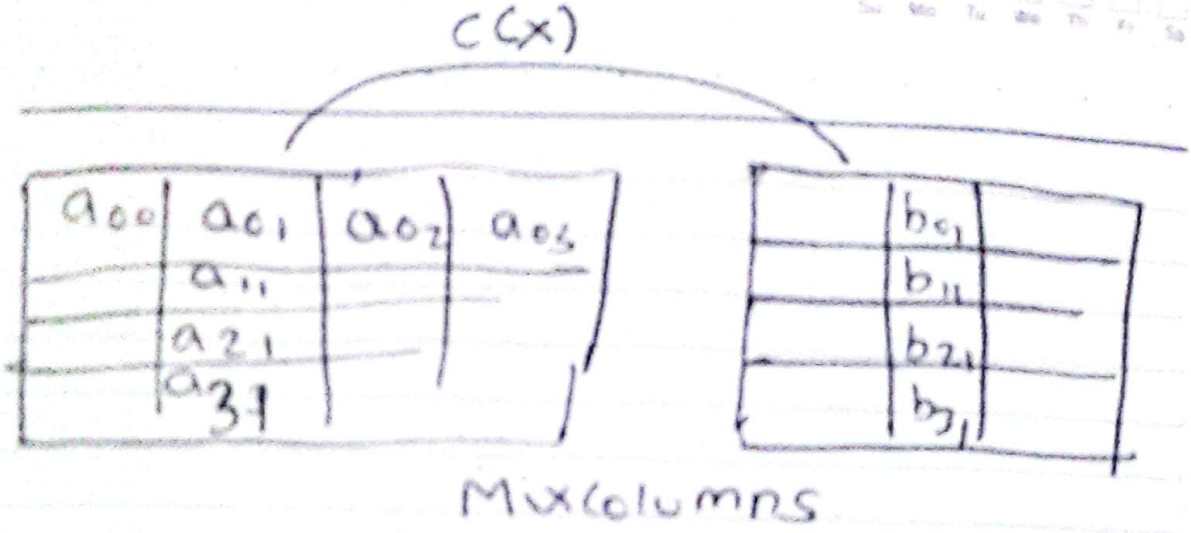
#Diffusion.

a) Without Diffusion, same byte in same position would get same transformation each round.

b) Ideal amount of diffusion causes change of one bit in plaintext to lead to a change in statistically half bits of ciphertext. It's called Avalanche effect.

c) ShiftRows & MixColumns steps combine to achieve diffusion.

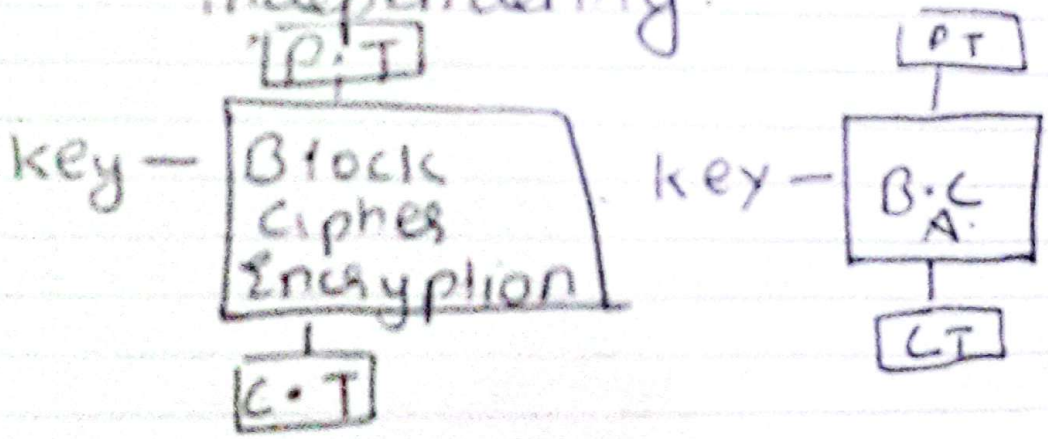
d) ShiftRows is simple as it says, MixColumns is complex & achieves it by matrix multiplication of state matrix & preset matrix.



