



Padding Oracle Attacks



Agenda

- History
- Foundations
 - Block Ciphers
 - Modes of Operation
 - Padding
- Padding Oracle Attacks
 - Decrypting the last Byte
 - Decrypting the last Block
 - Decrypting all Blocks
 - Encrypting Data?
- Hands-On / CTF

History

History

1998, **Daniel Bleichenbacher:**

- CCA against RSA with PKCS #1 v1.5 Padding
- Mostly used to attack SSL/TLS
- Recent Threat: ROBOT

2001, **James Manger:**

- CCA against RSA with OAEP Padding
- Only works for certain implementations

2002, **Serge Vaudenay:**

- CCA against CBC Block Ciphers with PKCS #7 Padding

History

1998, **Daniel Bleichenbacher:**

- CCA against RSA with PKCS #1 v1.5 Padding
- Mostly used to attack SSL/TLS
- Recent Threat: ROBOT

2001, **James Manger:**

- CCA against RSA with OAEP Padding
- Only works for certain implementations

2002, **Serge Vaudenay:**

- Side Channel against CBC Block Ciphers with PKCS #7 Padding

History – Bleichenbacher Oracles

- First side channel attack that abused properties of padding
- Against RSA with PKCS #5 v1.5 Padding
- Originally as an CCA(2) against SSL 3.0

Note: CCA(2) = Adaptive Chosen-Ciphertext Attack

[...] an attacker first sends a number of ciphertexts to be decrypted chosen adaptively, then uses the results to distinguish a target ciphertext [...]

src: https://en.wikipedia.org/wiki/Adaptive_chosen-ciphertext_attack

History – Vaudenay Oracles

- Initial attack against CBC
- PKCS #7 aka RC5-CBC-PAD was commonly used for padding block ciphers
- After that, more research was conducted: Thai Duong & Juliano Rizzo
 - Decrypting ASP.net viewstates
 - Decrypting captchas
- SSL/TLS was not disregarded
 - POODLE
 - BEAST

Foundations

Foundations – Block Ciphers

- Underlying Block Cipher doesn't matter
 - Blocksize 8/16
 - DES , 3DES, AES
 -

Encryption: $E()$ takes a fixed length plaintext p and a secret key K as input to produce a ciphertext c of the same length, as the output. The encryption function is written like this: $c = E(K, p)$

Decryption: $D()$ takes a fixed length ciphertext c and a secret key K as input to produce the corresponding plaintext p as the output. The decryption function is written like this: $p = D(K, c)$

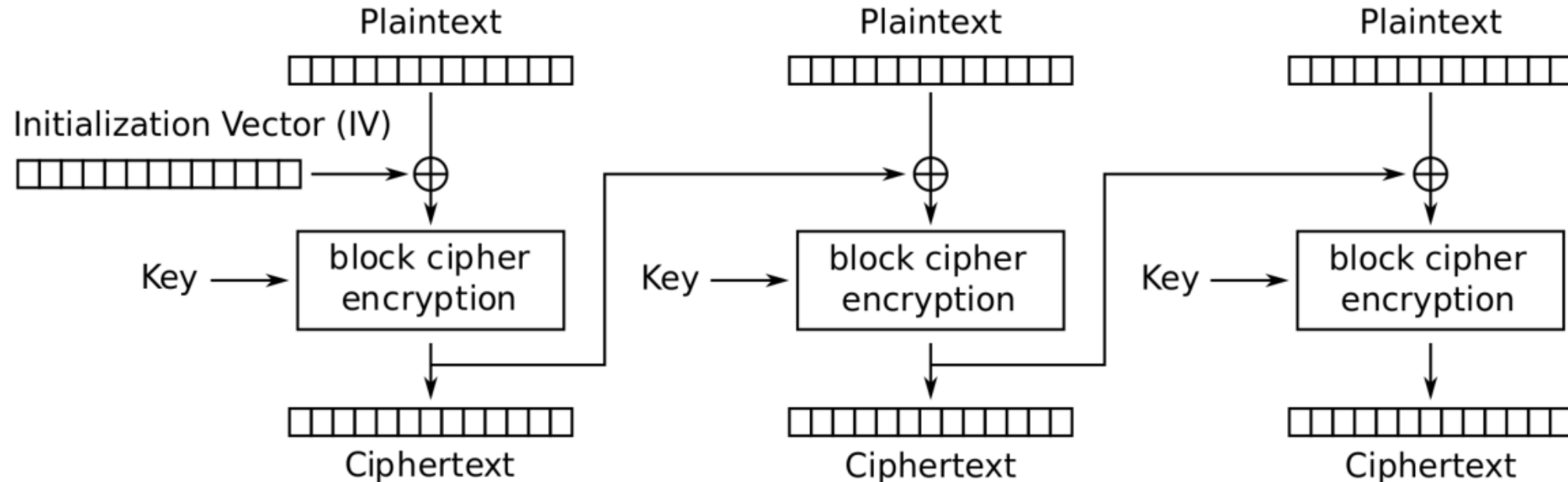
Foundations – Modes of Operation

- Many different modes: ECB (not really), CBC, CTR, ...
- For us important: **CBC**
 - Intention of CBC: Make ECB great again
 - Solve the problem of ECB that there shouldn't be two identical CT blocks
 - XOR is the 'solution'

Foundations – CBC

- **CBC Encryption:**

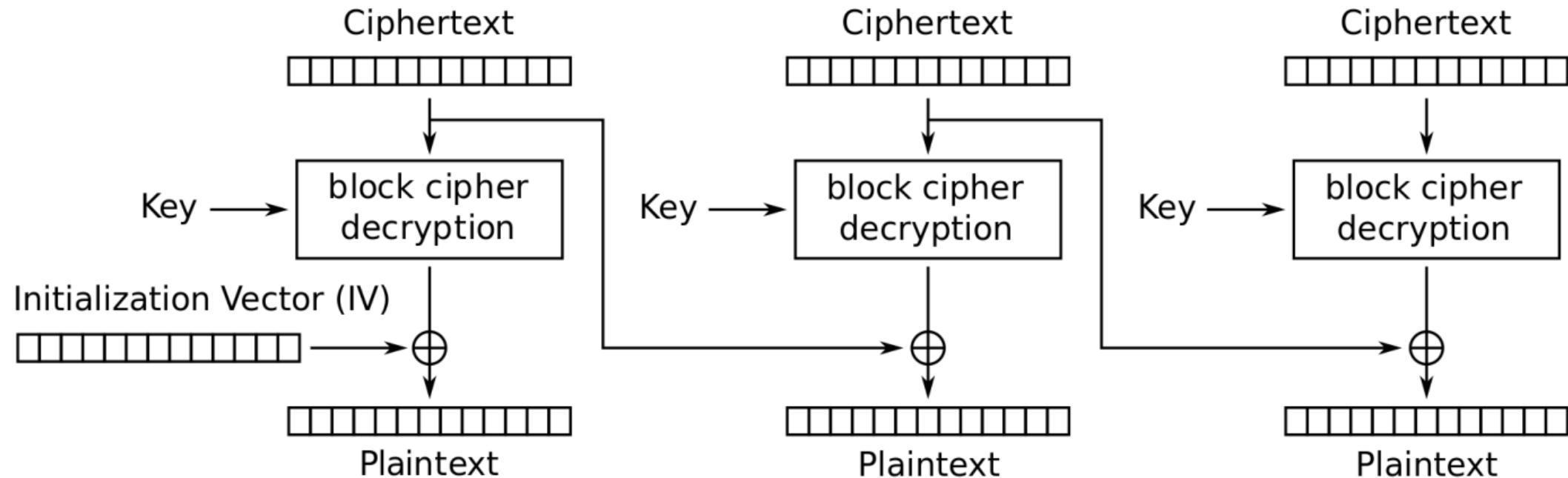
$$c_i = E(K, p_i \oplus c_{i-1}) \text{ for } i \in [1, \dots, b]$$



Foundations – CBC

- **CBC Decryption:**

$$p_i = D(K, c_i) \oplus c_{i-1} \text{ for } i \in [1, \dots, b]$$



Foundations – Padding

In general:

- Plaintext must be padded to a multiple of the blocksize
- Padding must be reversible

Important for us: PKCS #7 Padding

- PKCS #7 is byte oriented
 - max-blocksize: 256
- How: append the number of padding bytes n required to pad the block, each with the value n .

