Chair for Network Architectures and Services
Department of Informatics
TU München – Prof. Carle

Network Security

Chapter 13

Some More Secure Channel Issues

Outline

- In the course we have yet only seen catastrophic weaknesses where a stream cipher or a stream cipher mode was used.
- Goal
  - We want to show that this can also happen with block cipher modes.
  - This also helps to understand common recommendations for CTR mode.

- More on the order of MAC and encryption
- Goal
  - Give reasons why current research prefers the variant Encrypt-then-MAC.

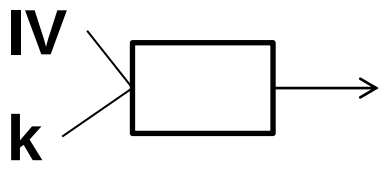
- What does a „Principle of…“ mean? How to understand? Follow?
- Goal
  - Encourage critical thinking

Attacks we have seen that utilize stream cipher properties

- Attacks we have seen that utilize stream cipher properties

Failures we have seen using stream ciphers/ modes

- Re-use of Initialization Vector (IV), e.g. in WEP or chapter 3



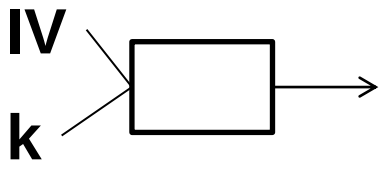
1 0 1 1 0 0 0 1 0 1 0 1 1 0 0 0 1 0 1

xor

P1 = 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 1 0 1 0

C1 = 1 0 0 1 0 0 1 1 1 1 0 1 1 0 1 1 1 1

Then some time later the same IV is used again:



1 0 1 1 0 0 0 1 0 1 0 1 1 0 0 0 1 0 1

xor

P2 = 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1

C2 = 0 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 1 1 0

Failures we have seen using stream ciphers/ modes

- Re-use of Initialization Vector (IV) continued

C1 = 1 0 0 1 0 0 1 1 1 1 1 0 1 1 0 1 1 1 1

C2 = 0 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 1 1 0

C1+C2 = 1 1 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1

= = = = = = = = = = = = = … → P1+P2=C1+C2

P1+P2 = 1 1 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1

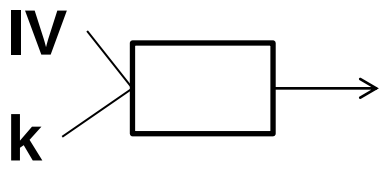
P1 = 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 1 0 1 0

P2 = 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 1

- As we see from the example, the attacker can computer C1+C2 because he observes C1 and C2, but that means he knows also P1+P2.
- Known Plaintext (e.g. P1) ➔ attacker can compute other plaintext
- Statistical properties of plaintext can be used if plaintext is not random-looking. That means if entropy of P1+P2 is low.

Failures we have seen using stream ciphers/ modes

- No integrity check or weak integrity check, e.g. CRC in WEP
  - To simplify example, we use the last bit as parity bit to check integrity.



1 0 1 1 0 0 0 1 0 1 0 1 1 0 0 0 1 0 1

xor

P =	**0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 1 0 1 0**	8 „1“s → 0
C =	**1 0 0 1 0 0 1 1 1 1 1 0 1 1 0 1 1 1 1**	
Δ =	**1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1**	
CΔ =	**0 1 0 1 0 0 1 1 1 0 1 0 1 1 0 1 1 1 0**	
PΔ =	**1 1 1 0 0 0 1 0 1 1 1 1 0 1 0 1 0 1 1**	11 „1“s → 1 → ok

- Attacker can target individual bits, plaintext and checksum are linear in ciphertext. Thus, checksum can be overcome and targeted edits in a text could be done (e.g. change price information).

