PA193 - Secure coding principles and practices

LABS: Defense in depth

Petr Švenda svenda@fi.muni.cz

CROCS

Centre for Research on
Cryptography and Security

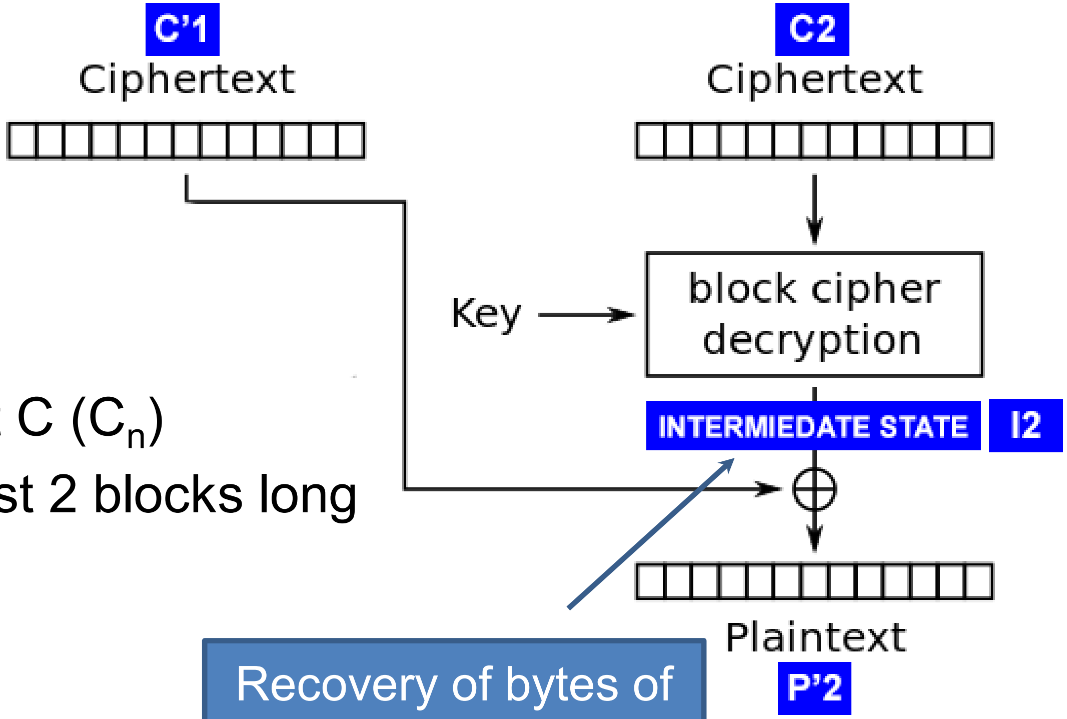
Cipher Block Chaining mode of encryption

[redacted]

- Why we need padding here?
  - – Plaintext is not aligned to cipher block length (e.g., 16B)
  - – PKCS#7 padding – add repeated value of missing length

CBC Padding Oracle Attack (Vaudenay 2002)

- Scenario:
  - Server with unknown secret key K
  - Decrypts incoming encrypted packet
    - Encrypted by AES in CBC mode with PKCS#7 padding
    - Check padding and returns OK or ERROR (no data)
  - Attackers intercept ciphertext C (+IV)
- Attack: recover plaintext P (but no knowledge of K)



