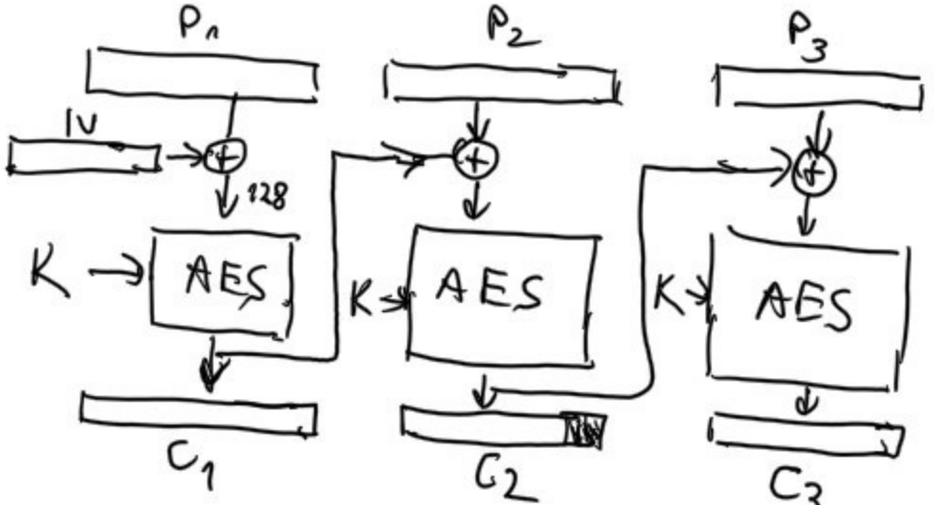
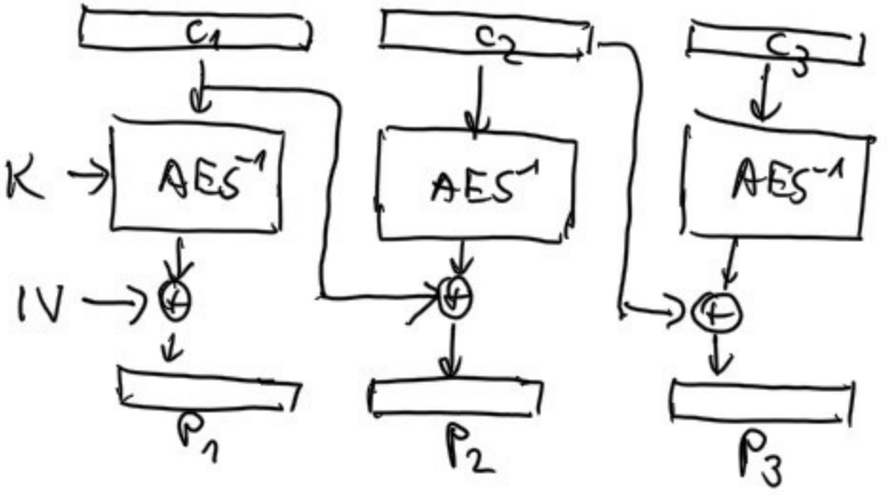
- No CBC in TLS 1.3



IV, C1, C2, C3 ← ciphertext

How to decrypt?



What happens if a plaintext is not a multiplicity of the block size of the encryption algorithm.

A padding is applied.

TLS 1.2 – PKCS padding is used

the last byte tells us how many padding bytes are added,

and pad + 1 bytes must contain this value.