Block cipher Modes

- The attacks from the previous slides would not have worked that way against a block cipher mode like CBC.
  - The re-use of an IV can give hints about identical first blocks, but plaintext cannot be calculated from it.
    - The plaintext is not linear in the ciphertext. Thus, such trivial attacks won‘t work.
  - Weak checksums cannot be attacked directly, since single individual bits cannot be controlled by an attacker modifying the cipher text, again due to the fact that the plaintext is sent through the encrpytion algorithm.

- However, the attacks resulted from severe usage errors and not from proper use.
  - Moreover, integrity is not the goal of encryption and, thus, the weak checksum algorithm should be blamed.
  - Block cipher modes can also fail when used badly.

MAC-then-Enc/Enc-then-MAC – design guidelines?

- **MAC-then-Enc/Enc-then-MAC – what do design guidelines / “principle of …” say?**

Kerckhoff‘s Principle (a good guideline)

- Kerckhoff’s Principle (short version): “A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.”

- Now, is this a true fact?
  - No, it is a *guideline for good design*, but not a universal truth.
  - The assumption is that you gain more from making the system design public and publicly scrutinized than from hiding a system design where flaws at first may be unknown to attackers, but overlooked by designers.
  - *Kerckhoff’s principle is widely accepted in cryptography.*

- Does all security technology follow this principle?
  - Well, it is about cryptography.
  - But philosophically, does a …. obey it? Firewall? NAT? IDS?
    - Some technologies are more an arms race between defender and attacker.

Horton Principle (is it a good guideline?)

- Horton principle: “Authenticate what you mean, not what you say”
- Typical conclusion: this means that the plaintext should be authenticated and not the ciphertext.
- So, in our secure channel, we should do MAC-then-Encrypt
  - Because then the MAC protects the plaintext
- Now, state-of-art in cryptography suggests that Encrypt-then-MAC is better.
  - Security proofs for Encrypt-then-MAC can be shown in more security models (~ succeed against a slightly stronger attacker)
  - Attacks (later)
- So, this guideline misleads us when we talk about the secure channel for data transmission.

Horton Principle (philosophic)

- Horton principle: “Authenticate what you mean, not what you say”

- But what is the meaning of a data transport channel?
  - Isn‘t it naive to think of the plaintext as what you mean?

- When we say the meaning is the data unit of the higher layer protocol, then it is the plaintext.
- When we say the meaning is the transport of bits, then we might also be able to think of the ciphertext as the meaning.
  - Logically it is not forbidden by the sentence that meaning and saying is the same.

- Whatever you might think about the philosophic question above, *the main message is that the mechanisms of one layer are not about meanings of other layers. Transport Layer has no semantics for application-specific data*.

Prerequisites for attacking CBC and MAC-then-Enc

- **Prerequisites for attacking CBC and MAC-then-Enc**

Guessing a secret (revisited)

- Passwords
  - N: size of alphabet (number of different characters)
  - L: length of password in characters
- Complexity of guessing a randomly-generated password / secret
  - The assumption is, we generate a password and then we test it.

  → O(N^L)
- Complexity of guessing a randomly-generated password character by character
  - The assumption is that we can check each character individually for correctness.
  - For each character it is N/2 (avg) and N (worst case)
  - So, overall L*N/2 (avg)
- In the subsequent slides we will show an attack that reduces the decryption of a blockcipher in CBC mode to byte-wise decryption (under special assumptions).

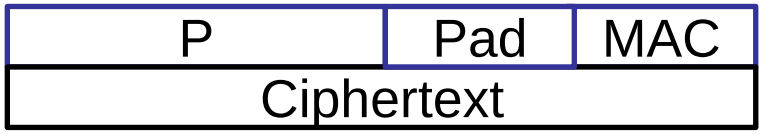
MAC-then-Encrypt Issues

- Operation
  - P and MAC are encrypted and hidden in the ciphertext.
  - Receiver
    - Decrypts P
    - Decrypts MAC
    - Computes and checks MAC →MAC error or success
- Consequence
  - MAC does not protect the ciphertext.
  - Integrity check can only be done once everything is decrypted.
  - As a consequence, receiver will detect malicious messages at the end of the secure channel processing and not earlier.
  - But is that more than a performance issue? Well, yes.

