CBC PADDING ORACLE ATTACK

[BE2M32IBEA] Information Security – Gabriele Gatti

PADDING: why and how?

Why?

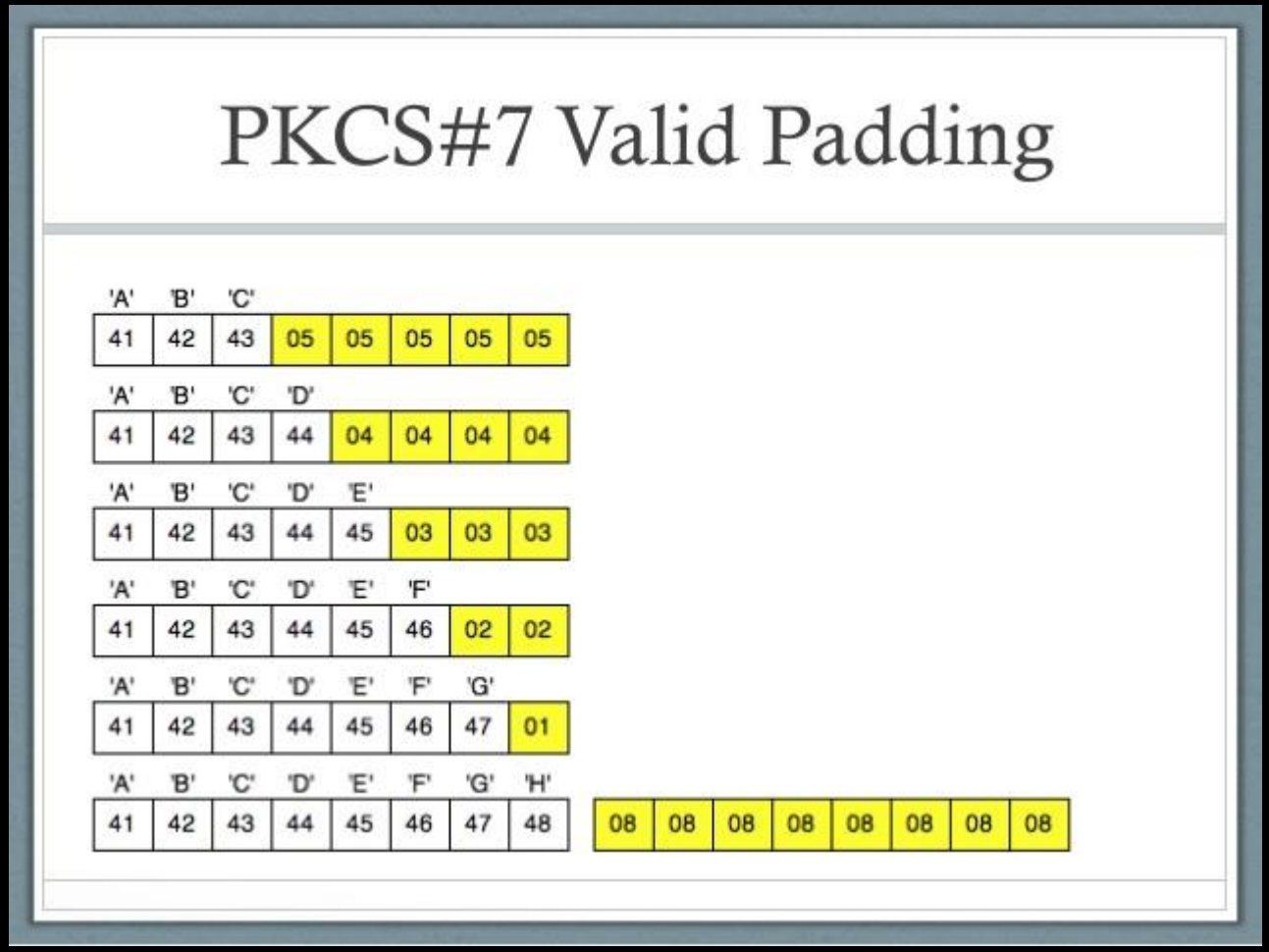
- Block ciphers need it to work
- Data in sizes multiple of a power of 2 are usually handled better
- Changes predictable starts/ends of messages that can facilitate cryptanalysis

How?