| pad |

| 0x00 |

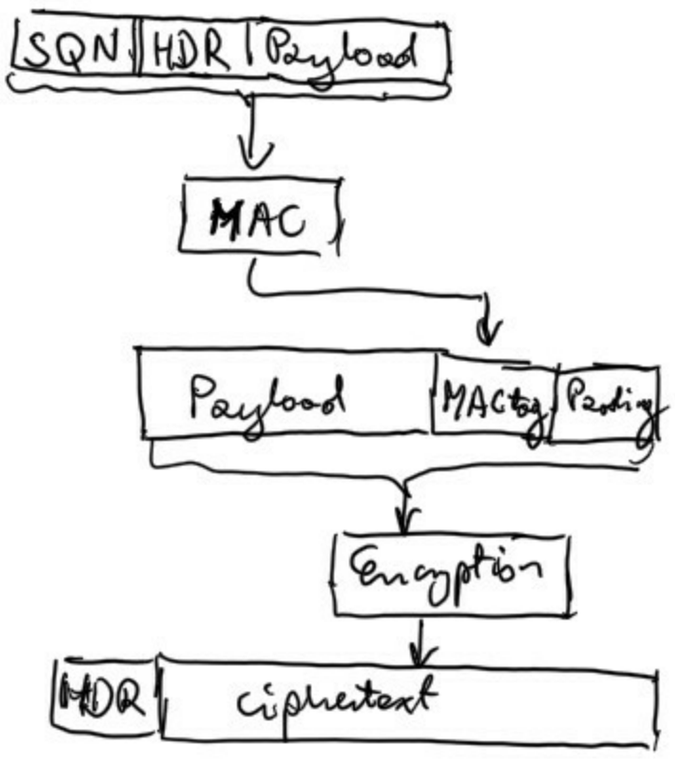
| 0x01 | 0x01 |

| 0x02 | 0x02 | 0x02 |

| 0xFF | 0xFF | ... | 0xFF |

256

TLS 1.2:



Lucky -13

Sha-1 → the output is 20 bytes

AES → single block - 16 bytes

Input to compression function of sha-1 is 512 bits.

Apart from that a weakness in implementation on the server side is needed (we perform a kind of timing attack).

MAC typically is computed on on 64-byte block (in case of sha-1).

Header takes 5 bytes

8 bytes for SQN,

and we are interested in the case that the padding byte is 0x00.

So we have 13-bytes. In a single ~~ciphertext composed of four AES blocks, which are of the plaintext.~~

How many bytes the payload has?

We know that we are able to distinguish between 55-bytes and 56-bytes

of the input to HMAC.

MAC uses 13 bytes of MDR (5) & SQN(8) ciphertext won't contain them, we have substract 55 - 13 = 42 bytes. - for the payload.

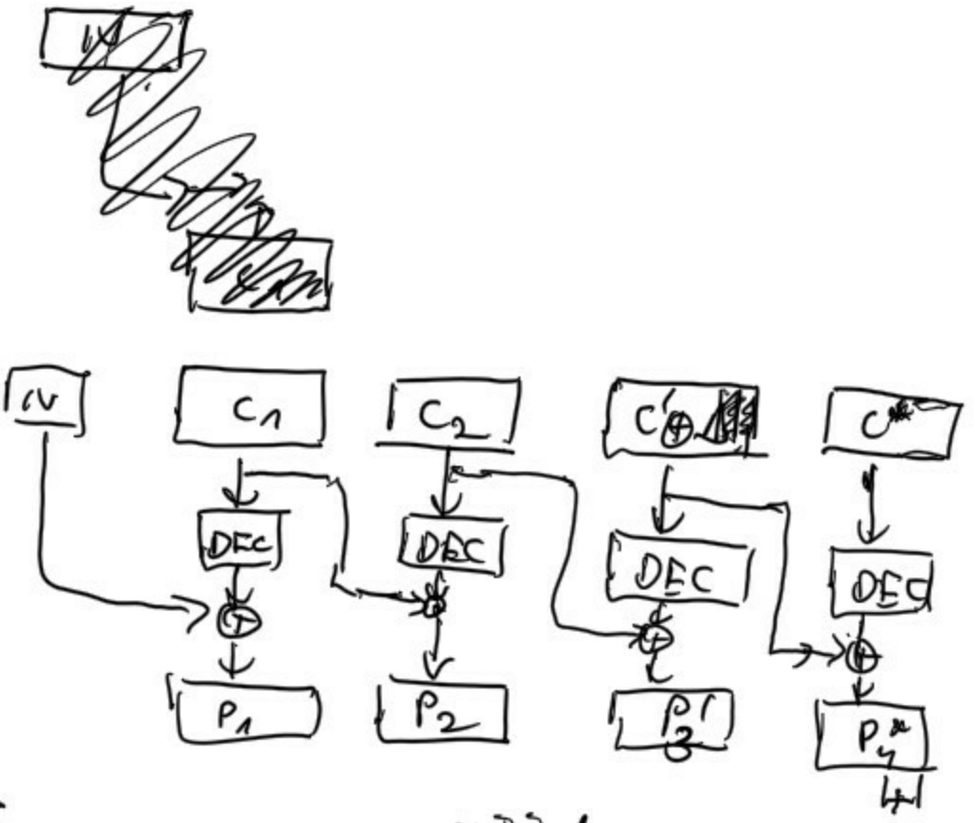
Now we add tag size : 20 bytes in case of SHA-1 ; 62 bytes, and 2 bytes of padding are needed for the attack.