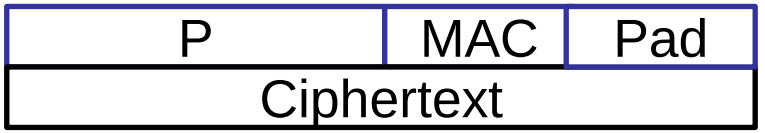
MAC-then-Encode-then-Encrypt

- If we use a block cipher, we have to ensure that the message encoding fits to the blocksize of the cipher.

- Encode-then-MAC-then-Encrypt:



  - Format P so that with the MAC added the encryption sees the right size.
  - Needs that we know the size of the MAC and blocksize of cipher when generating P | Padding.
- MAC-then-Encode-then-Encrypt



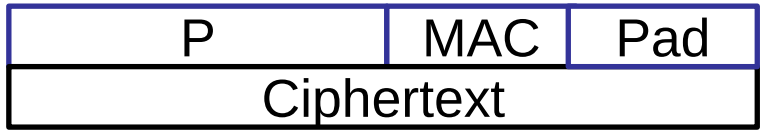
  - Used in TLS/SSL
  - Here, we add the MAC first and then pad the P | MAC to the correct size.
  - How do we know what is padding and what not? Padding in TLS/SSL:
    - If size of padding is 1 byte, the padding is 1.
    - If size of padding is 2 bytes, the padding is 2 2.
    - If size of padding is 3 bytes, the padding is 3 3 3.
    - ….

Padding Oracle Attack against CBC and MAC-then-Enc

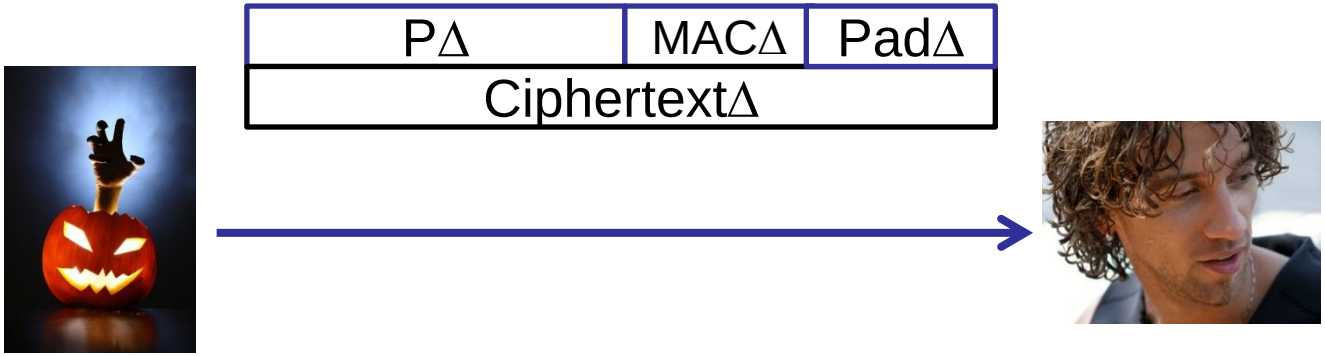
- **Padding Oracle Attack against CBC mode and MAC-then-Enc**

Concept of Padding Oracle Attack (against CBC)

