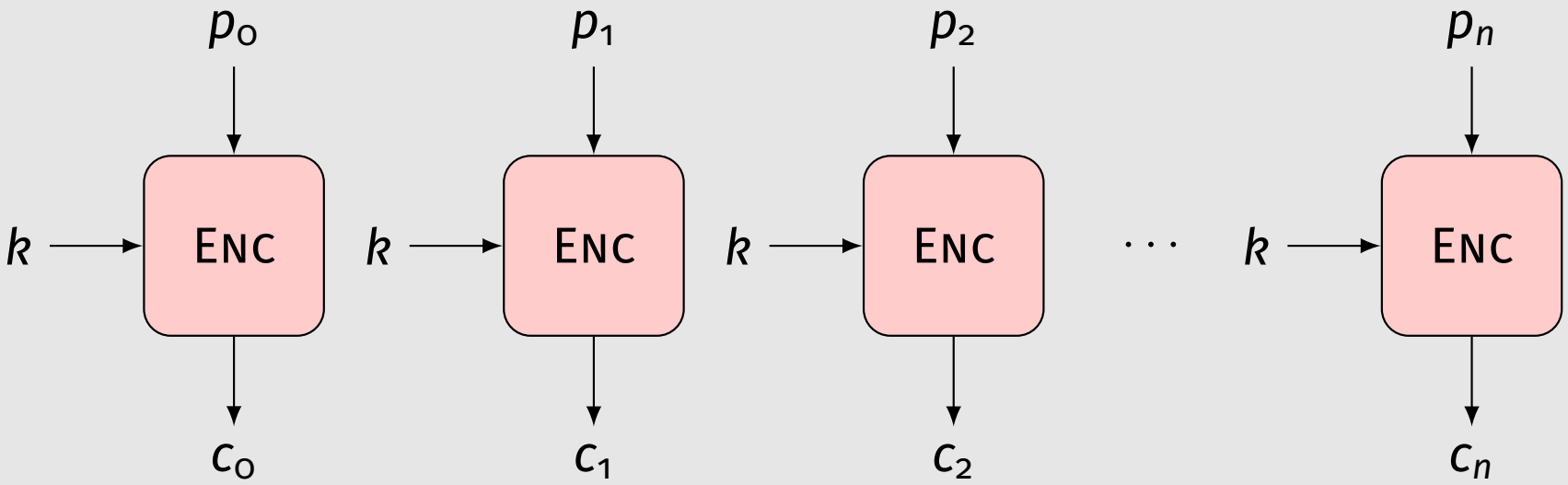
The Padding Oracle Attack

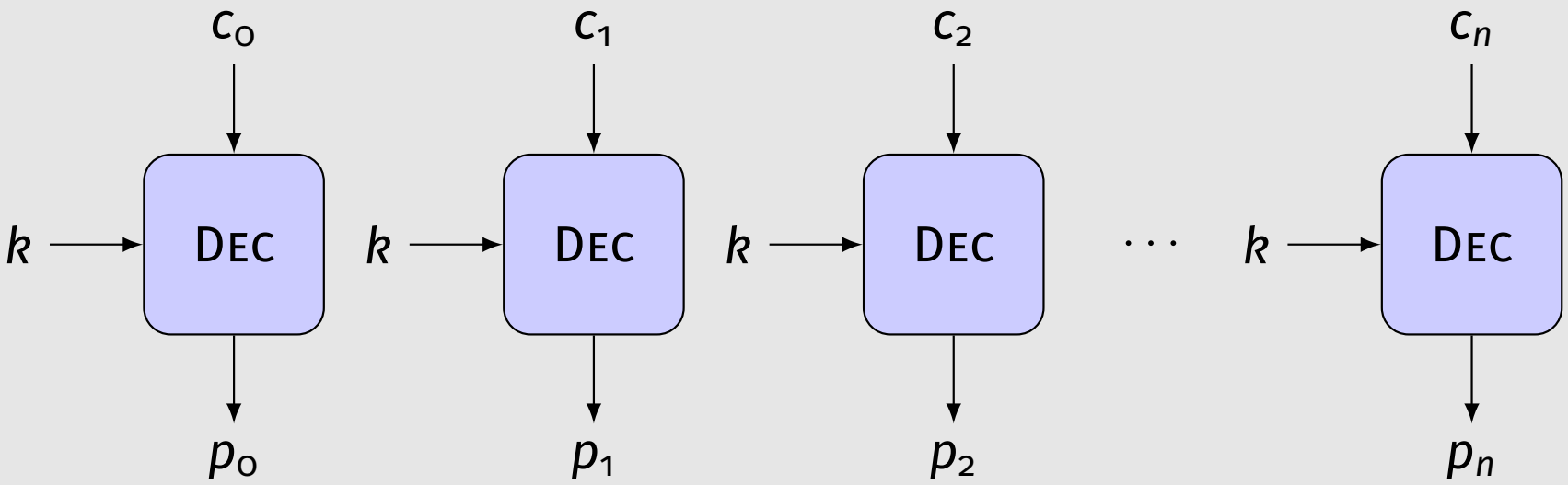
Fionn Fitzmaurice

[redacted]