Foundations – Padding

Example: (Blocksize of 8 Bytes)

If we have 15 bytes of plaintext, there is one byte missing for the plaintext to be two complete blocks. This means we append the byte 0x01 to fill up the plaintext block to fit the blocksize of 8 bytes:

... | BB BB BB BB BB BB BB BB | BB BB BB BB BB BB BB **01**

We do the same thing with two, three or more bytes:

... | BB BB BB BB BB BB BB BB | BB BB BB BB BB BB BB **02 02**

...

... | BB BB BB BB BB BB BB BB | BB BB BB BB **04 04 04 04**

And if the length of the plaintext block is a multiple of the blocksize, we append a complete block of padding bytes:

... | BB BB BB BB BB BB BB BB | **08 08 08 08 08 08 08 08**

El Attackó

Padding Oracle Attacks - Approach

- The properties of padded messages can be abused as side channel
 - What we need for that: **Oracle!**

Padding Oracle Attacks - Approach

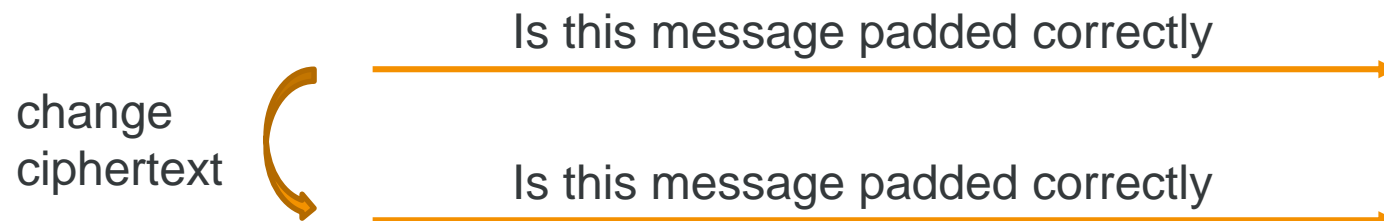
- The properties of padded messages can be abused as side channel
 - What we need for that: **Oracle!**

Is this message padded correctly



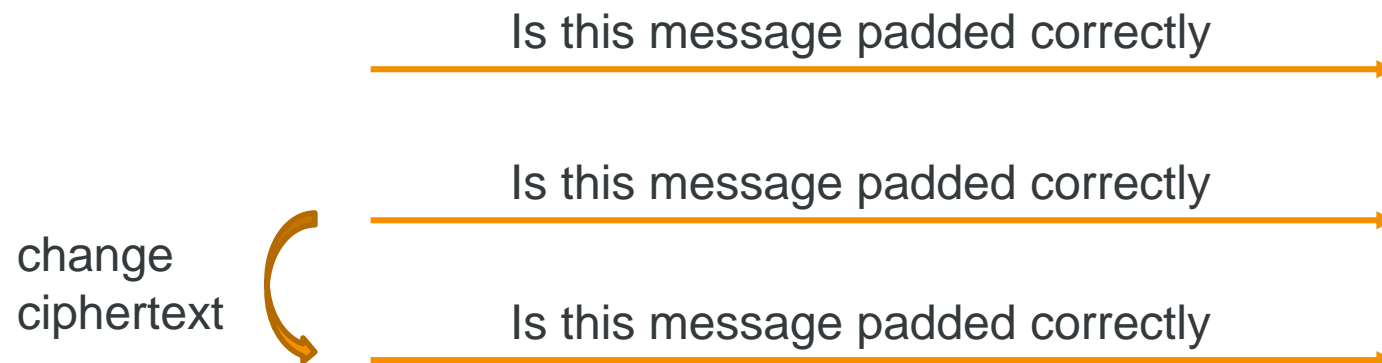
Padding Oracle Attacks - Approach

- The properties of padded messages can be abused as side channel
 - What we need for that: **Oracle!**



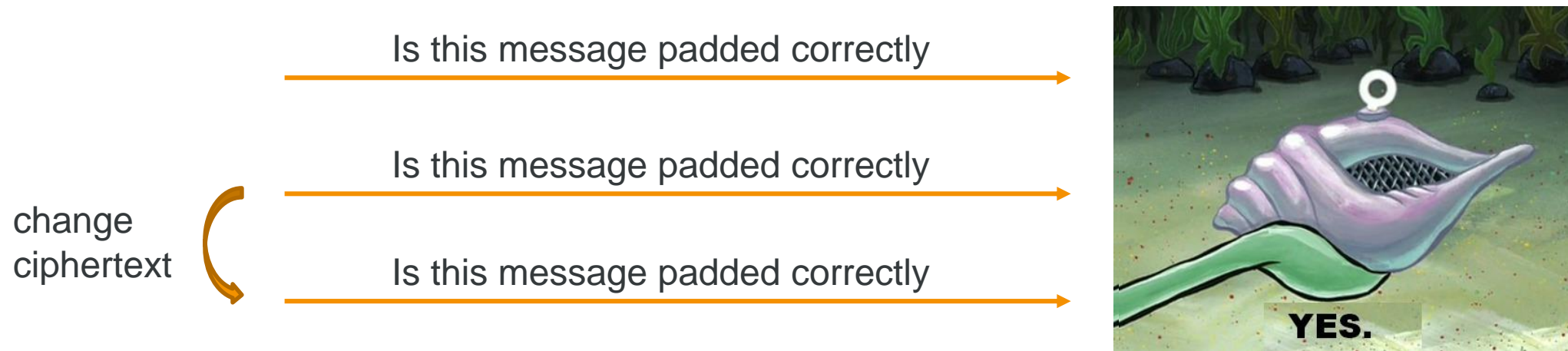
Padding Oracle Attacks - Approach

- The properties of padded messages can be abused as side channel
 - What we need for that: **Oracle!**



Padding Oracle Attacks - Approach

- The properties of padded messages can be abused as side channel
 - What we need for that: **Oracle!**



➔ This Difference + CBC can be abused to decrypt the ciphertext

Padding Oracle Attacks - Approach

- Scenario:
 - Client + (Web-) Server
- Web Server:
 - Stores session information in the cookie
 - Create Cookie:

```
cookie = aes_encrypt(pad(session_info))
```

- Received Cookie:

```
try:  
    session_info = unpad(aes_decrypt(cookie))  
except PaddingError:  
    [...]
```

Padding Oracle Attacks - Naming

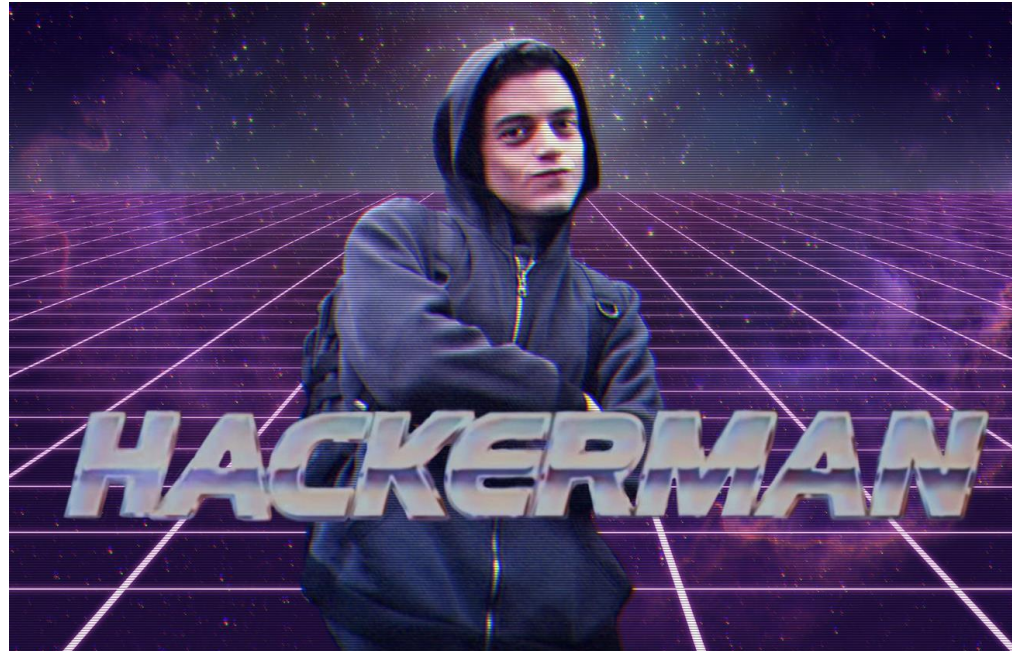
- Plaintext / Ciphertext: p & c
- Plaintext / Ciphertext block at index i : p_i & c_i
- Number of Blocks: b
- Byte at position x in block i : $p_i[x]$ / $c_i[x]$
- Encryption: $c = \text{Enc}(p)$
- Oracle: $\mathcal{O}(c) = [\text{true/false}]$ (true on valid padding)
- Blocksize: BS