- Attacker sees unknown ciphertext C = that was sent from Alice to Bob



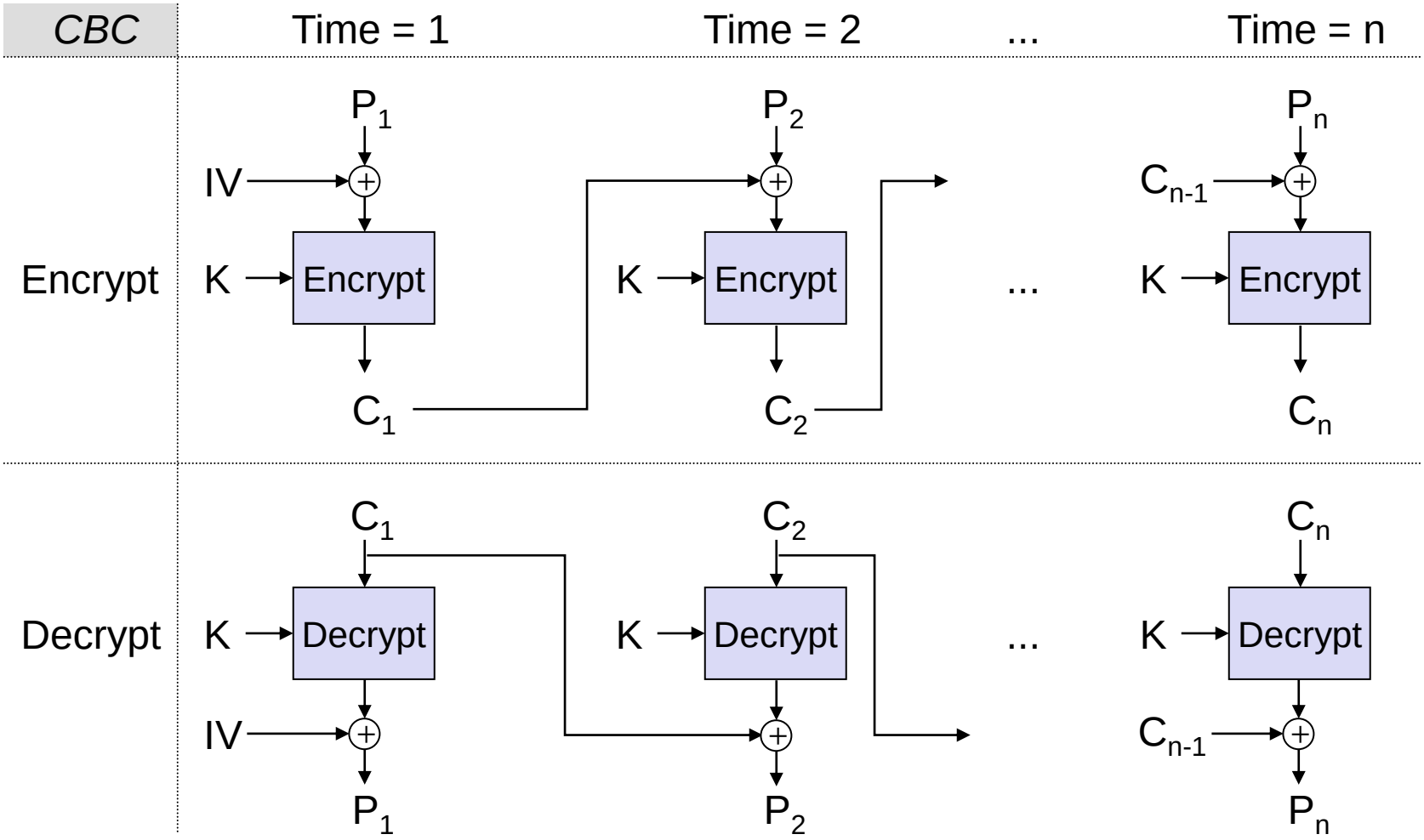
- To decrypt the ciphertext, the attacker modifies C and sends it to Bob.



- It is unlikely that the MAC and padding are correct. So, Bob will send an error back to Alice (and the attacker).
- In earlier versions of TLS, Bob sent back different error messages for padding errors and for MAC errors.

