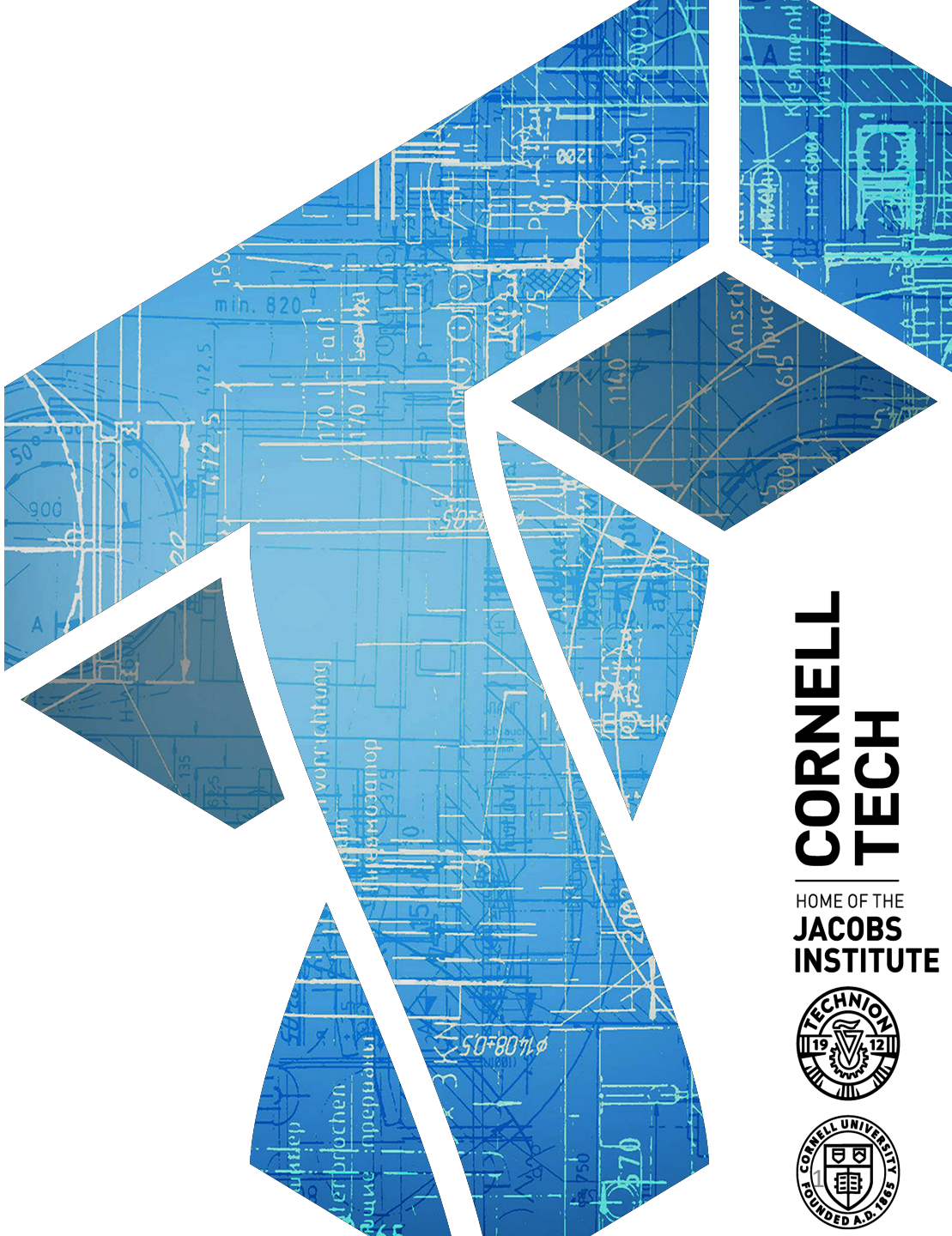
CS 5830
Cryptography

Instructor: Tom Ristenpart
TAs: Yan Ji, Sanketh Menda



CORNELL TECH
HOME OF THE JACOBS INSTITUTE

Recap and where we're at

- Blockciphers and their security goals
  - Assume good blockciphers that achieve PRF security up to implications of best-known generic attacks
  - ~ 2^k time (exhaustive key search)
  - ~ $2^{n/2}$ time (birthday attacks)

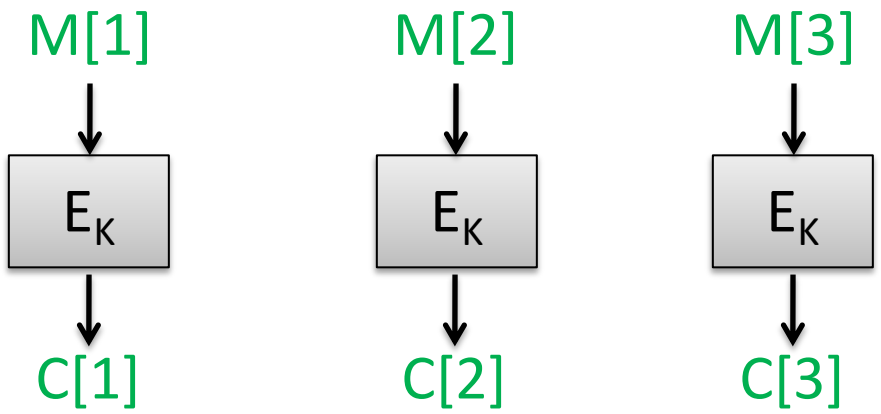
- Today: modes of operation and (time allowing) chosen-ciphertext attacks