So we assume that the attacked ciphertext is of the form:

$$C^{att}(\Delta) = MDR \,||\, C_0, C_1, C_2 \,||\, C' \oplus \Delta \,||\, C^*$$

C_0 = IV ; four blocks

Δ - 16 bytes that are added by MitM to the ciphertext.



The corresponding original plaintext is

$P = P_1 || P_2 || P_3 || P_4$ in which

$$P_4 = Dec(C^*) \oplus (C' \oplus \Delta) =$$

$$= P^* \oplus \Delta$$

Now we have 3 distinct possibilities for the timing oracle:

a) P_4 end with 0x00 byte

in this case we remove a single byte, hence payload is composed of:

$4 \cdot 16 - 21 = 64 - 21 = 43$,

but 42 bytes was our upper bound for single application of SHA-1 compression function. So 2 applications (longer execution) is needed.

b) P_4 ends with a valid pattern of lenght at least 2 bytes, then single application of SHA-1 compression function is needed if the padding is OK.

c) P_4 ends with any other pattern (the padding is incorrect) then the standard says that the plaintext should be treated as

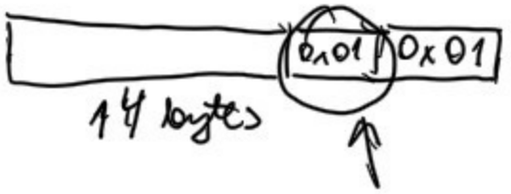
...	0x16	0x75

a plaintext with zero padding. (no padding is involved). So again we have two applications of SHA-1 compression function.

So we are going to recover the byte just before the padding:

	0x00

So we add $\Delta = rand \,||\, 0x01$ and we do so by brute-force attack. Finally we get the

$P^* =$ 

So we know Δ, we know the byte of $P^* \cdot (0x01)$, so we get that byte of P_4 recovered

$P_4 = P^* \oplus \Delta$.

So we assume that we have guessed the padding 0x01 0x01 in incorrect P^*

because probability is $\frac{1}{2^{16}}$,

in case of 0x02 0x02 0x02 we have probability of the success $= \frac{1}{2^{24}}$

In fact we don't have to have the correct padding 0x00. We may spoil MAC value by setting the leftmost byte of Δ to a non-zero value and in this way even if by manipulation of Δ we get 0x00 in P^* we do not distinguish this from the case c) which means that we wait longer for the error message from the server. 0x01 0x01

Having guessed the correct padding in non-genuine P^*, we get the last two bytes of P_4 and get the value of pad variable and pad+1 bytes of P_4 are recovered and we may extend the attack on next bytes

Other changes in TLS 1.3:

- key schedule
- 0-RTT mode (0-RTT attacks)
- smaller set of ciphersuites,
- handshake encrypted partially,
- DH in the first message from the client

TLS 1.3 faster than 1.2

- no renegotiation.
- client authentication in any time of the connection