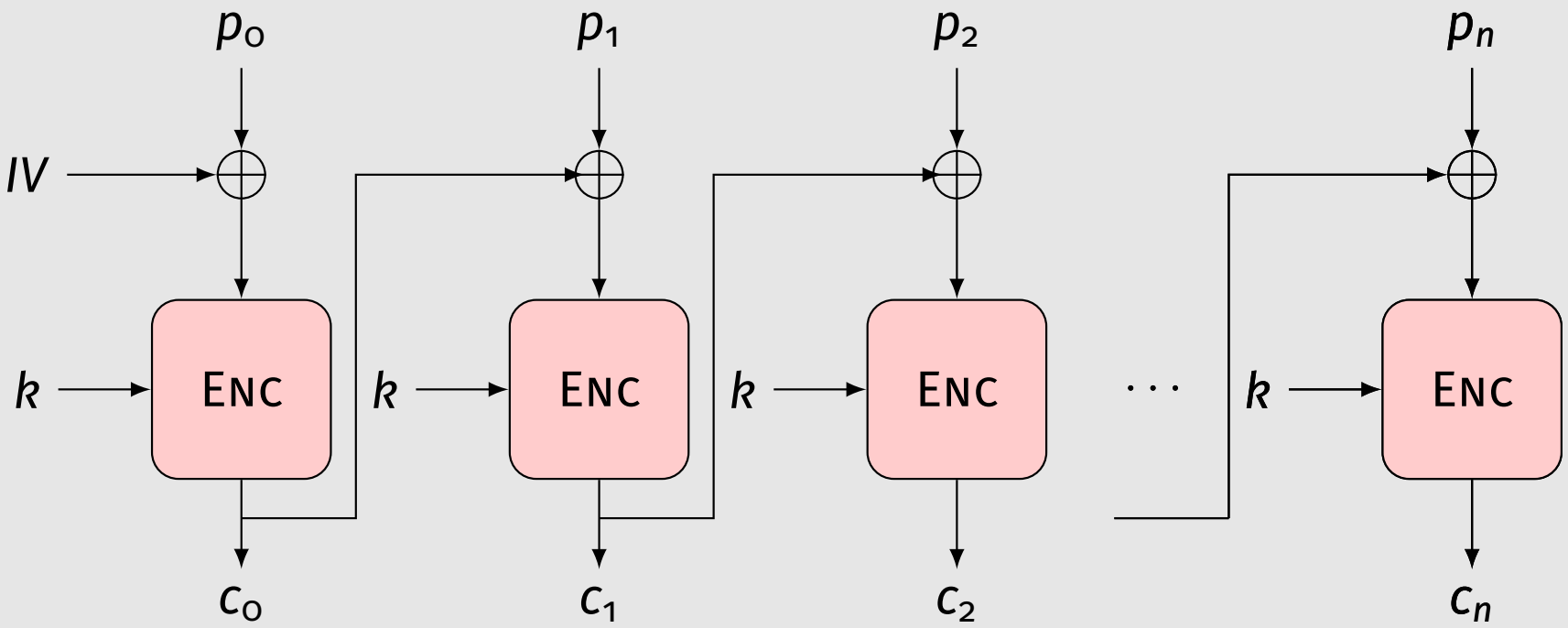
2018-11-01

ECB Mode

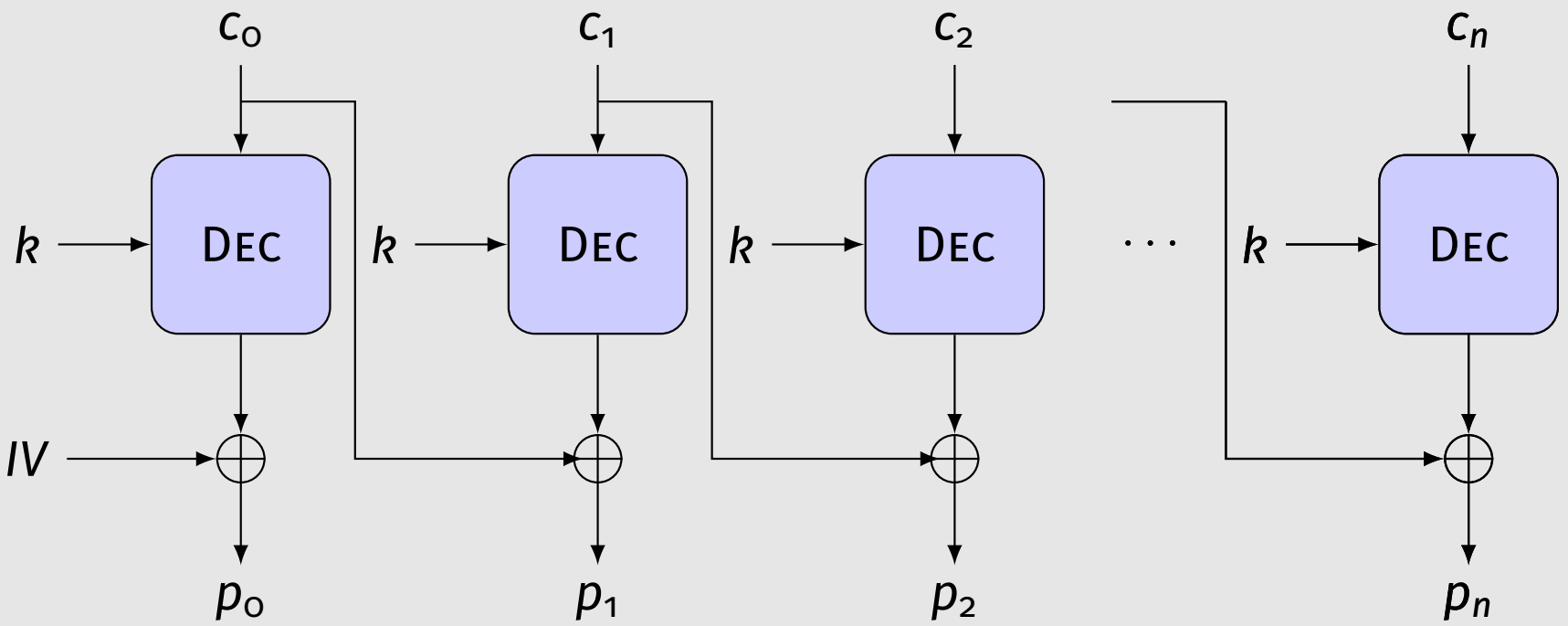

p_0
p_1
p_2
p_n
k
ENC
k
ENC
k
ENC
$\cdots$
k
ENC
c_0
c_1
c_2
c_n


c_0
c_1
c_2
c_n
k
DEC
k
DEC
k
DEC
$\cdots$
k
DEC
p_0
p_1
p_2
p_n

CBC Encryption



CBC Decryption



Padding Blocks (PKCS#7)