Block cipher modes of operation

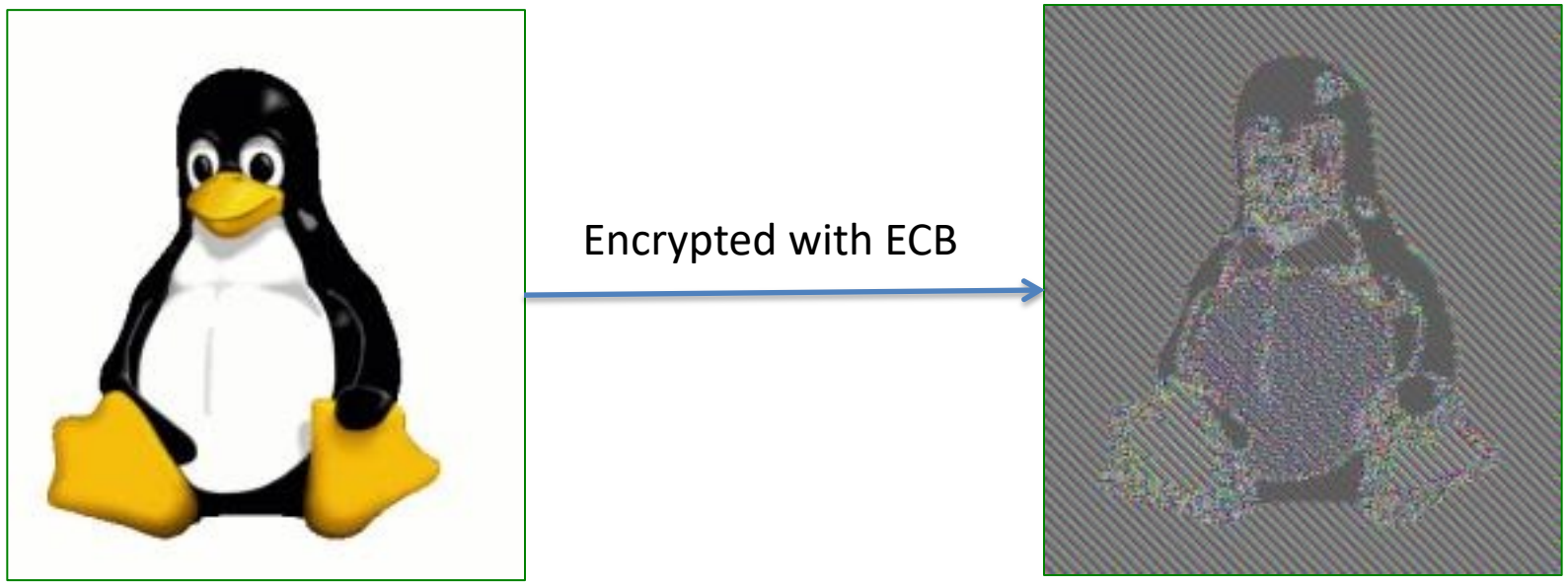
Electronic codebook (ECB) mode

Pad message M to M[1],M[2],M[3],... where each block M[i] is n bits

Then:



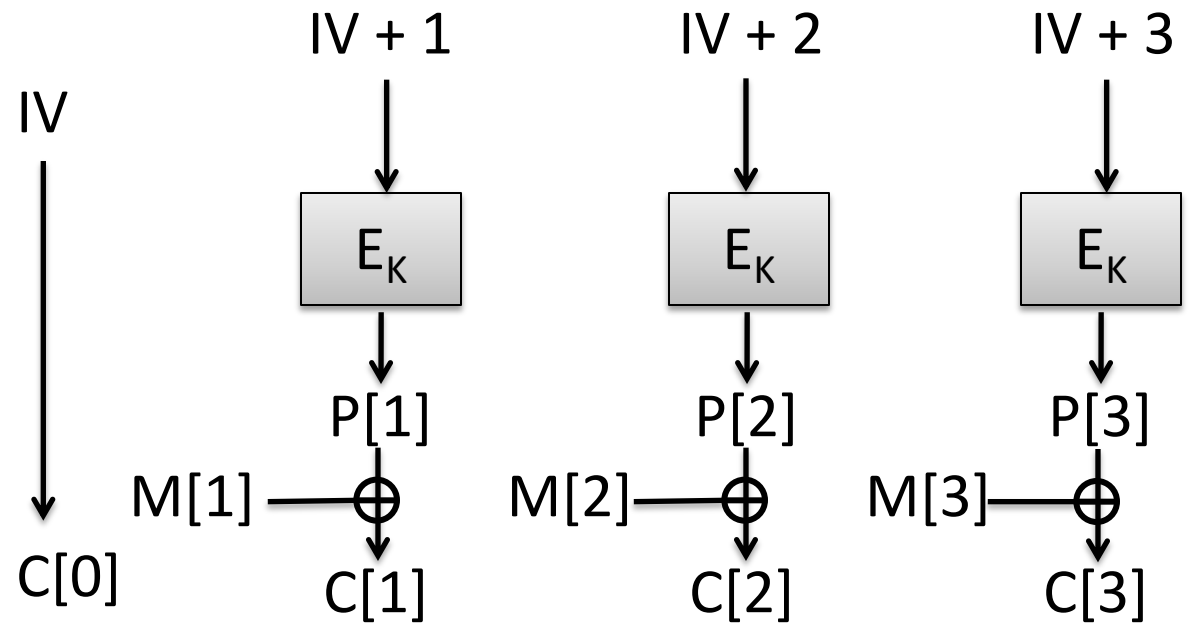
ECB is insecure



Images courtesy of
http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

CTR mode

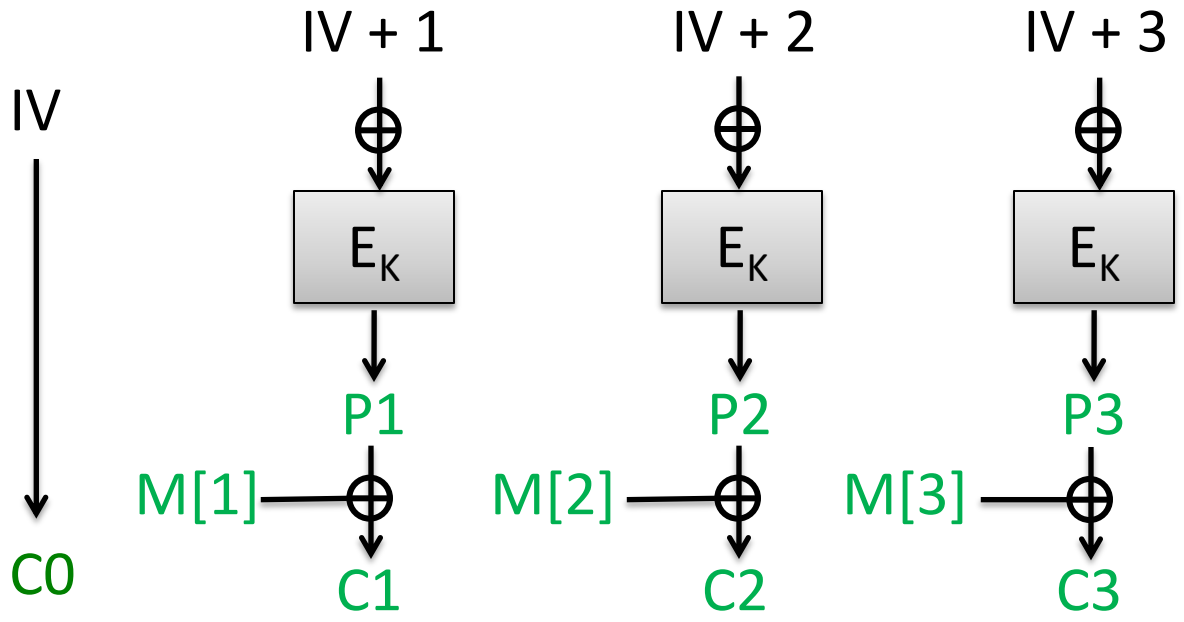
Block cipher $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a family of permutations
Should be secure as a pseudorandom function (PRF)



CTR mode provides message confidentiality (nothing about message from ciphertext) assuming E is a PRF and number of message blocks encrypted $<< 2^{n/2}$

CTR is a blockcipher mode of operation

- How do we encrypt long messages with block cipher?
  - Modes of operation
- Long history: NIST standard
  - First published in 1980 specifically for DES
  - 2001 version: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf
- Analyses starting in 1990s for chosen-plaintext attacks
- Analyses starting in 2000s for chosen-ciphertext attacks



Can attacker learn K from just C0,C1,C2,C3?

Implies attacker can break E, i.e. recover block cipher key

Can attacker learn M = M[1],M[2],M[3] from C0,C1,C2,C3?

Implies attacker can invert the block cipher without knowing K

Can attacker learn one bit of M from C0,C1,C2,C3?

Implies attacker can break PRF security of E

Passive adversaries cannot learn anything about messages

Formalizing security for CPA attacks

- Indistinguishability under chosen-plaintext attacks (IND-CPA)
  - Multi-message, adaptive security
  - Query encryption oracle q times adaptively with pair of equal-length messages M_0, M_1 ; get back $Enc(K, M_b)$ for secret key K
  - Can adversary infer b?

IND-CPA(SE, $\mathcal{A}$):
K <-$ Kg ; b <-$ {0,1}
b' <-$ $\mathcal{A}^{LR}$
Return (b = b')

LR(M_0,M_1):
C <-$ Enc(K,M_b)
Return C

Def. A symmetric encryption scheme is (t, q, L, ϵ)-IND-CPA if for any adversary $\mathcal{A}$ running in time at most t and making at most q queries of length at most L bits, it holds that

$$\Pr[\text{IND-CPA}(SE, \mathcal{A}) = 1] \leq 1/2 + \epsilon$$

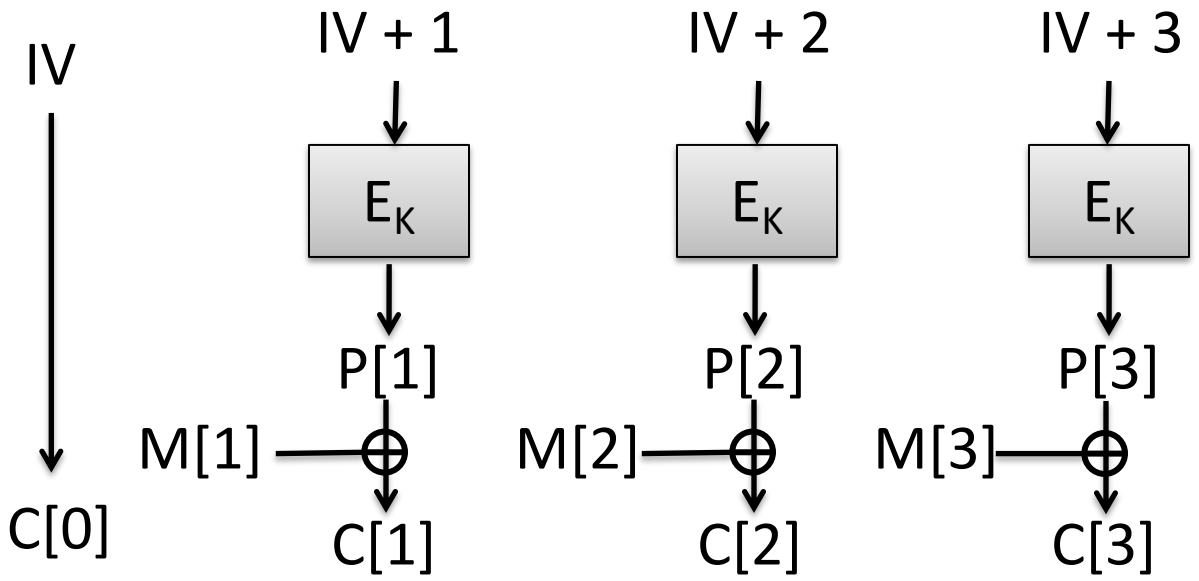
CTR mode IND-CPA security

Thm. Let E be a blockcipher with blocksize n. Then CTR-mode using E is (t,q,L,ϵ_{cpa})-secure with

$$\epsilon_{cpa} \leq \epsilon_{prf} + (\sigma q)^2 / 2^n$$

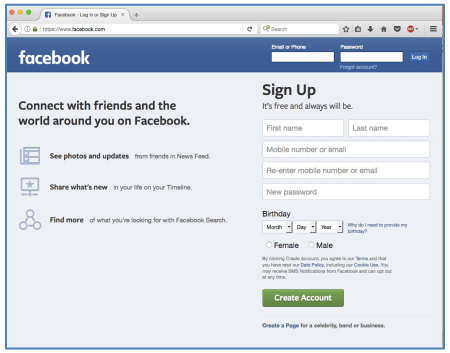
assuming E is $(t,q\sigma,\epsilon_{prf})$-PRF secure blockcipher for $\sigma = \lceil L/n \rceil$.

