Solving a fixed-key challenge without the key

or – how to padding oracle

malet ∧ crave

October 22, 2018

Given Description

```
Howdy, howdy...
nc 206.189.92.209 12345
[Attachments]
```

The Challenge (Encryption)

```
nc 206.189.92.209 12345

Welcome to our super secure enc/dec server.
We use hmac, so, plz don't hack us (and you can't). Thanks.
Choose one:
1. encrypt data
2. decrypt data
3. quit
1
prefix: hello
suffix: world
41d032627e632e3a9daad7e6b3001e593e5c573cc [ ...MORE BYTES ]
```

The Challenge (Decryption)

```
Choose one:
1. encrypt data
2. decrypt data
3. quit
2
data: 41d032627e632e3a9daad7e6b3001e593e5c573cc [ ...MORE BYTES]
OK
```

A First Look

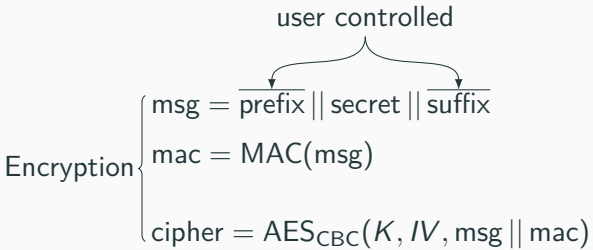
```
encrypt_key = '\xff' * 32
secret = 'MeePwnCTF{#flag_here#}'
hmac_secret = ''
blocksize = 16
hmac_size = 20
```

"*Don't look at the fixed key, it is not a fixed key challenge*" said someone.
And we listened to them...

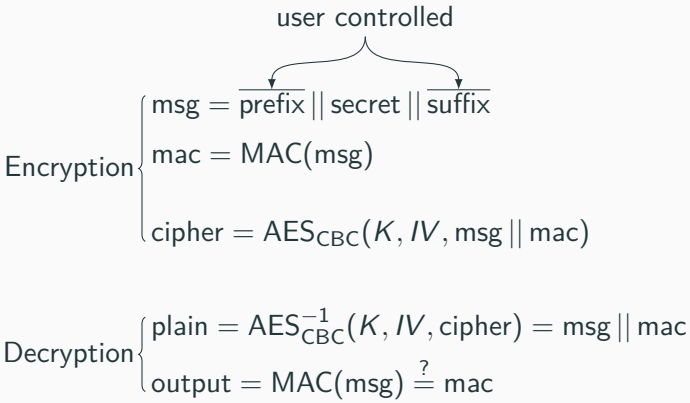
What we know by now

- The start of the flag
- The block and key sizes

An Abstract View



An Abstract View



What we know by now

- The start of the flag
- The block and key sizes
- CBC Mode
- Weird MAC construction
- Mac-then-Encrypt