Padding Oracle Attacks - Procedure

- 1) First, decrypt the last byte of the ciphertext using the oracle
- 2) Use the same approach to decrypt the last block
- 3) Because CBC: Use the same building block to decrypt all blocks

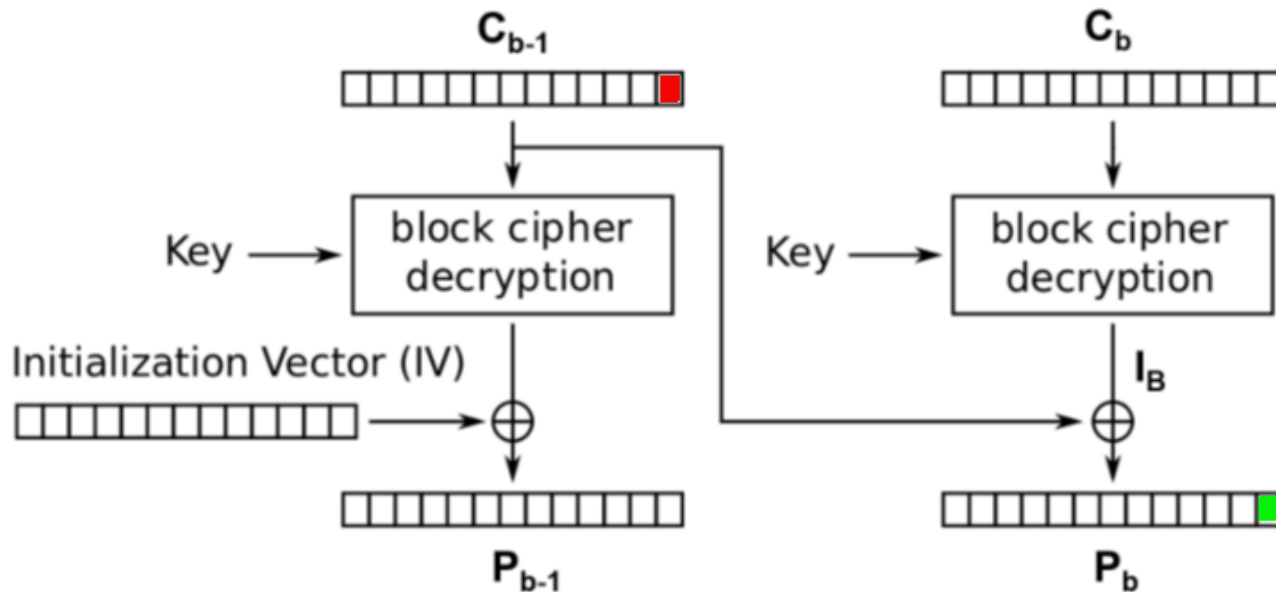
Padding Oracle Attacks - Procedure

- 1) First, decrypt the last byte of the ciphertext using the oracle
- 2) Use the same approach to decrypt the last block
- 3) Because CBC: Use the same building block to decrypt all blocks
- 4) Feel like hackerman



Padding Oracle Attacks – Decrypting the Last Byte

- Back to the basics: CBC

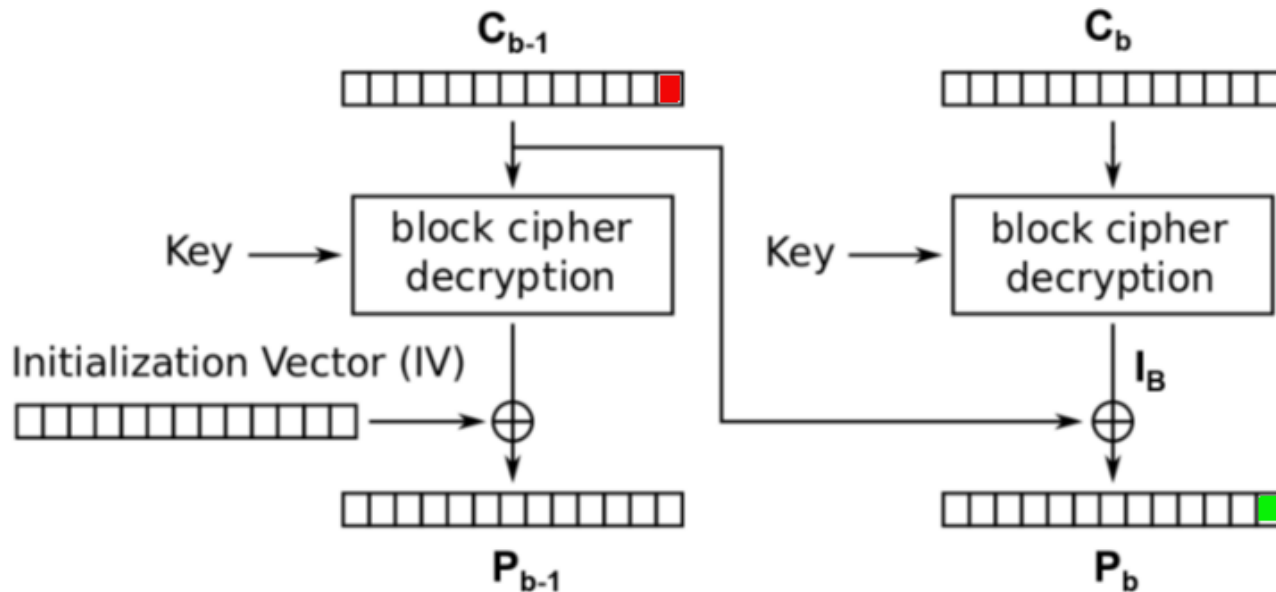


$$p_i = D(c_i) \oplus c_{i-1}$$

- To change the last plaintext byte $p_i[-1]$, the corresponding ciphertext $c_{b-1}[-1]$ must be changed.

Padding Oracle Attacks – Decrypting the Last Byte

- Back to the basics: CBC

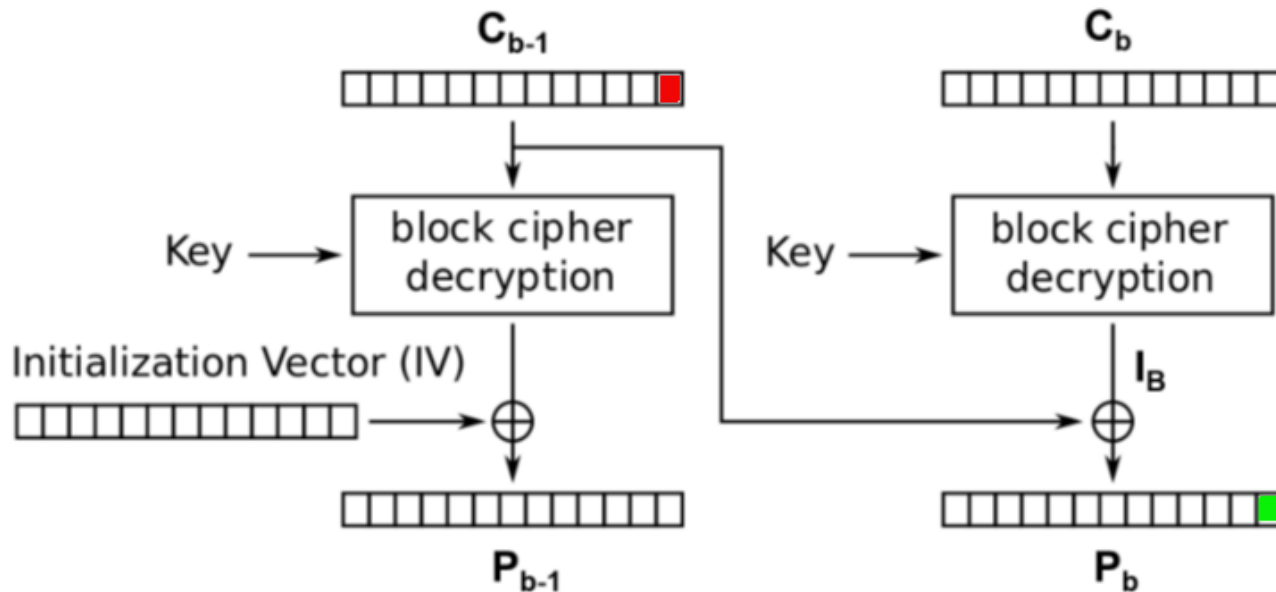


$$p_i = D(c_i) \oplus c_{i-1}$$

- To change the last plaintext byte $p_i[-1]$, the corresponding ciphertext $c_{b-1}[-1]$ must be changed.
- It must be changed in a way, that a valid padding arises: $O(c_{b-1}|c_b) = 1$
- A valid padding would be: e.g. **0x01**