Back to CTR mode



Attack type	Capability	Goal
Indistinguishability under chosen plaintext attack	Observe ciphertexts under secret key of chosen plaintexts with >=1 bit of uncertainty	Learn at least one bit of information about plaintext
Ciphertext integrity attack	Get example ciphertext(s); maul ciphertexts and submit to decryption oracle	Trick recipient into accepting forged plaintext
Indistinguishability under chosen-ciphertext attack	Get example ciphertext(s); maul ciphertexts and submit to partial decryption oracle	Learn information about plaintexts

Session handling and login



GET /index.html

Set-Cookie: AnonSessID=134fds1431

Protocol is HTTPS. Elsewhere else just HTTP.

POST /login.html?name=bob&pw=12345

Cookie: AnonSessID=134fds1431

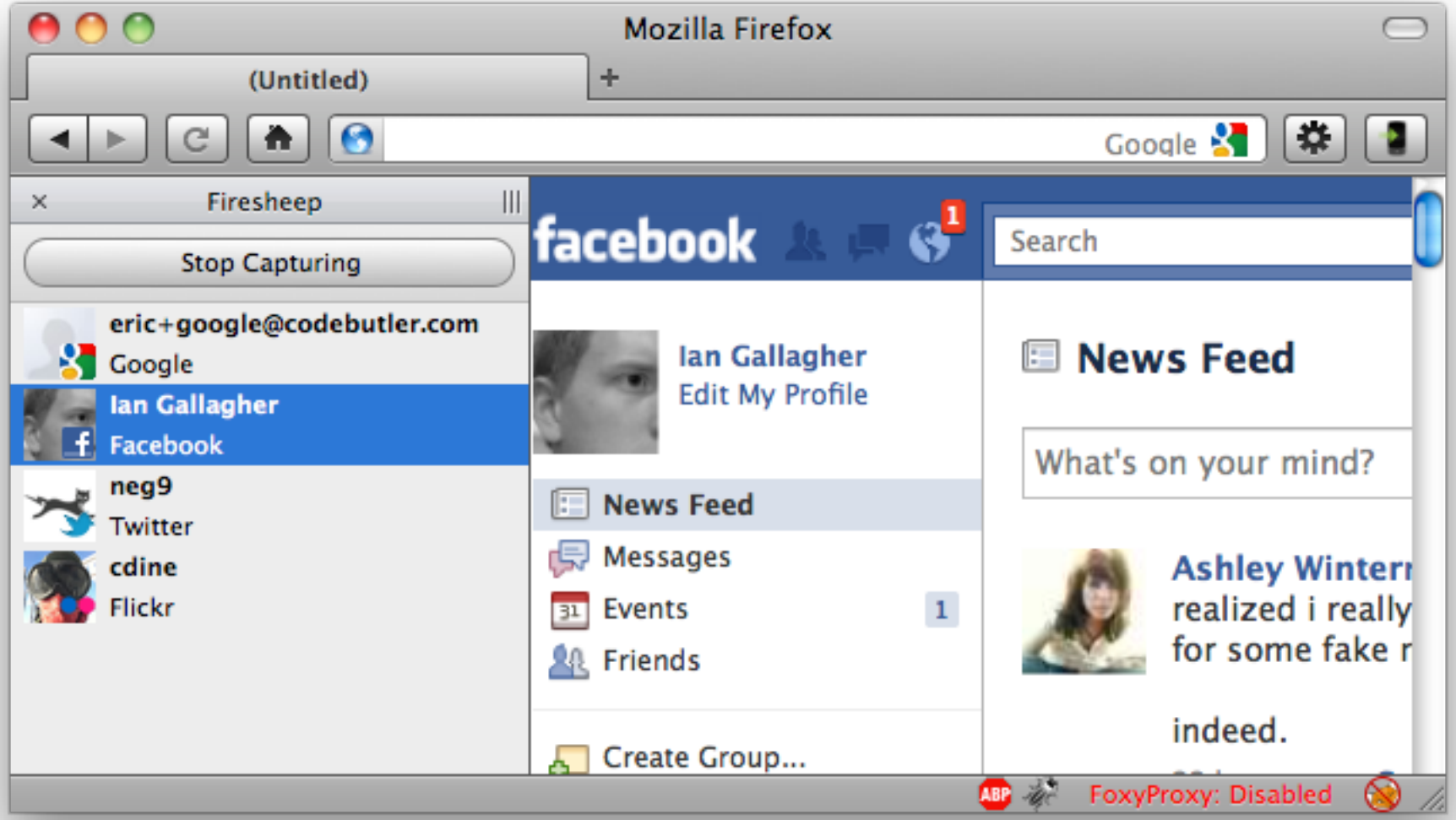
Set-Cookie: SessID=83431Adf

Nowadays increasingly all HTTPS

GET /account.html

Cookie: SessID=83431Adf

Without HTTPS: session hijacking often trivial



From http://codebutler.com/firesheep

Security problems here?