MixColumns

ECB

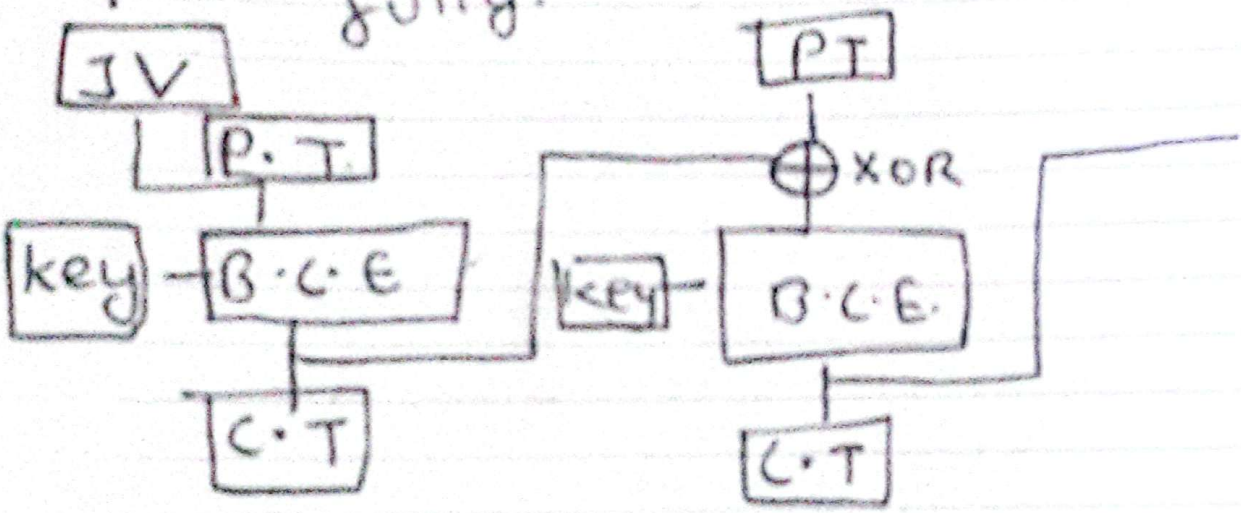
- Electronic code block
- Each Block encrypted independently



- Vuln
  - ECB mode encrypts identical plaintext blocks to identical ciphertext blocks
  - NO randomization IV.
- Characters Plaintext Attack
- Byte-Byte Extraction.

→ Initialization Vector.

CBC → Cipher Block - Chaining.
↳ Code block ciphers.
↳ Dependent on previous block.
↳ Secure than ECB but not fully.



↳ vuln
↳ Padding oracle
↳ Bit flipping

ECB CBC WIF → Flag
↳ Lets u encrypt in CBC but lets decrypt in ECB.

$C_0 = IV_1$, $C_i = AES_{CBC}(ENC(P_i))$

Decrypt only in ECB

$D(C_i) = AES\_ECB\_DEC(C_i)$

Date:

We need to recover flag using ECB decryption.

CBC Encryption working,

$C_0 = IV$

$C_1 = AES\text{-}ECB\text{-}ENC(P_i \oplus IV)$.

$C_2 = AES\text{-}ECB\text{-}ENC(P_2 \oplus C_1)$...

CBC Decryption

$P_1 = D(C_1) \oplus IV$

$P_2 = D(C_2) \oplus C_1$

Where $D(C_1)$ is ECB decryption of that block so if we find $D(C_i)$ we can get flag.

① Req & CBC enc flag.

$C = IV \,||\, C_1 \,||\, C_2$ ...