Padding Oracle Attacks – Decrypting the Last Byte

- Back to the basics: CBC



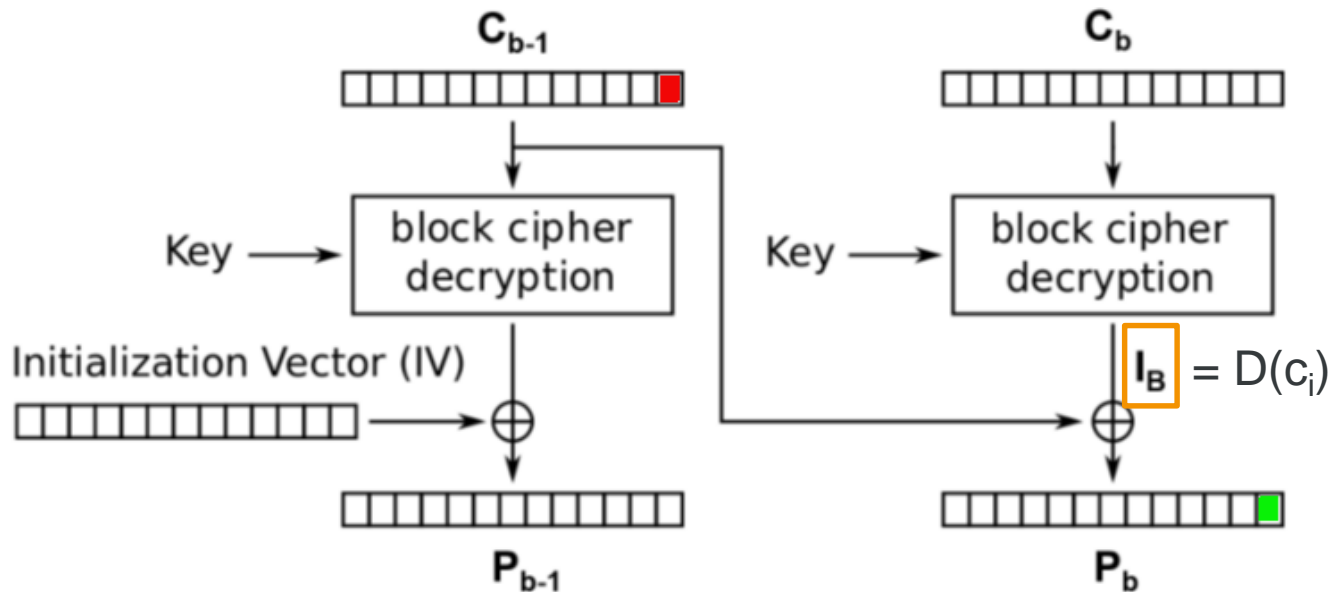
$$p_i = D(c_i) \oplus c_{i-1}$$

- To change the last plaintext byte $p_i[-1]$, the corresponding ciphertext $c_{b-1}[-1]$ must be changed.
- It must be changed in a way, that a valid padding arises: $O(c_{b-1}|c_b) = 1$
- A valid padding would be: e.g. **0x01**
- To achieve this: **guess!**

$$c_{b-1}[-1] = g ; 0 \leq g \leq 0xff$$

Padding Oracle Attacks – Decrypting the Last Byte

- Back to the basics: CBC



$$p_i = D(c_i) \oplus c_{i-1}$$

Some Quick Maths:

The plaintext can be calculated like this:

$$p_b = 0x01 \oplus g \oplus c_{b-1}$$

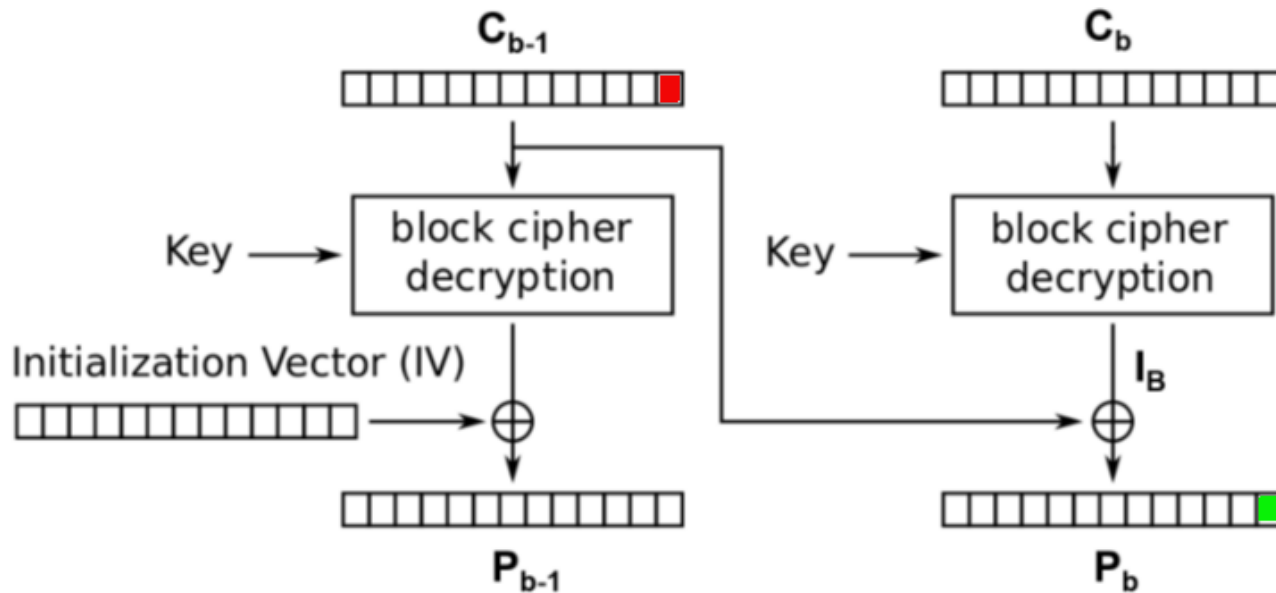
But why?

- We know that (Oracle): $g \oplus D(c_b) = 0x1$
- Same as: $D(c_b) = 0x1 \oplus g$
- Regular Decryption: $p_b = D(c_b) \oplus c_{b-1}$
- Replace $D(c_i)$: $p_b = 0x1 \oplus g \oplus c_{b-1}$

We call $D(c_i)$ the intermediate block I_i

Padding Oracle Attacks – Decrypting the Last Block

- Back to the basics: CBC

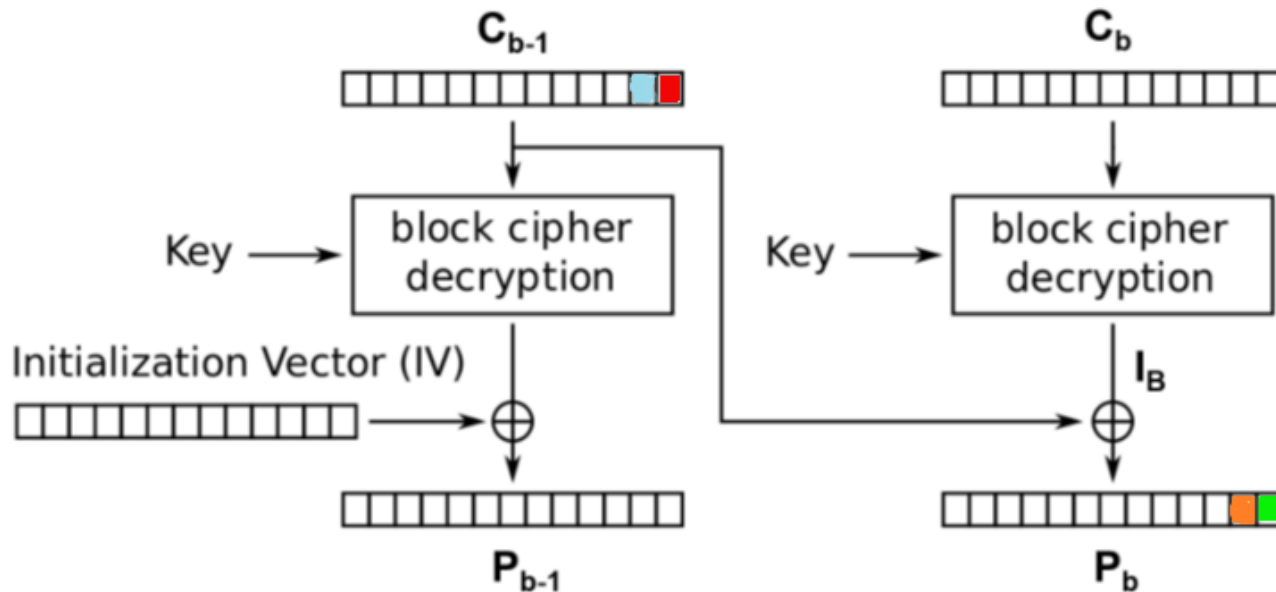


$$p_i = D(c_i) \oplus c_{i-1}$$

- Same approach can be used to decrypt the last block
- **But:** Padding must be adjusted!