Assume blocksize k. ($k < 256$.)

$$[x_1, x_2, \ldots, x_{k-l}] \to [x_1, x_2, \ldots, x_{k-l}, \underbrace{l, l, \ldots, l}_{l}].$$

E.g.

$$[x_1, x_2, \ldots, x_{k-4}] \to [x_1, x_2, \ldots, x_{k-4}, 4, 4, 4, 4]$$

(where 4 is the *byte* `\x04`).

Padding Blocks (PKCS#7)

Assume blocksize k. ($k < 256$.)

$$[x_1, x_2, \ldots, x_{k-l}] \to [x_1, x_2, \ldots, x_{k-l}, \underbrace{l, l, \ldots, l}_{l}].$$

E.g.

$$[x_1, x_2, \ldots, x_{k-4}] \to [x_1, x_2, \ldots, x_{k-4}, 4, 4, 4, 4]$$

(where 4 is the *byte* `\x04`).

If the data is already of length k, append a block.

$$[x_1, x_2, \ldots, x_k] \to [x_1, x_2, \ldots, x_k] + [\underbrace{k, k, \ldots, k}_{k}].$$

The Padding Oracle

$$\mathcal{O}(c) = \begin{cases} 1 & \text{if } p \text{ has valid padding,} \\ 0 & \text{otherwise.} \end{cases}$$

Malicious Cyphertext

Let there be n blocks of plaintext, each of blocksize k. For $i \in [1, n]$,

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

where c_i is the i^{th} cyphertext block, p_i the i^{th} plaintext block and c_0 the initialisation vector.

Construct some block c' and concatonate with c_n. The corresponding plaintext is $c'|c_n \leftrightarrow p'_1|p'_2$.

Construct some block c' and concatonate with c_n. The corresponding plaintext is $c'|c_n \leftrightarrow p'_1|p'_2$.

$$p'_1 = D(c') \oplus c_0, \quad p'_2 = D(c_n) \oplus c'$$
$$= D(E(p_n \oplus c_{n-1})) \oplus c'$$
$$= p_n \oplus c_{n-1} \oplus c'$$

giving

$$p_n = p'_2 \oplus c' \oplus c_{n-1}.$$

Construct some block c' and concatonate with c_n. The corresponding plaintext is $c'|c_n \leftrightarrow p'_1|p'_2$.

$$p'_1 = D(c') \oplus c_0, \quad p'_2 = D(c_n) \oplus c'$$
$$= D(E(p_n \oplus c_{n-1})) \oplus c'$$
$$= p_n \oplus c_{n-1} \oplus c'$$

giving

$$p_n = p'_2 \oplus c' \oplus c_{n-1}.$$

p'_2 is at the end of our plaintext block so must have valid padding. This constrains c'!

If only there was some way to tell if the plaintext had valid padding...

How to construct c' such that $p'_2 = D(c_n) \oplus c'$ has valid padding?

By submitting $c'|c_n$ to the oracle!

$$\mathcal{O}(c) = \begin{cases} 1 & \text{if } p \text{ has valid padding,} \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathcal{O}(c'|c_n) = 1 \;\Rightarrow\; p'_2 \text{ has valid padding.}$$

The Last Byte

Take $p_n = p'_2 \oplus c' \oplus c_{n-1}$ and consider only the last byte. If we constrain $p'_2[k] = 1$, p'_2 has valid padding.

Construct $c' = c'_k$ such that this is true. Then

$$\begin{aligned} p_n[k] &= p'_2[k] \oplus c'_k[k] \oplus c_{n-1}[k] \\ &= 1 \oplus c'_k[k] \oplus c_{n-1}[k]. \end{aligned}$$

The Last Byte

Take $p_n = p'_2 \oplus c' \oplus c_{n-1}$ and consider only the last byte. If we constrain $p'_2[k] = 1$, p'_2 has valid padding.