② Split in blocks

$IV = C[:16]$

$C1 = C[16:32]$

$C_2 = C[32:48]$

③ $D(C_1) = AES$ ECB dec of C_i

④ Recover P using CBC

$P_1 = D(C_1) \oplus IV$

$P_2 = D(C_2) \oplus C_1$.

⑤ Concanate all blocks for flag.

Flliping cookie

↳ Its using CBC bit flipping Attack.

↳ Cookie is enc in AES-CBC mode with random IV,

$C = AES\ CBC\ ENC(cookie, IV)$.

↳ we can decrypt cookie using AE-S, CBC, checks if admin =True & only return flag.

CBC-Bit-Flip

- CBC Dec:

$$P = D(C_i) \oplus C_{i-1}$$

↳ $C_0 = IV$
C_1 = first ciphertext block.
Each plaintext block is XOR of ECB dec CT.

↳ By flipping bits in C_{i-1} for IV for first block, you can flip in P_i then it bitwisely effect

Eg:
admin = false; expiry = 12345

We want admin = Tave;;
↳ ① find block where admin = false is
② Flip "F -> T in prev block or IV if it's first block
③ when dec, server sees "admin = Tave"

OFB Mode:

- Output Feedback Mode
- Turns Block cipher (AES) to stream ciphers
- Produces keystream independent of plaintext

$K_1 = AES(key, IV)$ $K_2 = AES(KEY, K_1)$ $K_3 = AES(KEY, K_2)$

Encryption/Dec is:

$C_i = P_i \oplus k_i$ $P_i = C_i \oplus k_i$

Steps

1. Keystream depends on key, IV.
2. XORing known P.T with its C.T reveals keystream.
3. If you know part of P.T, you can decrypt other msg using same IV.
4. This is similar to one time pad reuse.

Date: Su Mo Tu We Th Fr Sa

Symmetry (chall)

① Server gives IV + Cflag

② You can send P.T = "A" * N with same IV, get Cknown.

③ $K = C_{known} \oplus P.T$

④ $Flag = C_{flag} \oplus K$.

Only XOR operations.

#

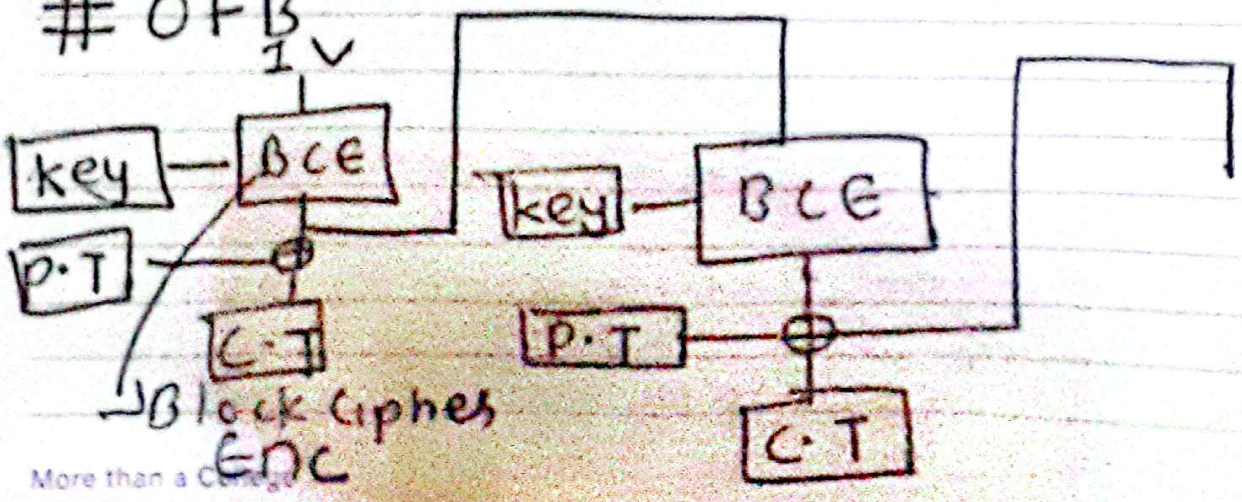
① encrypt-flag → get IV + Cflag

② call encrypt(Cknown P.T, IV) - get Cknown

③ XOR Cknown with known. P.T - get keystream.