Facebook.com

POST /login.html?name=bob&pw=12345

Cookie: AnonSessID=134fds1431

Set-Cookie: SessID=83431Adf

GET /account.html

Cookie: SessID=83431Adf

Secret key K only known to server

83431Adf = CTR-Enc(K, “admin=0”)

Malicious client can simply flip a few bits to change admin=1

Example of an ***integrity / authenticity violation***

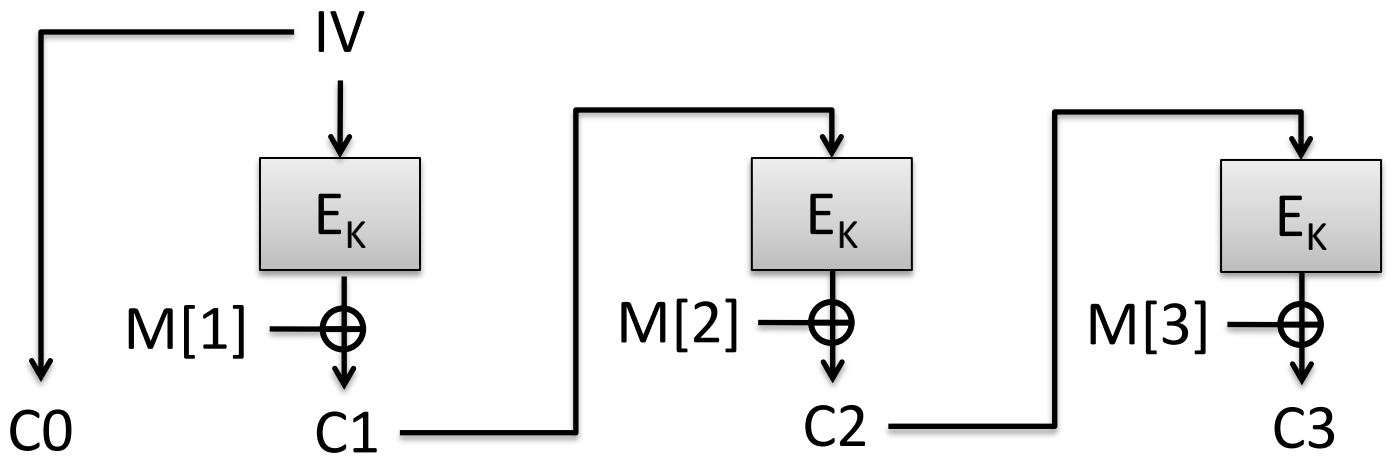
NIST Modes

- Electronic codebook mode (ECB)
- Counter mode (CTR)
- Ciphertext feedback mode (CFB)
- Offset feedback mode (OFB)
- Ciphertext block chaining mode (CBC)

CTR, CBC found widespread use

CFB mode

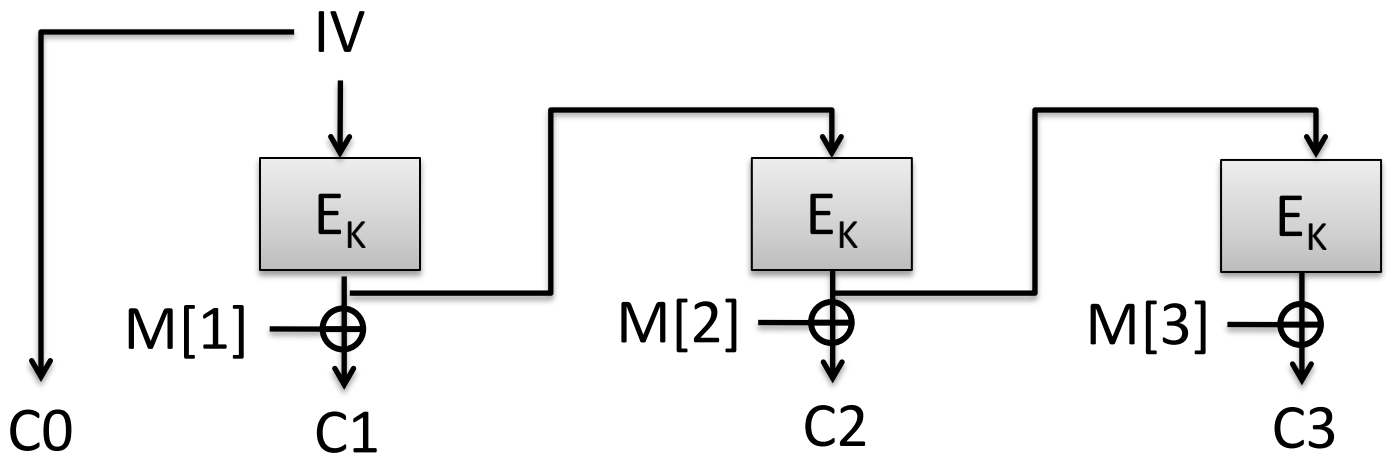
Ciphertext feedback mode (CFB)
Pad message M to M[1],M[2],M[3],... where each block M[i] is n bits
Choose random n-bit string IV
Then:



How do we decrypt?

OFB mode

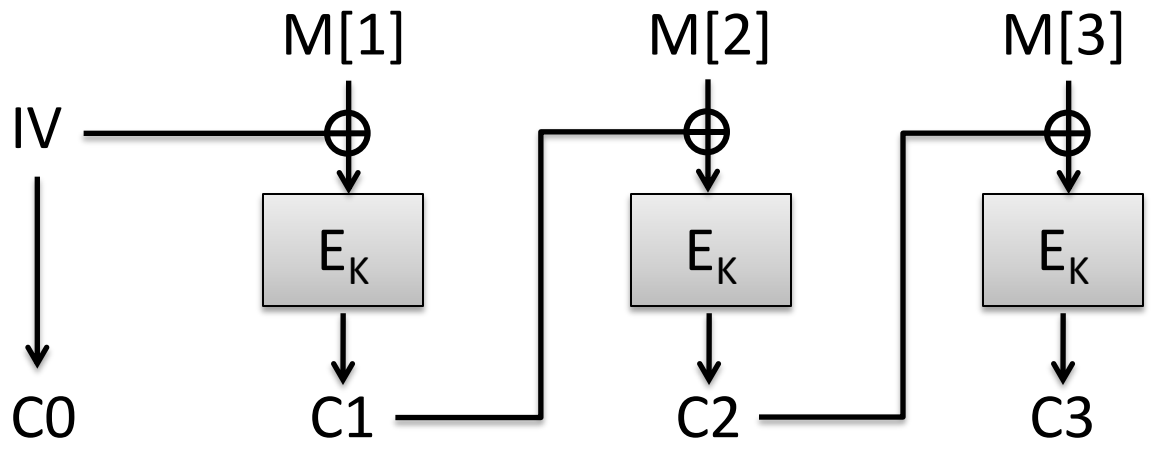
Offset feedback mode (OFB)
Pad message M to M[1],M[2],M[3],... where each block M[i] is n bits
Choose random n-bit string IV
Then:



How do we decrypt?