Construct $c' = c'_k$ such that this is true. Then

$$\begin{aligned} p_n[k] &= p'_2[k] \oplus c'_k[k] \oplus c_{n-1}[k] \\ &= 1 \oplus c'_k[k] \oplus c_{n-1}[k]. \end{aligned}$$

We only care about $p'_2[k] = D(c_n)[k] \oplus c'_k[k]$, so let

$$c'_k = [\underbrace{0, 0, \ldots, 0}_{\text{(anything)}}, b].$$

Iterate over all values of b until $\mathcal{O}(c'_k | c_n) = 1$ and, via $p_n[k] = 1 \oplus b \oplus c_{n-1}[k]$, you've found the last byte.

The Second-Last Byte

Another valid plaintext would be $p'_2[k] = p'_2[k-1] = 2$.

$$\therefore p_n[k-1] = 2 \oplus c'_{k-1}[k-1] \oplus c_{n-1}[k-1]$$

and

$$c'_{k-1}[k] = c'_k[k] \oplus 1 \oplus 2.$$

All the Bytes!

$$
\begin{aligned}
p_n[k] &= 1 \oplus c'_k[k] \oplus c_{n-1}[k], \\
p_n[k-1] &= 2 \oplus c'_{k-1}[k-1] \oplus c_{n-1}[k-1], \\
&\vdots \\
p_n[k-i] &= (i+1) \oplus c'_{k-i}[k-i] \oplus c_{n-1}[k-i]
\end{aligned}
$$

for $i \in [0, k)$ and

$$
c'_{k-i}[k-j] = c'_{k-i+1}[k-j] \oplus i \oplus (i+1)
$$

where $j < i$.

All the Blocks!

Just repeat this for every block (except the first one).

Caveats

To find $p_n[k]$ we iterate over possible values of $c'_k[k] = b$ and when $\mathcal{O}(c'_k|c_n) = 1$ we conclude that the corresponding $p'_2[k] = 1$.

But there's a chance that $p'_2[k-1] = 2$ and we found $c'_k[k]$ corresponding to $p'_2[k] = 2$.

Similarly, if $p'_2[k-2] = p'_2[k-1] = 3$.

Solution 1

Take $c_{n-1}|c_n \leftrightarrow p_{n-1}|p_n$ and vary c_{n-1} from left to right. When

$$\mathcal{O}(c'_{n-1}|c_n) = 0$$

on the i^{th} byte, the remaining $k - i$ bytes are all $k - i$.

Solution 2

For the last byte, $c'[k]$, check if it really corresponds to $p'_j[k] = 1$ by varying $c'[k-1]$.

If $\mathcal{O}(c''|c_j) = 0$, we were wrong about the last byte.

This only needs to be checked once per block.

This is Real

- SSL,
- IPSec,
- lots of web applications,
- Steam,
- &tc.

```
def attack_block(block, c=cbc_oracle()):
    c = [c[i:i+16] for i in range(0, len(c), 16)]
    c_prime, p = bytearray(16), bytearray(16)
    for i in range(15, -1, -1):
        for b in range(256):
            c_prime[i] = b
            if padding_oracle(bytes(c_prime) + c[block]):
                if i == 15:
                    c_test = c_prime
                    c_test[i - 1] ^= c_prime[i]
                    if padding_oracle(bytes(c_test) + c[block]):
                        break
                else:
                    break
        p[i] = (16 - i) ^ c_prime[i] ^ c[block - 1][i]
        for j in range(i, 16):
            c_prime[j] = c_prime[j] ^ (16 - i) ^ (16 - i + 1)
    return bytes(p)

def attack(c=cbc_oracle()):
    p = [attack_block(i, c) for i in range(len(c) // 16 - 1, 0,
        -1)]
    return de_pkcs7(b''.join(p[::-1]))
```

https://asciinema.org/a/81051