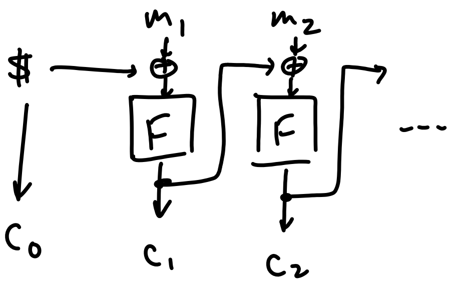
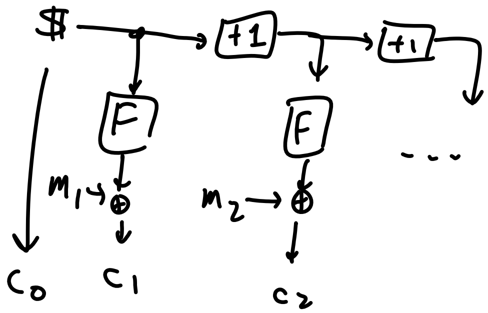
Padding & Stealing & Oracles

HW2 returned via email tonight (sorry)
exam in 1 week (sorry)

CBC mode:

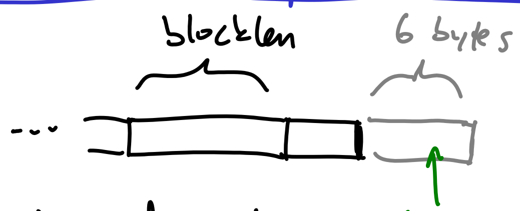


CTR mode



If ptxt not exact multiple of blocklength:

Padding



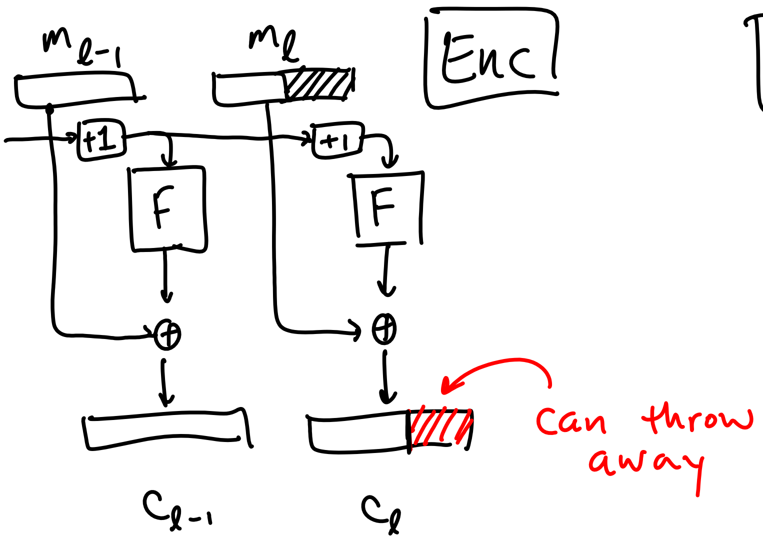
in this example, pad with Øx 00 00 00 00 00 06 (ANSI X.932)

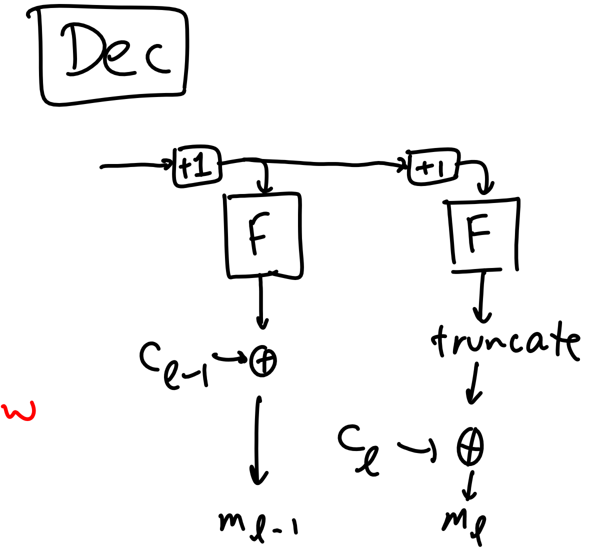
0x 06 06 06 06 06 06 (some standard)

0x 80 00 00 00 00 00 (some other standard)