Padding Oracle Attacks – Decrypting the Last Block

- Back to the basics: CBC



$$p_i = D(c_i) \oplus c_{i-1}$$

- Same approach can be used to decrypt the last block
- **But:** Padding must be adjusted!

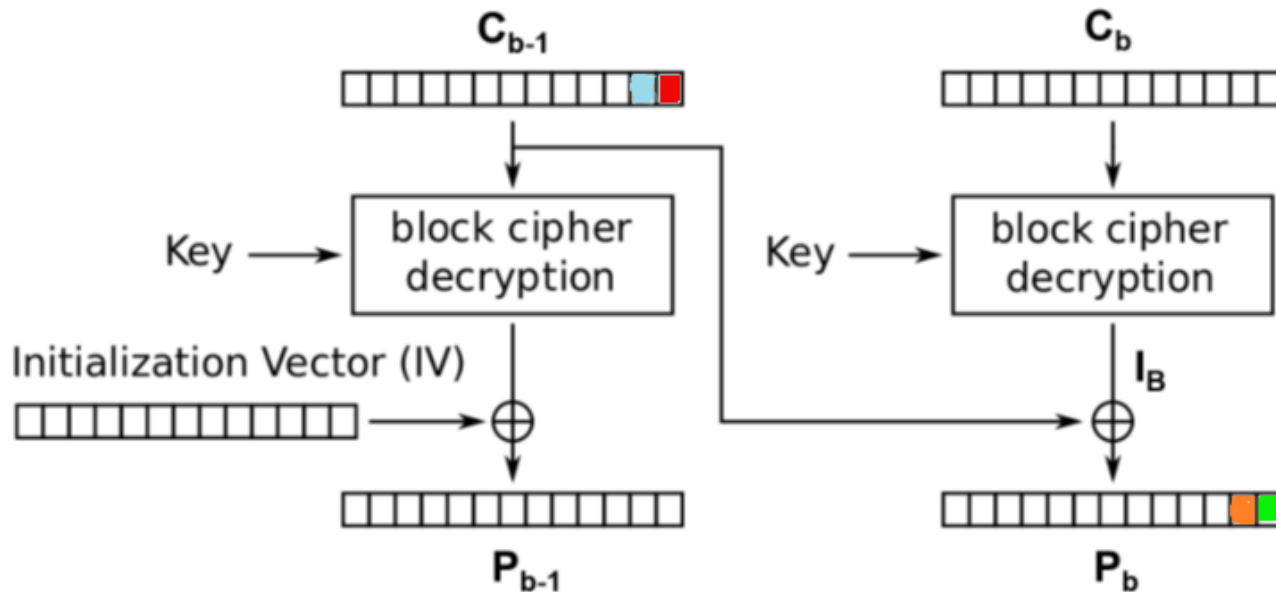
Step 1: Second to last byte $c_{b-1}[-2]$

$$p_b[-2] = c_{b-1}[-2] \oplus D(c_b[-2])$$

What would be a valid padding?

Padding Oracle Attacks – Decrypting the Last Block

- Back to the basics: CBC



$$p_i = D(c_i) \oplus c_{i-1}$$

- Same approach can be used to decrypt the last block
- **But:** Padding must be adjusted!

Step 1: Second to last byte $c_{b-1}[-2]$

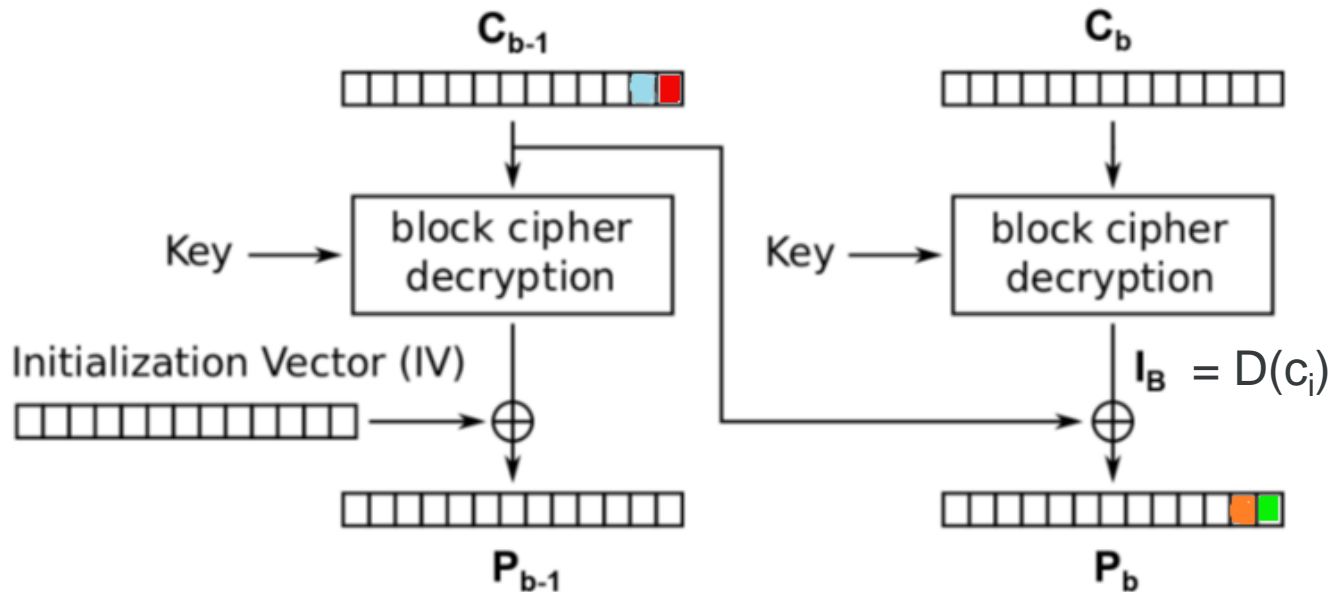
$$p_b[-2] = c_{b-1}[-2] \oplus D(c_b[-2])$$

What would be a valid padding? **0x02** & 0x01

Same procedure: guess $c_{b-1}[-2]$ until $O(c_{b-1}|c_b) = 1$

Padding Oracle Attacks – Decrypting the Last Block

- Back to the basics: CBC



$$p_i = D(c_i) \oplus c_{i-1}$$

Quick Maths again:

The plaintext can be calculated like this:

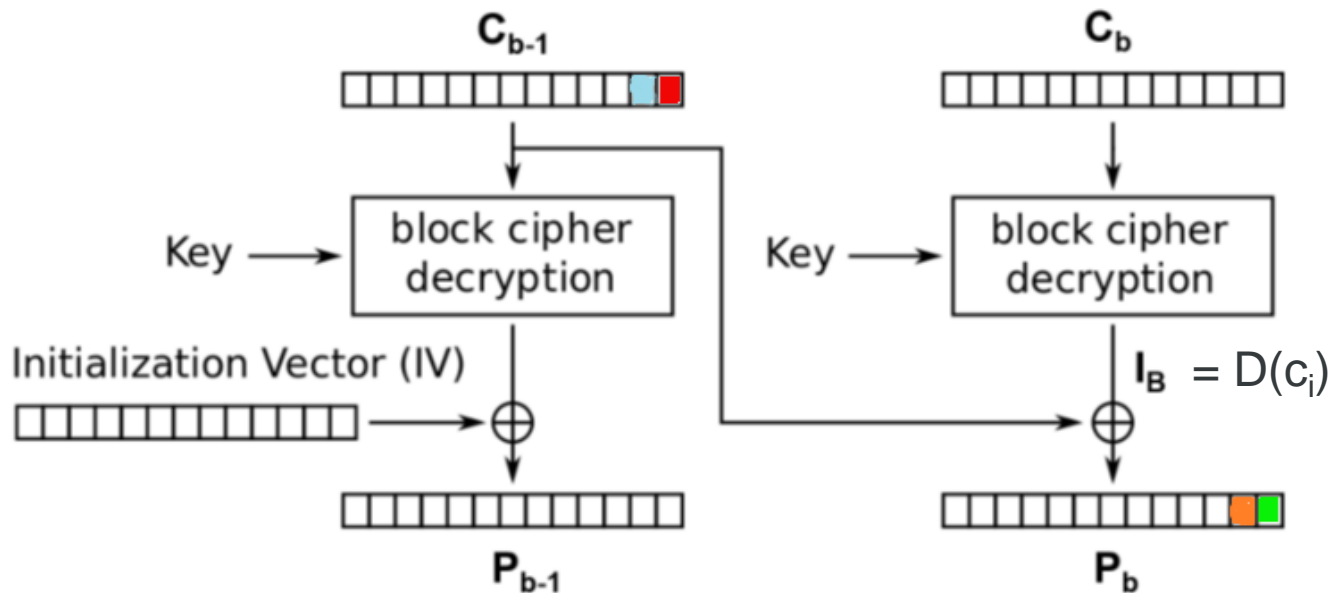
$$p_b[-2] = 0x02 \oplus g \oplus c_{b-1}[-2]$$

But why? You now that!

- One thing is missing:
- For a valid padding of 0x02, the last byte of the block ($c_{b-1}[-1]$) must be adjusted.
- Done by: $c_{b-1}[-1] = 0x02 \oplus D(c_b)$
 $= 0x02 \oplus I_B(c_b)$


Padding Oracle Attacks – Decrypting the Last Block

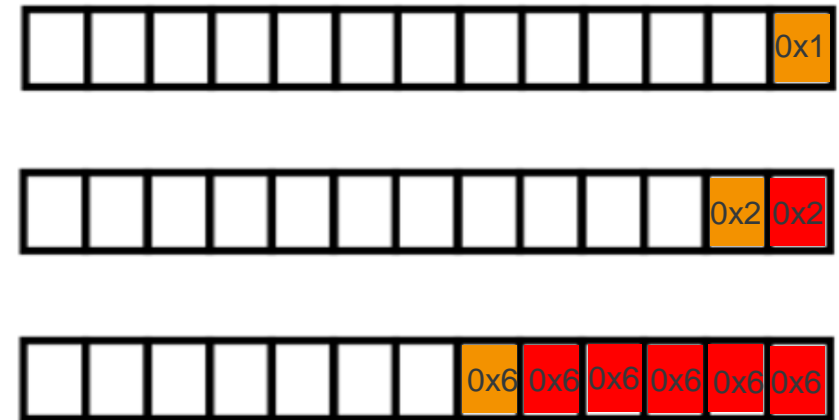
- Back to the basics: CBC



$$p_i = D(c_i) \oplus c_{i-1}$$

And so on...

 Decrypting  Adjust Padding



Vaudenay calls this “Block Decryption Oracle (BDO)”

Padding Oracle Attacks – Decrypting all Blocks

- To fully decrypt a ciphertext ($c = c_1 | c_2 | \dots | c_b$) perform the BDO for every block!
- After every block, cut off the already decrypted and repeat the process
- Pseudo Code:

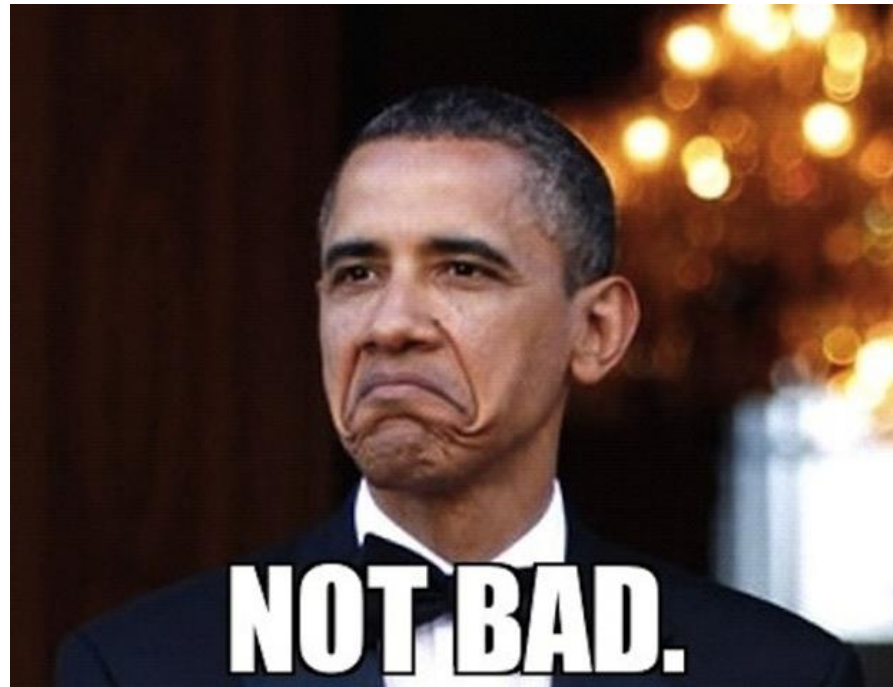
1 for i in $b .. 1$

2 $BDO(c_i)$

3 $c = c_1 | \dots | c_{i-1}$

Padding Oracle Attacks – Encrypting Data

- Guess what? Its possible to re-encrypt custom data using a padding oracle!



Padding Oracle Attacks – Encrypting Data

- Two possibilities to deliver ciphertexts:
 - <ciphertext>
 - <iv><ciphertext>

Padding Oracle Attacks – Encrypting Data

- Two possibilities to deliver ciphertexts:
 - <ciphertext>
 - <iv><ciphertext>

To change data, the ciphertext block **before** the changed block must be adjusted!

➔ Because the IV contains random bytes, a change in the IV has no semantic consequences.

Padding Oracle Attacks – Encrypting Data

1) Changing **one** block

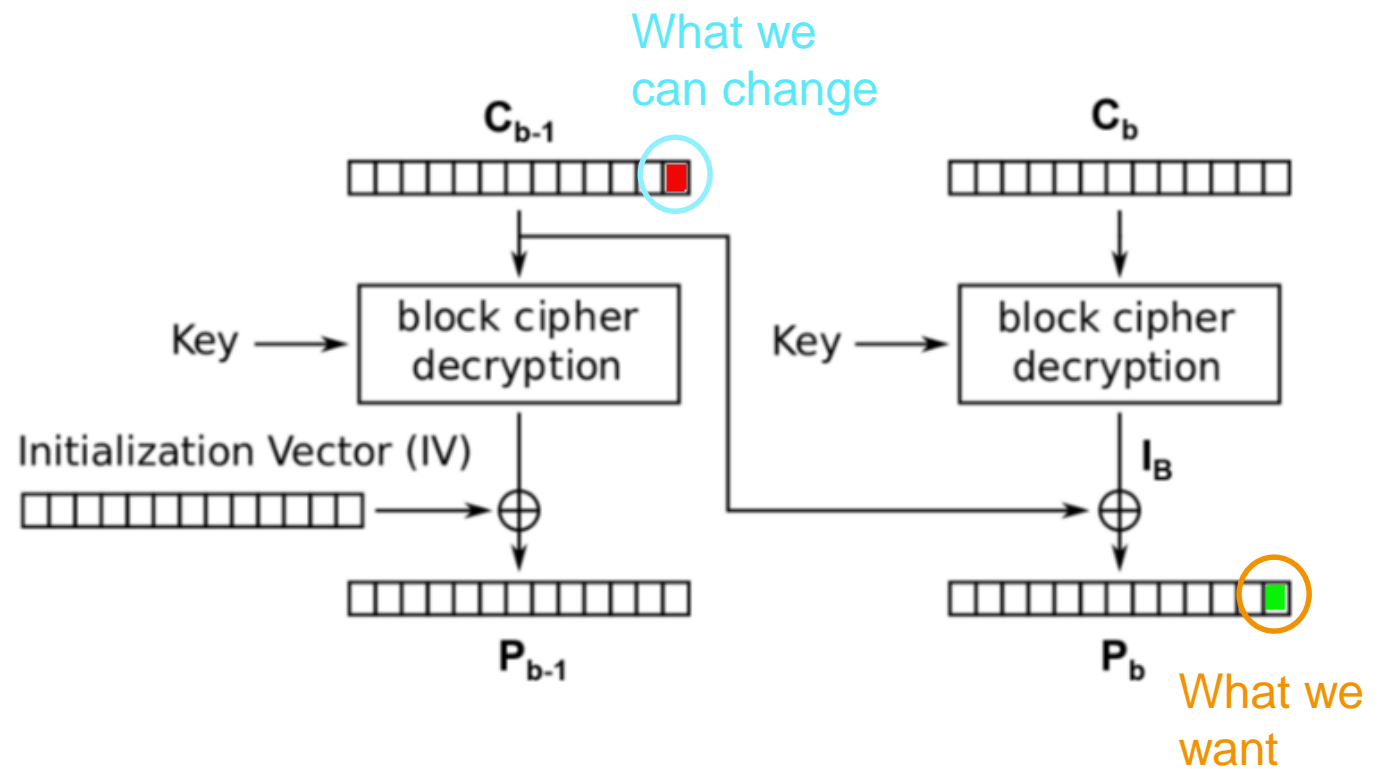
Z.B.:

$$P_i = I_i \oplus C_{i-1}$$

$$0x41 = I_i \oplus C_{i-1}$$

$$C_{i-1} = I_i \oplus 0x41$$

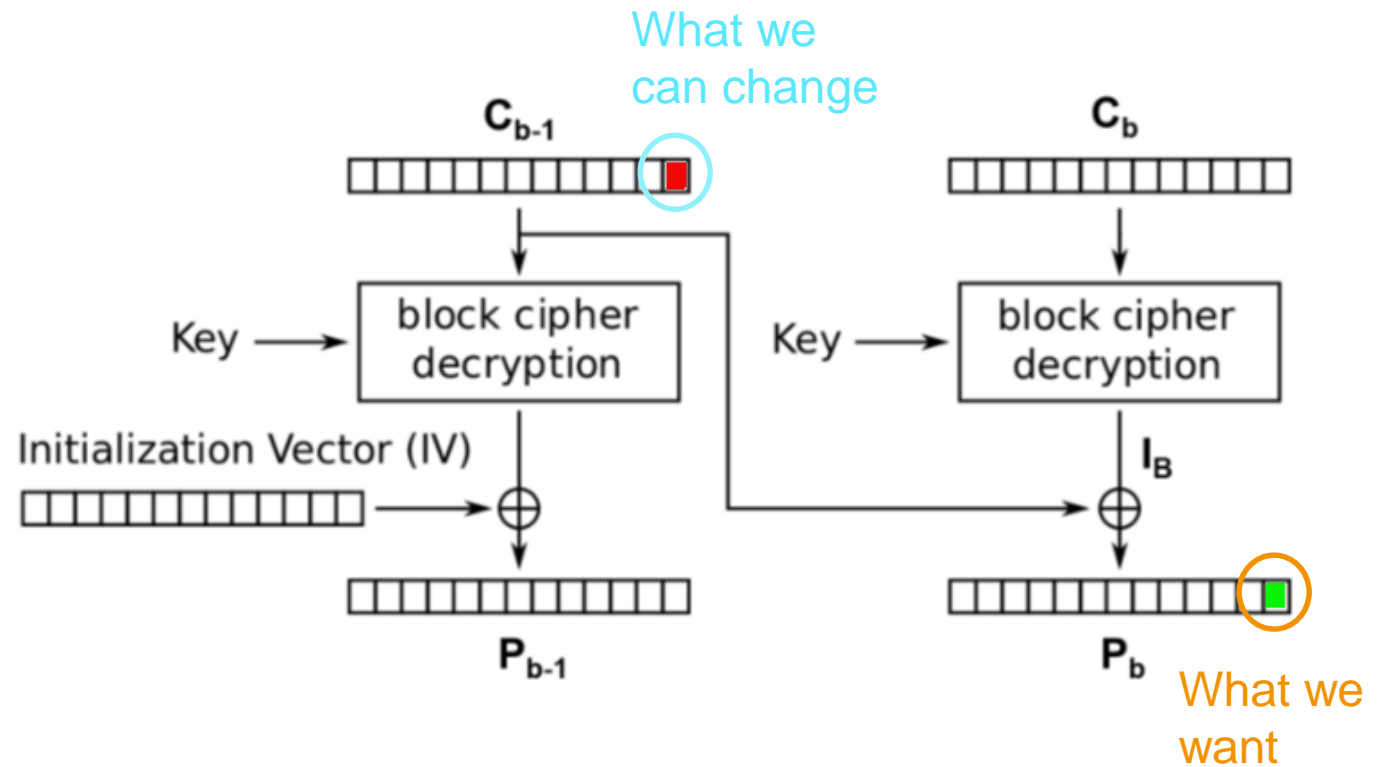
- We need to change ciphertext to encrypt custom data
- Meaning: the changed ciphertext decrypts to **garbage**



Padding Oracle Attacks – Encrypting Data

2) How to avoid garbage?

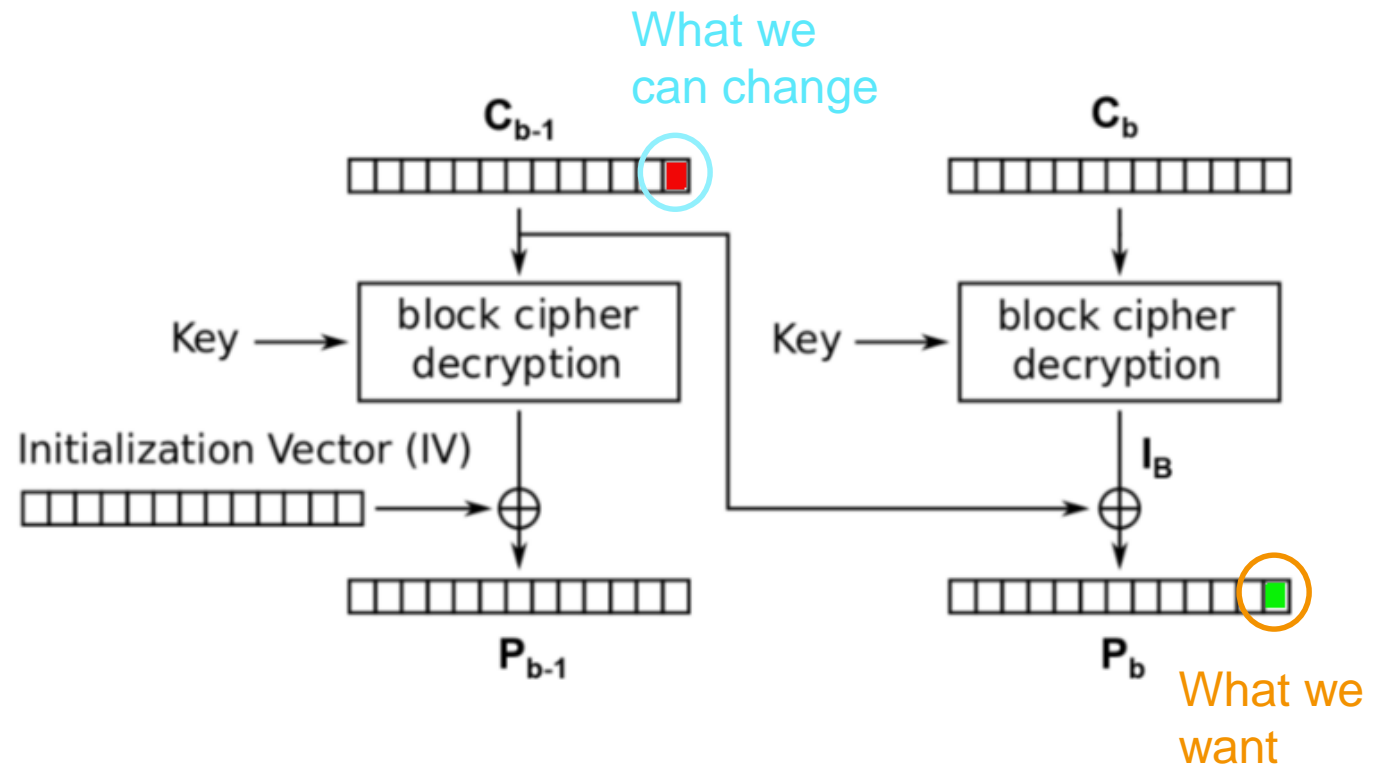
Adjust the already *changed* ciphertext block again, by changing the previous ciphertext block.