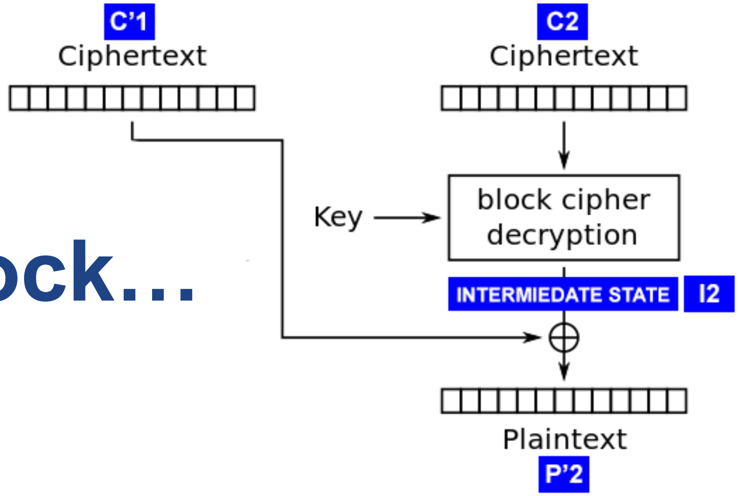
How it works

- Take last block of ciphertext C (C_n)
- Pretend that ciphertext is just 2 blocks long
  - C_n is becoming C_2
- Generate new block C'_1
  - $C'_1[0..14]$ are random bytes
  - $C'_1[15]$ is 0x00
- Send C'_1 | C_2 to server
  - If response is ERROR => increment $C'_1[15]$ and repeat
  - If response is OK => decrypted $P'_2[15]$ is probably 0x01
    - 0x01 is valid padding of length 1
    - => $I_2[15] == C'_1[15] \oplus P_n[15]$
    - => We now have last byte of original plaintext $P_n[15]$!
- Now repeat for $P_n[14]$, $P_n[13]$ … $P_1[1]$, $P_1[0]$

Recovery of bytes of intermediate state

Why probably?

Image curtesy of http://robertheaton.com/2013/07/29/padding-oracle-attack/



Second byte of the last block…

- We now like to obtain $P_n[14]$
- Try to generate decryption with valid padding 02 02
  - We already know value of last byte for $Dec(C_n) = I_2[15]$
  - $C'_1[15]$ will be set to value so $I_2[15] \oplus C'_1[15] == 02$
- Generate new block C'_1
  - $C'_1[0..13]$ are random bytes
  - $C'_1[14]$ is 0x00 (… 0xFF until OK is obtained)
  - $C'_1[15] = I_2[15] \oplus 0x02$
- Send $C'_1 \mid C_2$ to server…

TODO for this lab

1. Download&compile example source code from IS
   - Your favourite IDE (MS Visual Studio, QTCreator…)
2. Implement send input and receive response status
   - Try to modify intercepted ciphertext, try to get OK
3. Implement recovery of last byte of plaintext
4. Implement recovery of all bytes of plaintext

- Use any language you want (example givne in C)
- Reference
  - http://robertheaton.com/2013/07/29/padding-oracle-attack/
  - https://www.iacr.org/archive/eurocrypt2002/23320530/cbc02_e02d.pdf

Helpful online tools

- AES calculator
  - http://extranet.cryptomathic.com/aescalc/index
- String to hex convertor (and vice versa)
  - http://codebeautify.org/string-hex-converter

Questions (try answer on your own)

- How many requests you need to perform to obtain single byte of plaintext?
- Do attack requires plaintext to be exactly two blocks long?
- Is attack more difficult when original plaintext is already aligned and whole additional block with 0x10’s is added?
- Is attack impossible when different padding is used?
  - e.g., data|10000…00 or data data|0123
- Under which condition can attacker decrypt first block?
- Can attacker influence value of resulting plaintext? Can be a specific byte set to a specific value (on decryption)?

Lab - Homework

- Finalize implementation of padding oracle attack
  - And demonstrate recovery of plaintext with lab's code
  - *Submit source code with recovery example*
- Describe at least 5 different, meaningful and the most robust defenses you can think of (against this attack)
  - Explain in one paragraph how it prevents padding oracle attack and what are limitations, cite source, be concrete
  - Assume also other leakage than error message only
  - Compare your defenses (security, overhead, complexity…)
  - *Submit text report: 1-2 A4*
- Implement the defense you think is the strongest one
  - *Submit defense source code*

Lab - Homework

- Upload your solution to IS homework vault
  - Source code of attack and defense + report
- Deadline: 29.9.2016 23:59 (full number of 7 points)
  - Every additional 24h started means 2 points penalization

Project groups and initial setup

- Selection of people in groups (3 people)
- First steps
  - Exchange contact info among the group
  - Send me format you like to write parser for
  - Get confirmation
  - Setup GitHub repository (repo), add developers, setup client git
    - Git cmd, GitHub client, SourceTree…
- Deadlines
  - 23.9. Send me email with names of people in group
  - 25.9. Send me email with format you like to parse and repo name
  - 13.10. Presentation of final parser code (your seminar)