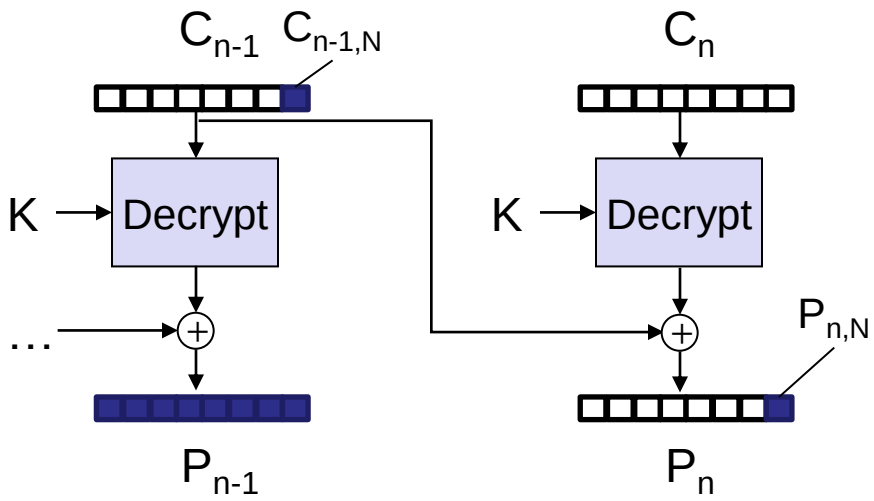
Padding Oracle Attack – CBC mode decryption (revisited)

- Encryption and Decryption in CBC mode



Padding Oracle Attack against CBC

- We have n blocks and N bytes per block. The attacker first wants to decrypt the last block C_n.
- In order to do so, he starts with the last byte $C_{n-1,N}$ of the block C_{n-1}. If he changes this byte (blue bytes are changed bytes)



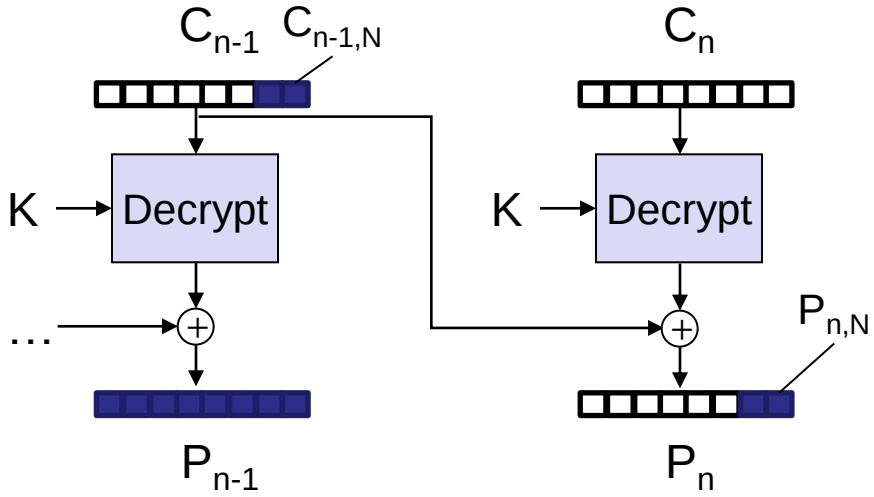
  - the MAC will most likely be invalid (chance 1 in 2^m for MAC length m)
  - the padding will be invalid unless $C_{n-1,N}$ xor $P_{n,N}$= 1 (chance 1 in 256)

➔ After testing the 256 values for **C**$_{n-1,N}$ all of them produced padding errors except for one that matches $C_{n-1,N}$ xor $P_{n,N}$= 1.

➔ We know $P_{n,N}$.