④ XOR Cflag with keystream - get flag.

OFB

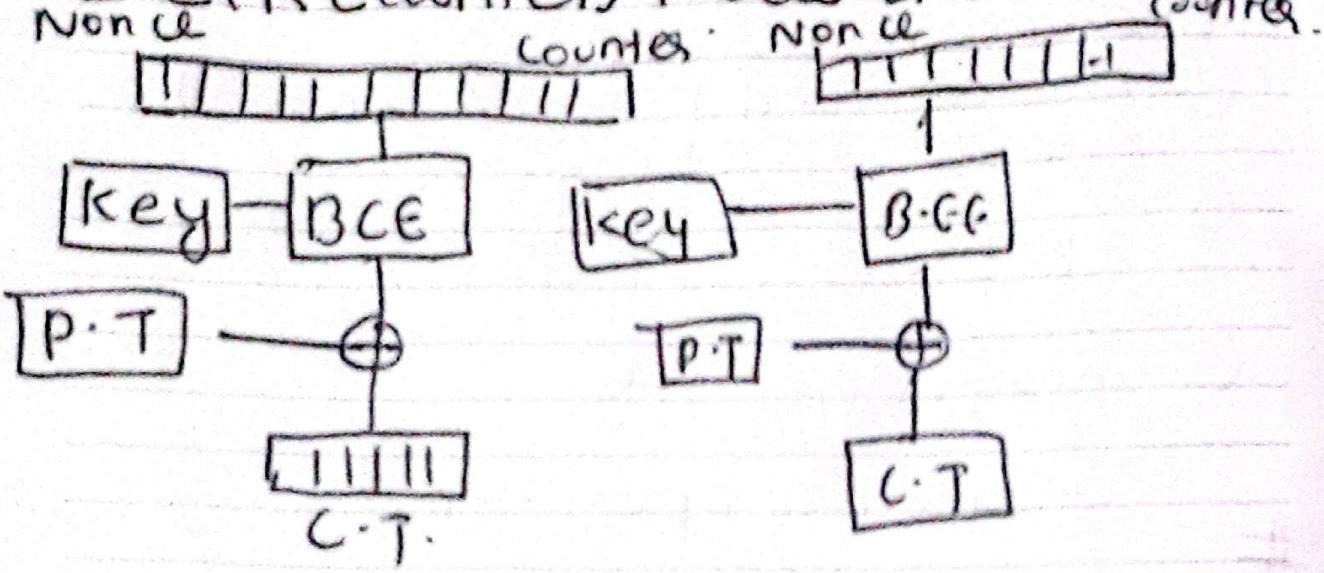


More than a College

Date

Su Mo Tu We Th Fr Sa

CTR (Counter) Mode Enc.



① AES in CTR mode generates keystream by encrypting counter (IV, nonce, etc).

② P.T is enc by XOR with keystream

$$C_i = P_i \oplus K_i$$

③ Decryption is same

$$P_i = C_i \oplus K_i$$

$$K = C \oplus P_{known}$$

once keystream is known

$$P = C \oplus K.$$

Date: CTR = If any plaintext bytes are known we can recover keystream.

Su Mo Tu We Th Fr Sa

Bean Counter (chall).

① Server implements CTR manually using AES-ECB.

keystream = AES.new(key, AES_ECB).encrypt(ctr.increment())

xored = P.T block XOR keystream.

② Counter is step up or step down, but for each encryption, you get deterministic keystream. to recover

So we have PNG and we know PNG always starts from same 16 byte header.

So, First 16 of C.T Bytes:

$K_0 = C_0 \oplus$ PNG HEADER

Then repeat keystream across file to recover full image.