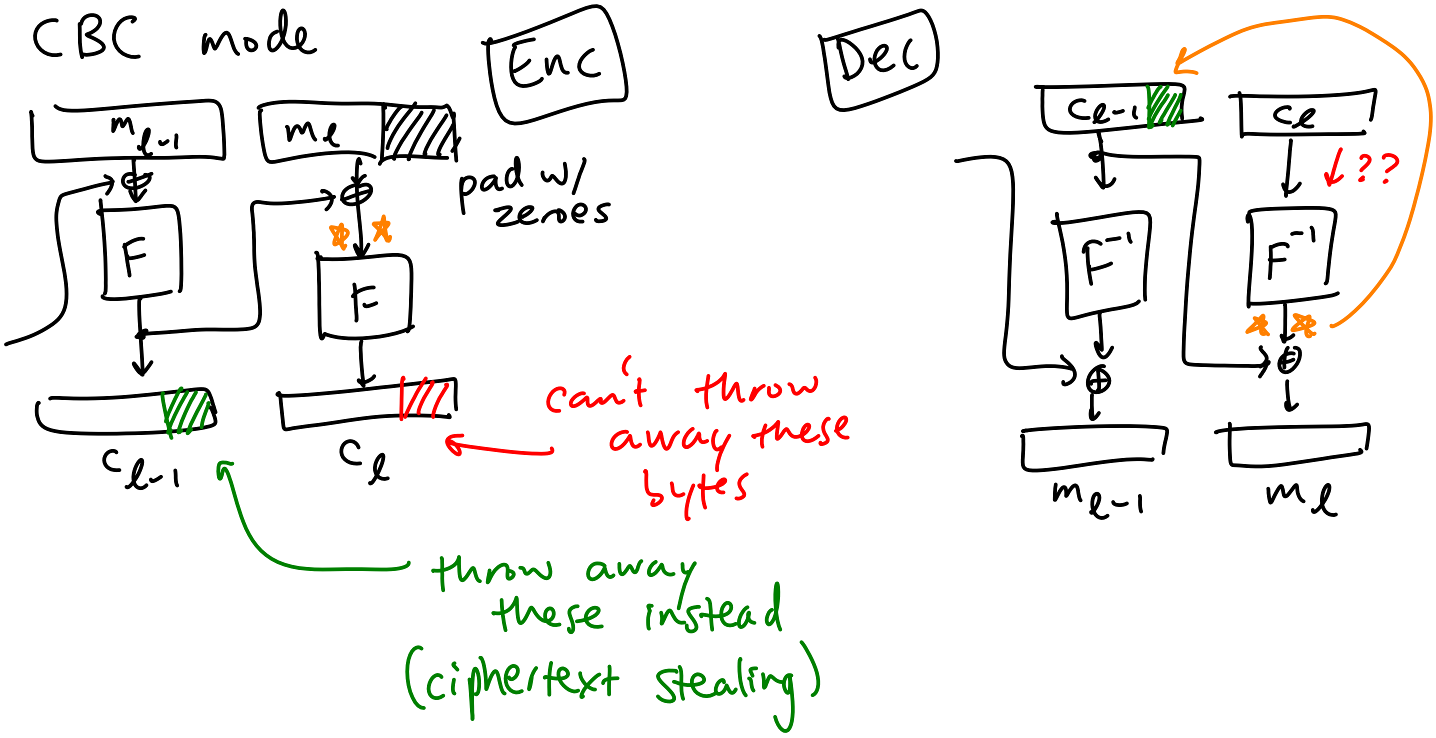
Truncation

In CTR mode





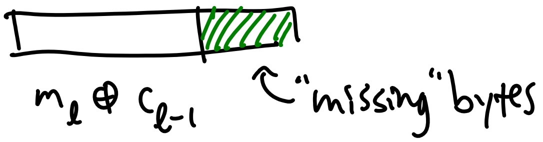
In CBC mode



can't throw away these bytes

throw away these instead (ciphertext stealing)

value on ☆ is



Padding Oracle Attacks

(Padding is not a bad thing, but is just most common culprit in practice)

Webserver

```
get ctxt c from browser
m = Dec(k, c)        // CBC mode
if m has invalid padding
    return error
else
    do something w/ m
```

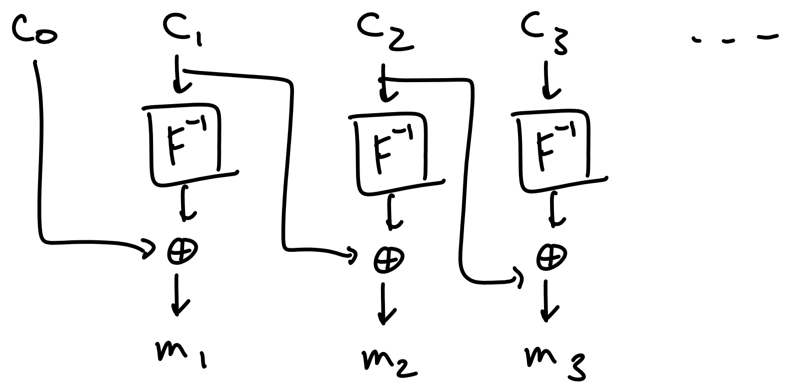
"padding oracle"

client can tell whether Dec(k, c) has valid padding

Claim: No matter what else Webserver does, an attacker can decrypt any ciphertext now ☹

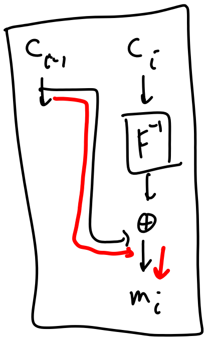
Observation:

CBC decryption



Suppose $(c_0, c_1, \ldots, c_\ell)$ is $\mathrm{Enc}(k, m_1, \ldots, m_\ell)$
for unknown m

What is $\mathrm{Dec}(k, (c_{i-1}, c_i))$? m_i



Suppose x is chosen by Adv,
What is $\mathrm{Dec}(k, (x \oplus c_{i-1}, c_i))$? $x \oplus m_i$

If I submit $(x \oplus c_{i-1}, c_i)$ to webserver,
I learn whether $x \oplus m_i$ has valid padding

Observation #2:

Submit $(c_{i-1} \oplus [000\cdots001], c_i)$ to server $\Rightarrow$ learn whether $m_i \oplus [000\cdots001]$ has valid padding

Submit $(c_{i-1} \oplus [000\cdots002], c_i)$ to server

Submit $(c_{i-1} \oplus [000\cdots003], c_i)$ to server

$\vdots$

Submit $(c_{i-1} \oplus [000\cdots0ff], c_i)$ to server

$\Rightarrow$ probably only 1 of these has valid padding ex: $m_i \oplus [000 \cdots 00c4]$ has valid padding

⇒ $m_i \oplus [\cdot \cdots 00c4]$ must end in byte Ø1

⇒ last byte of m_i must be c3