Padding Oracle Attack against CBC (2)

- Now, the byte $P_{n,N-1}$. For that we produce a padding of length 2.
- Since we know $P_{n,N}$ we can calculate $C_{n-1,N}$ so that $C_{n-1,N}$ xor $P_{n,N}$= 2
- Now, we have to find the $C_{n-1,N-1}$ that satisfies $C_{n-1,N-1}$ xor $P_{n,N-1}$= 2

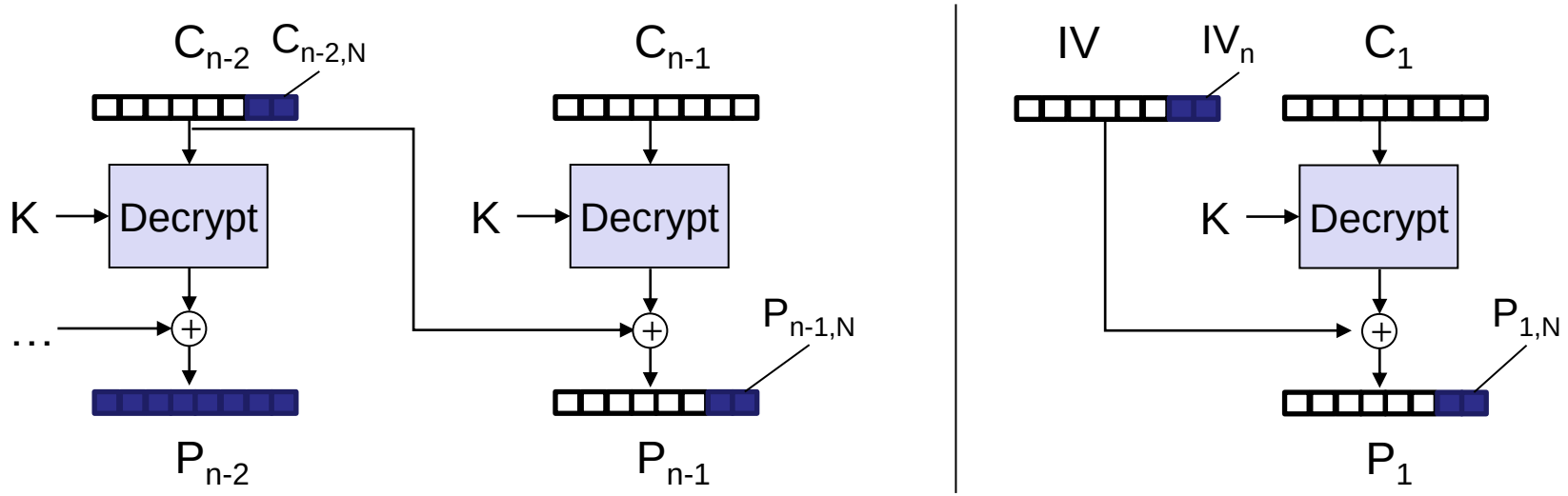


- With the same argument as before, we need to try up to 256 values, all values except for the correct one will generate a padding error. The correct one will produce a MAC errror.

  ➔ We know $P_{n,N-1}$.

Padding Oracle Attack against CBC (3)

- To completely decrypt C_n we have to repeat the procedure until all bytes of the block are decrypted. In the figure with 8 bytes per block, the last padding we generate is 8 8 8 8 8 8 8 8.
- To decrypt C_{n-1} we can cut off C_n and repeat the same procedure with C_{n-1} as last block. For decrypting C_1 we can use the IV as ciphertext for the attack modifications.



Padding Oracle Attack against CBC (4)

- We have seen a severe known-ciphertext attack against block ciphers in CBC mode.
  - Complexity is in the order of the length of message.
- Due to that attack, TLS/SSL stopped to differentiate between these two kinds of errors.
  - However, timing can still be an issue in case of local attackers as the padding error occurs a bit earlier (order of 1 ms earlier according to literature).
- Limitations: Alice and Bob might also start a rekeying due to these attack messages and then the oracle attack would not be possible as the key changed.
- The attack would not have been possible with (Encode-then-)Encypt-then-MAC as the MAC is checked first.
  - Protecting the ciphertext stops known-ciphertext attacks, which protecting the plaintext does not.