Cipher Block Chaining – encryption

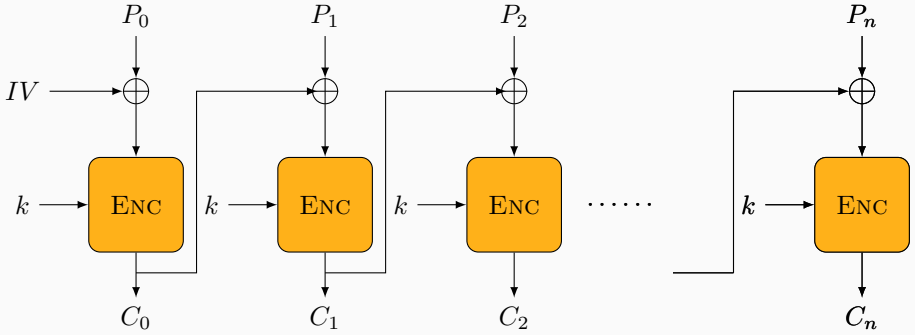


Figure 1: CBC encryption

Cipher Block Chaining – decryption

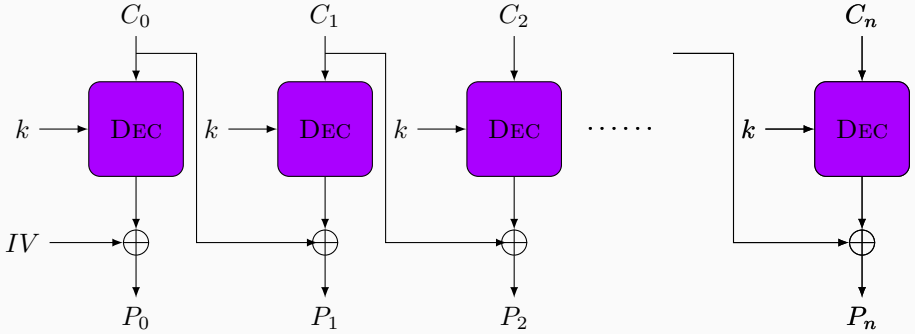
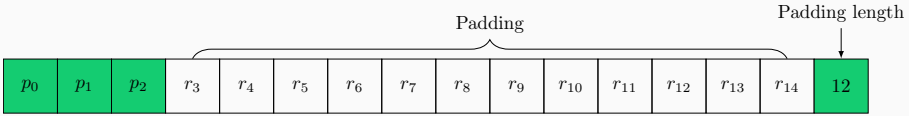
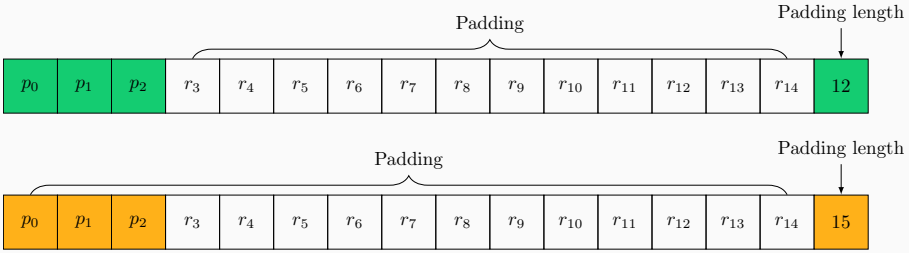


Figure 2: CBC decryption

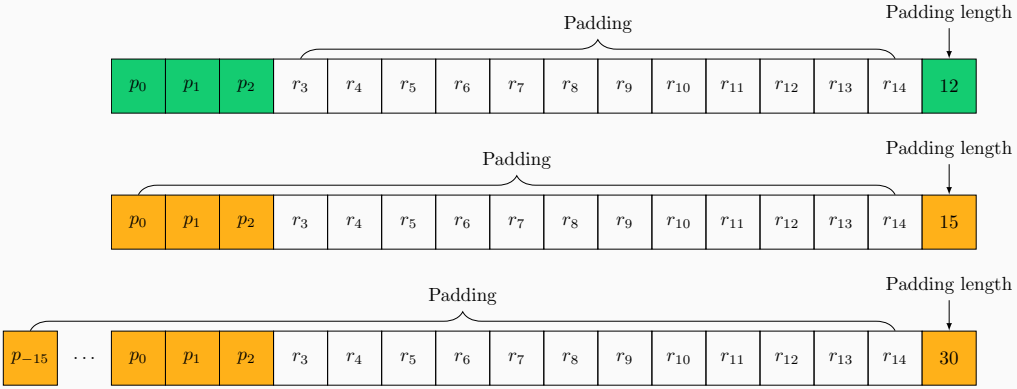
Padding


Padding
Padding length
p_0
p_1
p_2
r_3
r_4
r_5
r_6
r_7
r_8
r_9
r_{10}
r_{11}
r_{12}
r_{13}
r_{14}
12

Padding


Padding
Padding length
p_0 p_1 p_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} 12
Padding
Padding length
p_0 p_1 p_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} 15