- Adding meaningless data to the original message to match required sizes
- Several standards are defined

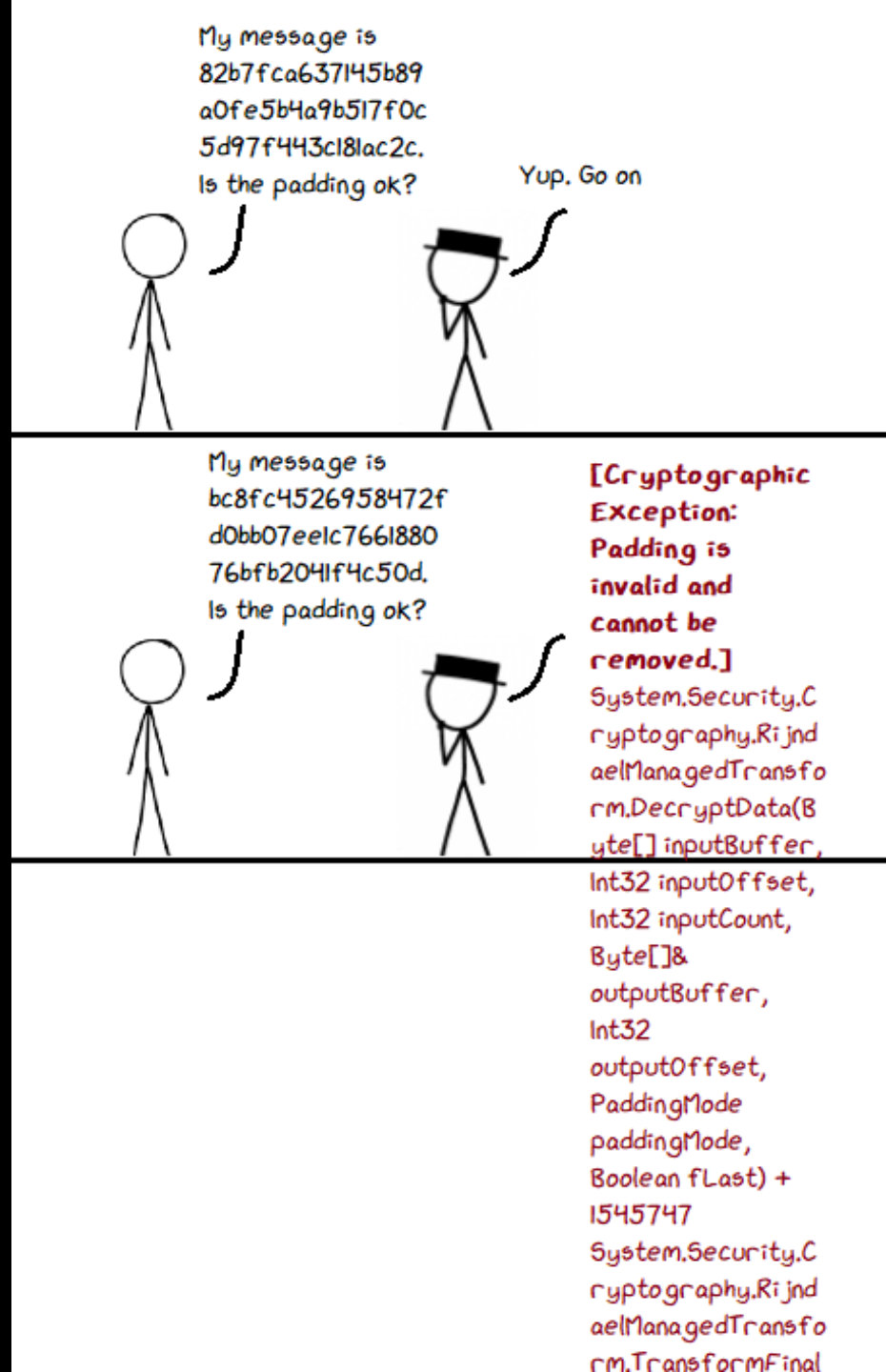
PADDING: PKCS#5 and PKCS#7

- Bytes are appended to the last block
- The value of the bytes is the total number of added bytes
- To remove padding we inspect the last byte of the last block and remove the same number of bytes from the end of the block



What is a PADDING ORACLE?

- Anything that provides us with information about padding correctness of a target **encrypted** message
- Usually a binary correct/wrong answer is easier to understand, but also timings can be exploited
- Exposes the system to side channel attacks!
  Information about padding can be transformed into information about the message!