CBC mode

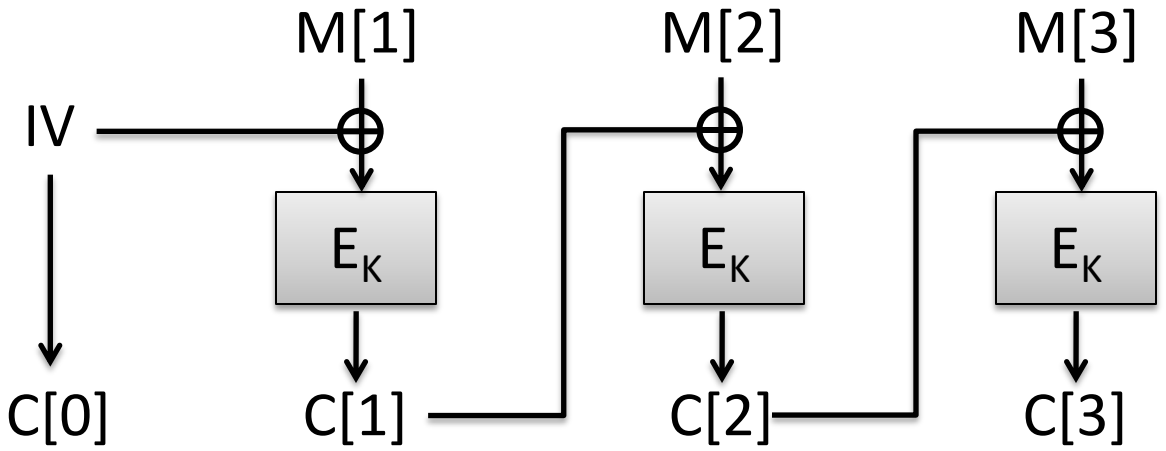
Ciphertext block chaining (CBC)
Pad message M to M[1],M[2],M[3],... where each block M[i] is n bits
Choose random n-bit string IV
Then:



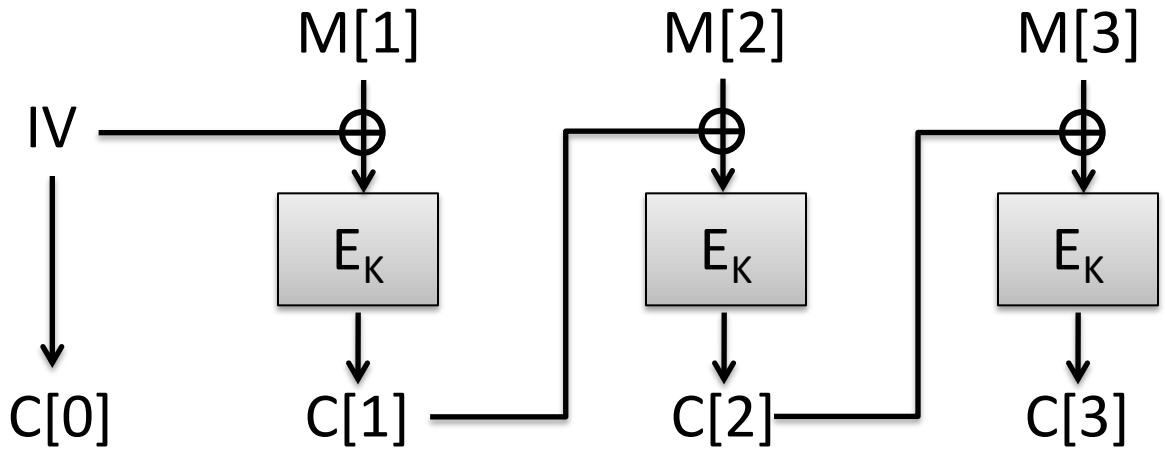
How do we decrypt?

IND-CPA of CBC mode

- CBC mode can also be shown to be IND-CPA secure, with similar security level as that of CTR mode
  - What if IV collides?
  - Must switch keys before number of encryptions q gets close to $2^{n/2}$

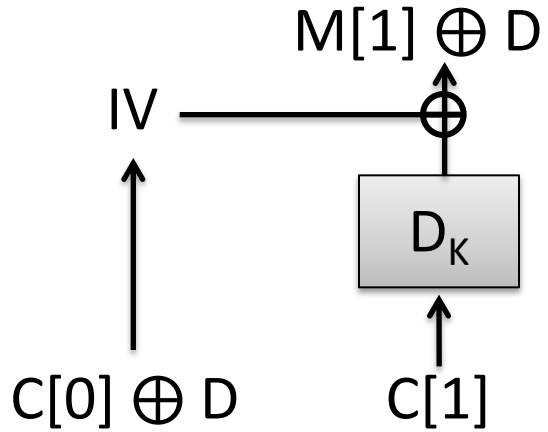


CBC mode has malleability issues, too



How do we change bits of M received by server?