Padding

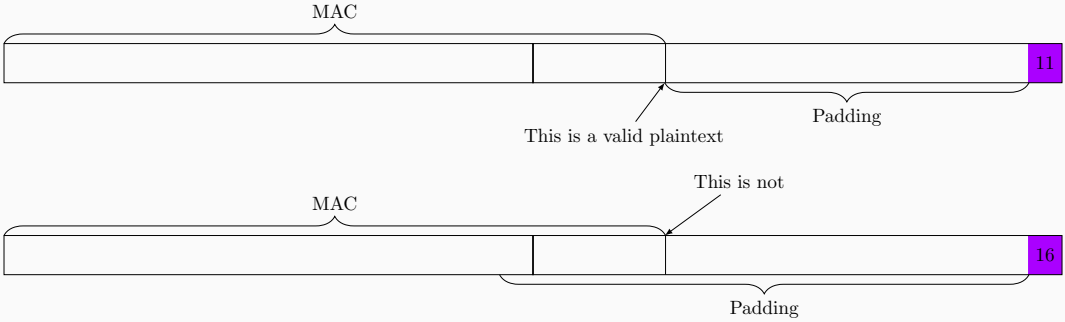

Padding
Padding length
p_0 p_1 p_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} 12
Padding
Padding length
p_0 p_1 p_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} 15
Padding
Padding length
p_{-15} ... p_0 p_1 p_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} 30

A Second Look

```
data = _aes.decrypt(data[blocksize:])
data = unpad(data)
plaintext = data[:-hmac_size]
mac = data[-hmac_size:]
computed = compute_hmac(plaintext)
if mac == computed:
    return True
else:
    return False
```

This looks like an oracle

A Padding Oracle



What we know by now

- The start of the flag
- The block and key sizes
- CBC Mode
- Weird MAC construction
- Mac-then-Encrypt
- Padding Oracle

Attack preliminaries

1. Compute the empty MAC blocks (there is no key)
2. Get the encrypted padded MAC blocks
3. Get the Schrottblock
4. Get the target block

Attack preliminaries

1. Compute the empty MAC blocks (there is no key) ✓
2. Get the encrypted padded MAC blocks
3. Get the Schrottblock
4. Get the target block

Attack preliminaries

1. Compute the empty MAC blocks (there is no key) ✓
2. Get the encrypted padded MAC blocks ✓
3. Get the Schrottblock
4. Get the target block

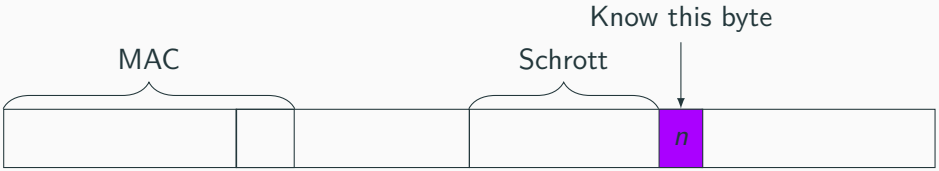
Attack preliminaries

1. Compute the empty MAC blocks (there is no key) ✓
2. Get the encrypted padded MAC blocks ✓
3. Get the Schrottblock ✓
4. Get the target block

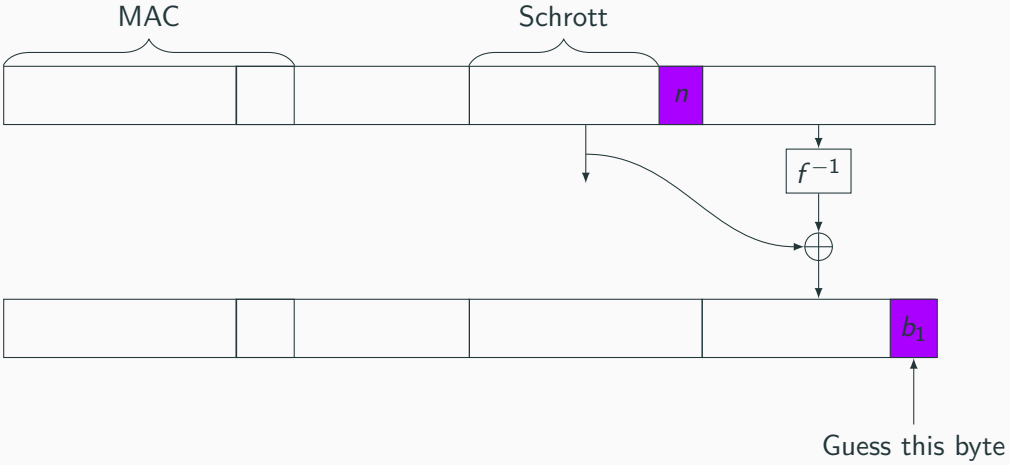
Attack preliminaries

1. Compute the empty MAC blocks (there is no key) ✓
2. Get the encrypted padded MAC blocks ✓
3. Get the Schrottblock ✓
4. Get the target block ✓