Padding Oracle Attacks – Encrypting Data

2) How to avoid garbage?

Adjust the already *changed* ciphertext block again, by changing the previous ciphertext block.



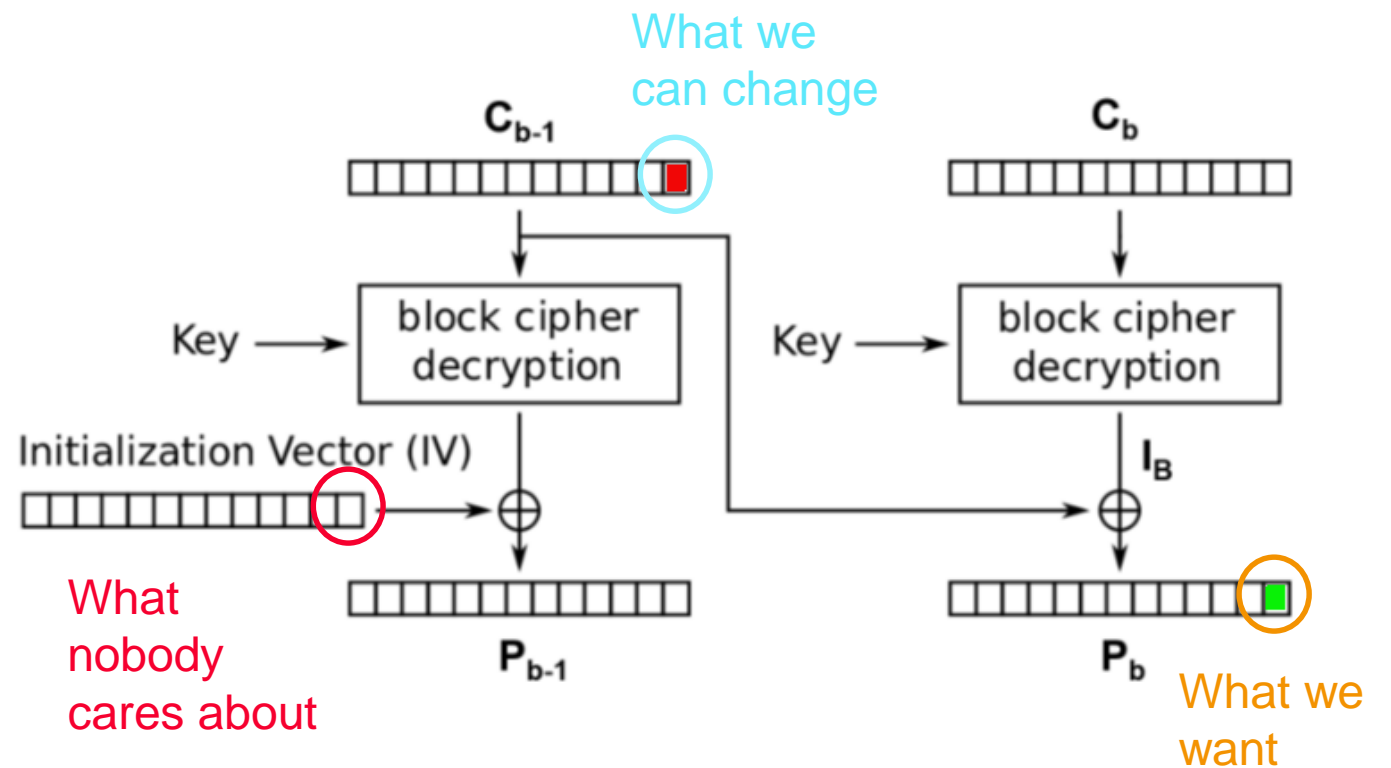
Padding Oracle Attacks – Encrypting Data

This can be done until we reach the **first** block.

If we are lucky, the first block is the **IV**.

No one cares about the IV

~_(\ツ)_/



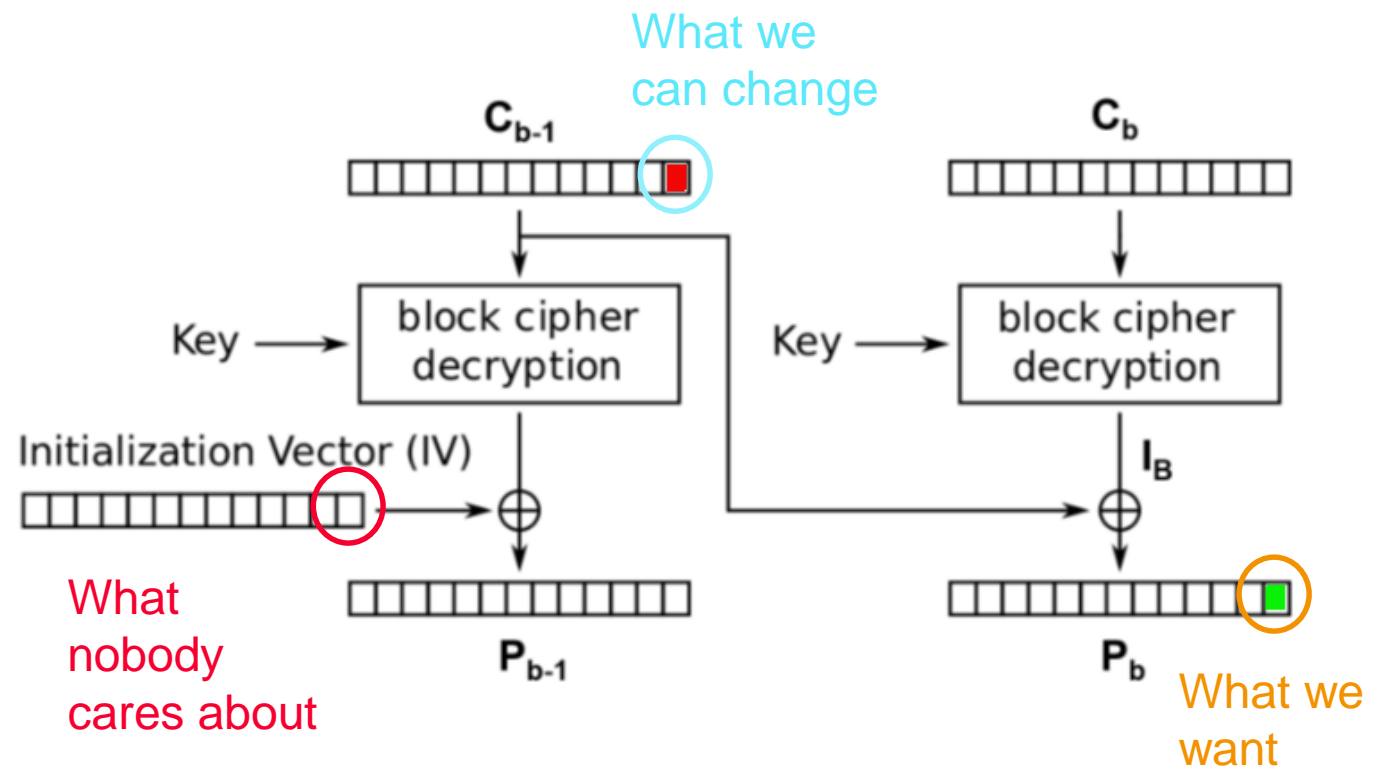
Padding Oracle Attacks – Encrypting Data

This can be done until we reach the **first** block.

If we are lucky, the first block is the **IV**.

No one cares about the IV
~_(\ツ)_/

Btw: This is how to change multiple blocks



Padding Oracle Attacks – Efficiency

- How efficient are padding oracle attacks?
- Efficiency is based on the two factors: b and BS .
 - For every byte, 256 possibilities exists
 - $\sim 256/2 = 128$ guesses on average
 - **Average:** $b * BS * 128$ Requests
 - **Max:** $b * BS * 256$ Requests

Questions?

Any Questions so far?



imgflip.com

Hand-On / CTF

- Mini Padding Oracle CTF
- 3 Challenges in total

1) Installation:

- Good ol' python2
- `pip2 install flask`
- `python2 material/training/main.py`

2) Leeeets go

- <http://127.0.0.1:5000>

We need Tooools

- I've done the heavy lifting for you!
- `material/attack/padd0r.py`
 - Library to attack padding oracles
 - Interesting methods:
 - `crack_block(...)`
 - `decrypt_all_blocks(...)`
 - `change_block(...)`
- + Proof of Concept Skeleton for the first Challenge
 - `/material/attack/poc_po1.py`
 - Finish the PoC and get all the FI4gs