For any D:



Padding for CBC mode

- CBC mode handles messages with length a multiple of n bits
- We use padding to make it work for arbitrary message lengths
  - PadCBC, UnpadCBC map to, from strings of length multiple of n

- Padding checks often give rise to chosen-ciphertext attack called ***padding oracle attacks***
  - Given CBC mode encryption C = Enc(K,M) for unknown M
  - Access to oracle that reveals just whether decryption succeeds
  - Recover M

Pseudocode for CBC mode with padding

Kg():
K <-$ $\{0,1\}^k$

Pick a random key

CBC-Enc(K,M):
L <- |M| ; m <- ceil(L/n)
C[0] <- IV <-$ $\{0,1\}^n$
M[1],…,M[m] <- PadCBC(M,n)
For i = 1 to m do
    C[i] <- $E_K(C[i-1] \oplus M[i])$
Return C[0] || C[1] || … || C[m]

PadCBC unambiguously pads M to a sequence of n bit message blocks

CBC-Dec(K,C):
For i = 1 to m do
    M[i] <- $C[i-1] \oplus D_K(C[i])$
M <- UnpadCBC(M[1],…,M[m],n)
Return M

UnpadCBC removes padding, returns appropriately long string
May output error if padding is wrong
In crypto, errors often denoted by $\perp$

Partial decryption oracles arise frequently in practice

C_1 = Enc(K, “secret data”)

C_2 = Enc(K, “more secrets”)

K

K

C_1

C_3 = Enc(K, “more secrets”)

C’ = modified C_1

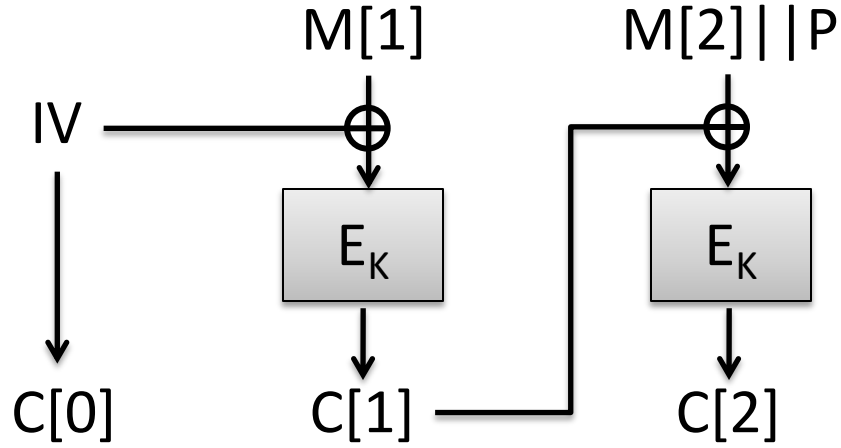
C_4 = Enc(K, “error”)

In practice usually easy to distinguish C_3 from C_4 even without K

$|C_4| \neq |C_3|$
Timing differs for successful vs. unsuccessful decryption

TLS/HTTPS canonical examples where decryption oracles arise

Simple situation: pad by 1 byte



Assume that M[1]||M[2] has length 2n-8 bits

P is one byte of padding that must equal 0x00

Adversary obtains ciphertext C[0],C[1],C[2]

C[0] , C[1] , C[2]

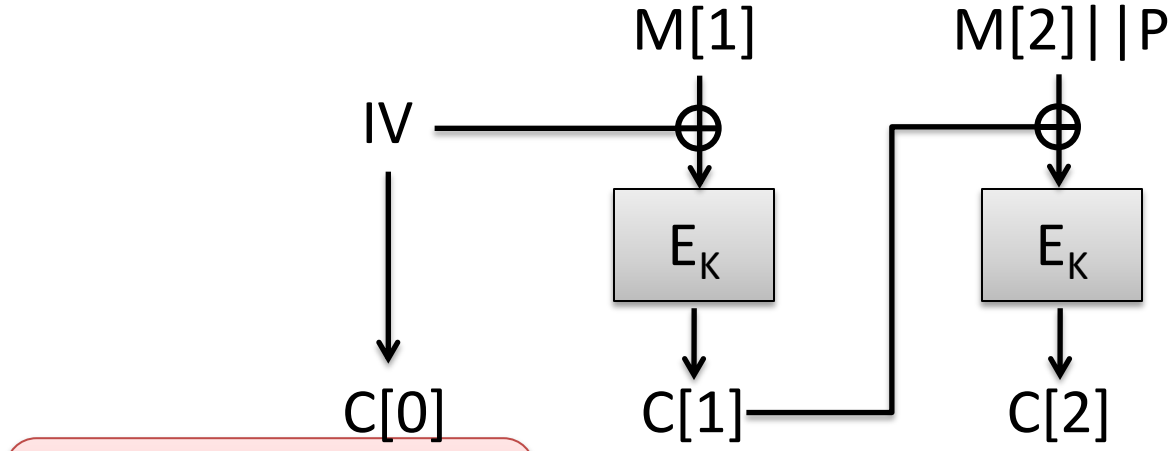
ok

C[0], C[1]⊕1 , C[2]

error

Dec(K, C')
M'[1]||M'[2]||P' = CBC-Dec(K,C')
If P' ≠ 0x00 then
    Return error
Else
    Return ok

Simple situation: pad by 1 byte



Assume that M[1]||M[2] has length 2n-8 bits

P is one byte of padding that must equal 0x00

Low byte of M1 equals i

Adversary obtains ciphertext C[0],C[1],C[2]
Let R be arbitrary n bits

R, C[0], C[1]
error
R , C[0]⊕1 , C[1]
error
R , C[0]⊕2 , C[1]
error
…
R , C[0]⊕i , C[1]
ok

Dec(K, C')
M'[1]||M'[2]||P' = CBC-Dec(K,C')
If P' ≠ 0x00 then
    Return error
Else
    Return ok

PKCS #7 Padding

PKCS#7-Pad(M) = M || P || … || P

P repetitions of byte encoding number of bytes padded

Possible paddings:

01
02 02
03 03 03
04 04 04 04
…
FF FF FF FF … FF

For block length of 16 bytes, don't need more than 16 bytes of padding (10 10 … 10)