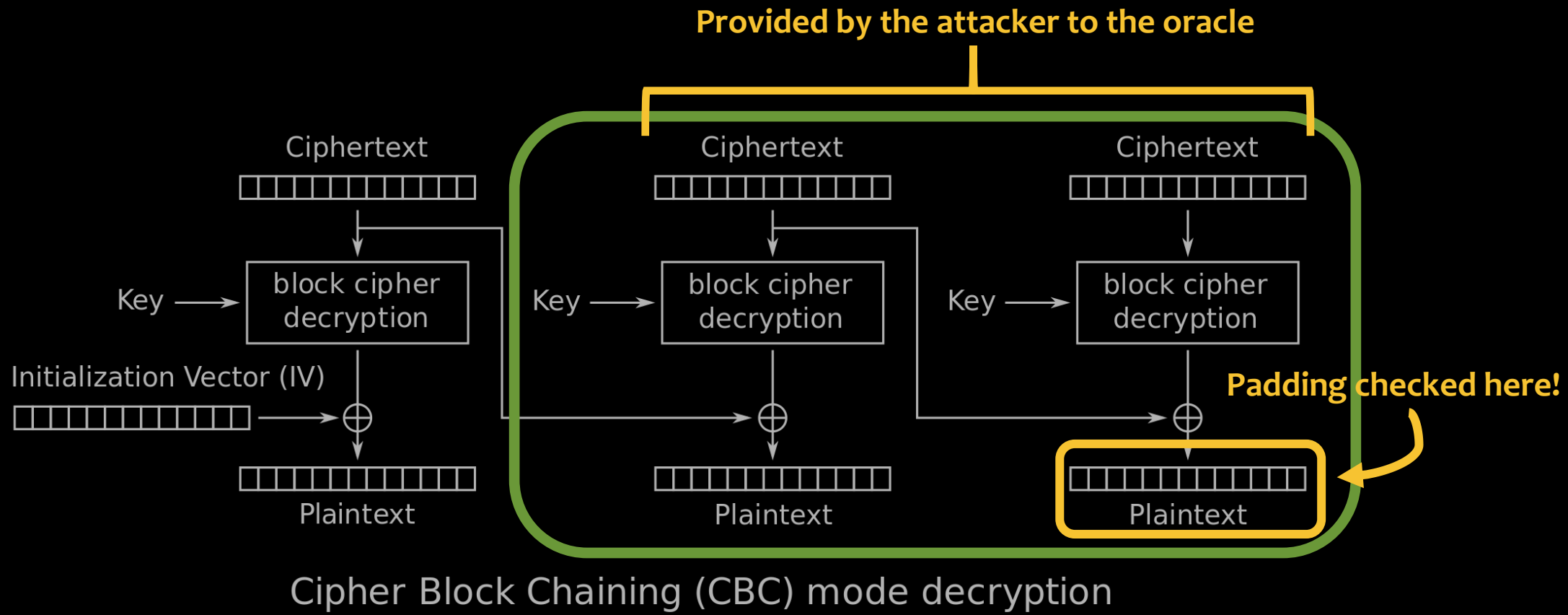
CBC PADDING ORACLE ATTACK

When the oracle answers it is basically telling us:

"The last bytes of the decrypted message do (not) correspond to the expected values"

Which sounds pretty innocent… ***as long as an attacker cannot control those last bytes!***