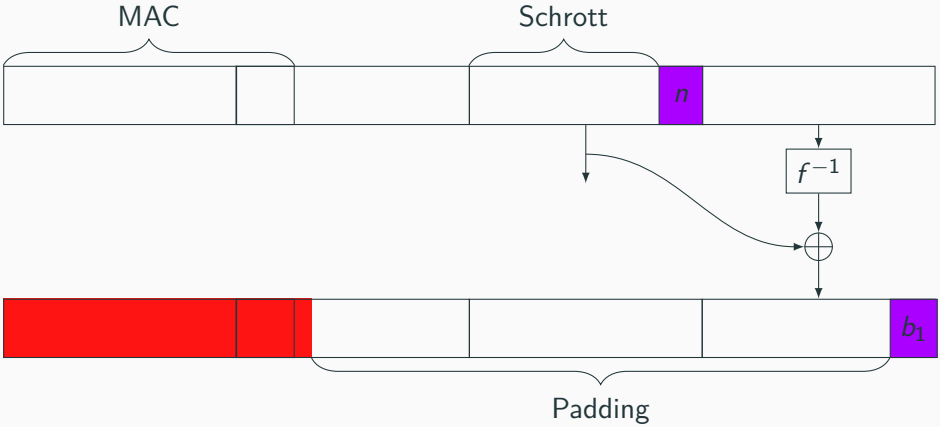
Attack explained



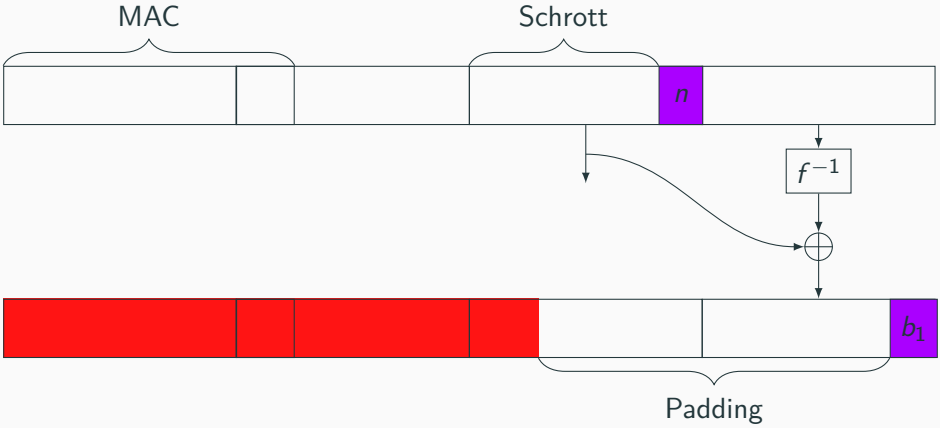
Attack explained



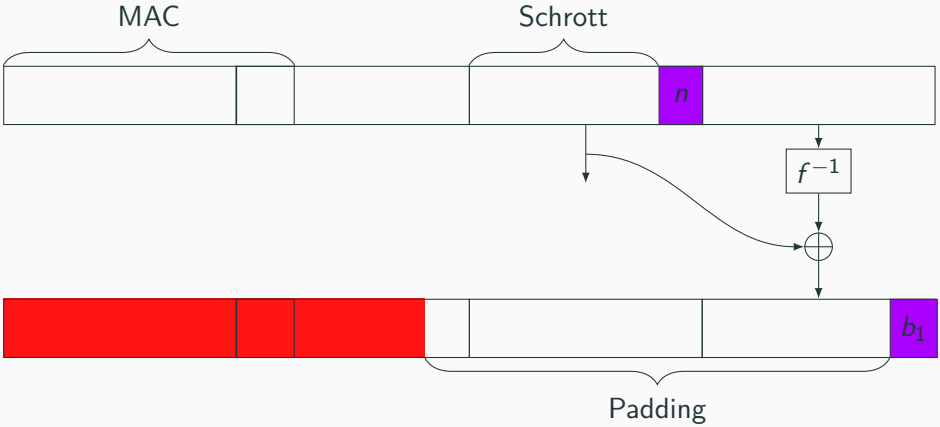
Attack explained



Attack explained



Attack explained



Attack explained