CBC mode refresher



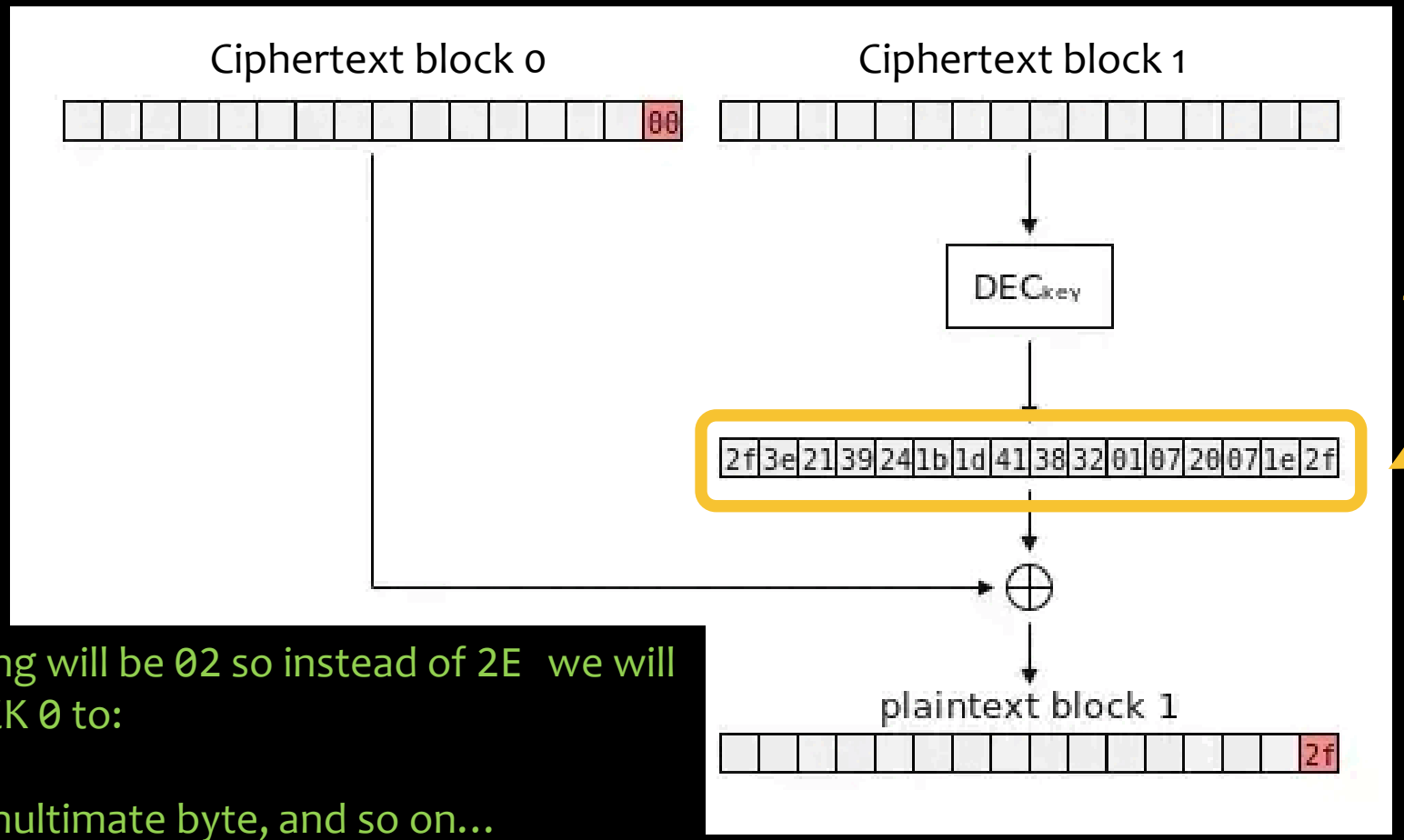
Cipher Block Chaining (CBC) mode decryption

When the oracle decrypts the last block, before checking the padding, the decryption output is **XORed with the previous block, which is provided by the attacker!**

CBC PADDING ORACLE ATTACK

Since the previous block is XORed byte by byte, and it is unmodified by the oracle, an attacker can build this block in a way to (brute)force the values of the padding!

Let's see an example:



For the next byte padding will be 02 so instead of 2E we will fix the last byte of BLOCK 0 to:

2F ⊕ 02 = **2D**

then brute force the penultimate byte, and so on…

CBC PADDING ORACLE ATTACK

What does this mean?

1. Starting from the last byte of the block we can recover each byte by forcing the correct padding for that position, to do so we try all possibilities until the oracle informs us about a correct padding. Inverting the XOR we obtain the decrypted byte

2. With a maximum of 256 trials for each byte of the block, we can recover the decryption output of our target ciphertext block.

3. XORing the obtained decryption output with the previous ciphertext block/IV gives us the plaintext block ☺

4. The process can be applied to all the ciphertext blocks, leading to full decryption in O(256*N) ≃ **O(N)**


LIVE DEMO! (hopefully)

Conclusions

Are encryption in CBC mode and padding broken???

NO, being a side channel attack the target is the unsafe implementation of a safe algorithm!

How can we avoid this?

General rule: always provide the users with the information they require, nothing more and nothing less (even error messages can be an oracle!)

Bibliography

- Padding image: https://stackoverflow.com/questions/34865313/bouncy-castle-pkcs7-padding
- Comic strip: my MS Paint skills & xkcd
- CBC mode image: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Cipher_block_chaining_(CBC)
- A better explanation (with animations): https://research.nccgroup.com/2021/02/17/cryptopals-exploiting-cbc-padding-oracles/Changes predictable starts/ends of messages that can facilitate cryptanalysis
- Demo and presentation available on my GitHub: https://github.com/gaaat98/cbc-padding-oracle-demo

Thank You!