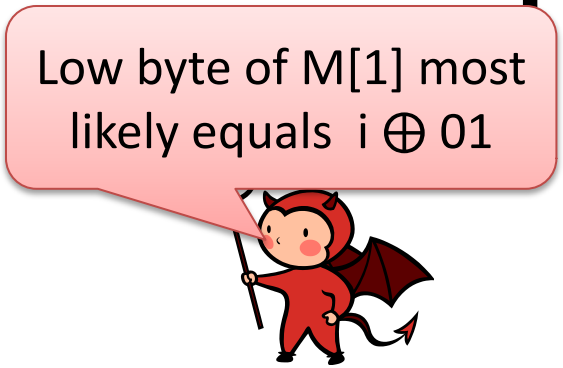
Decryption
(assuming at most one block of padding)

```
Dec( K, C )
M[1] || … || M[m] = CBC-Dec(K,C)
P = RemoveLastByte(M[m])
while i < int(P):
        P’ = RemoveLastByte(M[m])
        If P’ != P then
                Return error
Return ok
```

“ok” / “error” stand-ins for some other behavior:

- Passing data to application layer (web server)
- Returning other error code (if padding fails)

PKCS #7 padding oracles



Adversary obtains ciphertext C[0],C[1],C[2]
Let R be arbitrary n bits

R, C[0] , C[1] →
← error
R , C[0]⊕1 , C[1] →
← error
R , C[0]⊕2 , C[1] →
← error
…
R , C[0]⊕i , C[1] →
← ok

Dec(K, C)
M'[1] || … || M'[m] = CBC-Dec(K,C)
P = RemoveLastByte(M'[m])
while i < int(P):
    P' = RemoveLastByte(M'[m])
    If P' != P then
        Return error
Return ok

Why? Let X[1] = D(K,C1)

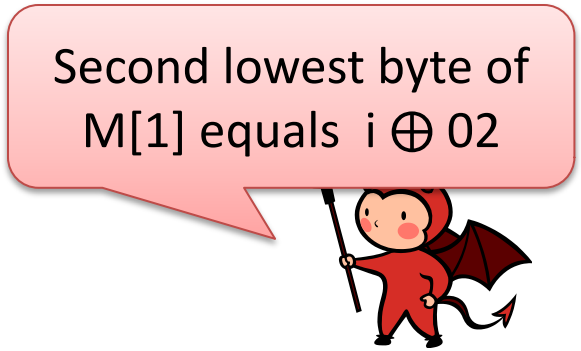
C[0][16] ⊕ X[1][16] = M[1][16]
C[0][16] ⊕ i ⊕ X[1][16] = 01

M[1][16] ⊕ i = 01

Actually, it could be that:
M[1][16] ⊕ i = 02

Implies that M[1][15] = 02
We can rule out with an additional query

PKCS #7 padding oracles



Adversary obtains ciphertext C[0],C[1],C[2]
Let R be arbitrary n bits

R, C[0]⊕0||j , C[1] →
← error
R , C[0]⊕1||j , C[1] →
← error
R , C[0]⊕2||j , C[1] →
← error
…
R , C[0]⊕i||j , C[1] →
← ok

Dec(K, C)
M'[1] || … || M'[m] = CBC-Dec(K,C)
P = RemoveLastByte(M'[m])
while i < int(P):
    P' = RemoveLastByte(M'[m])
    If P' != P then
        Return error
Return ok

Set j = M[1][16] ⊕ 01 ⊕ 02

Keep going to recover entire block of message M[1]
Can repeat with other blocks M[2], M[3], …
Worst case: 256*16 queries per block

Can we change decryption implementation?

```
Dec( K, C )
M[1] || … || M[m] = CBC-Dec(K,C)
P = RemoveLastByte(M[m])
while i < int(P):
        P’ = RemoveLastByte(M[m])
        If P’ != P then
                Return error
Return ok
```

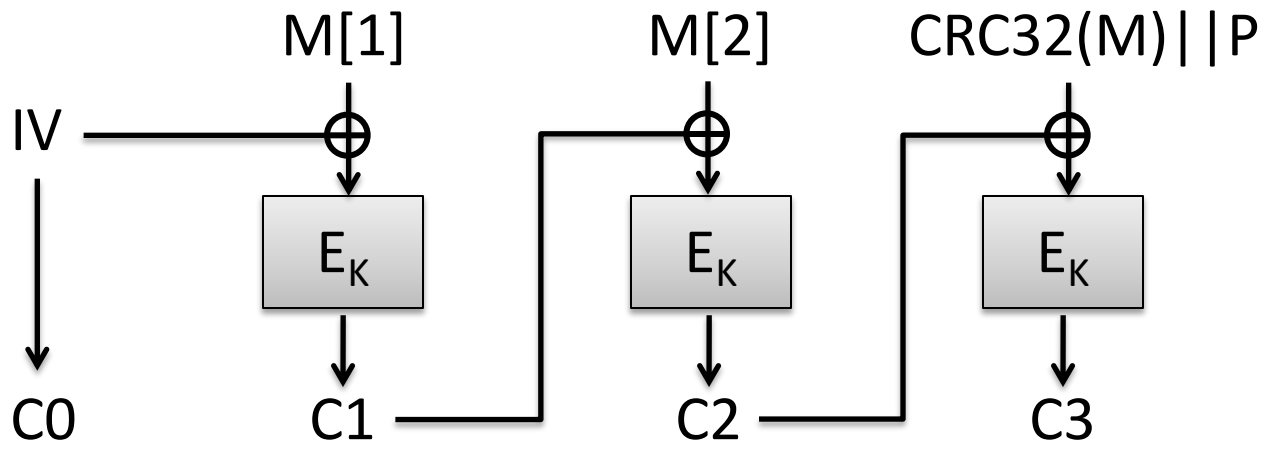
“ok” / “error” stand-ins for some other behavior:

- Passing data to application layer (web server)
- Returning other error code (if padding fails)

Chosen ciphertext attacks against CBC

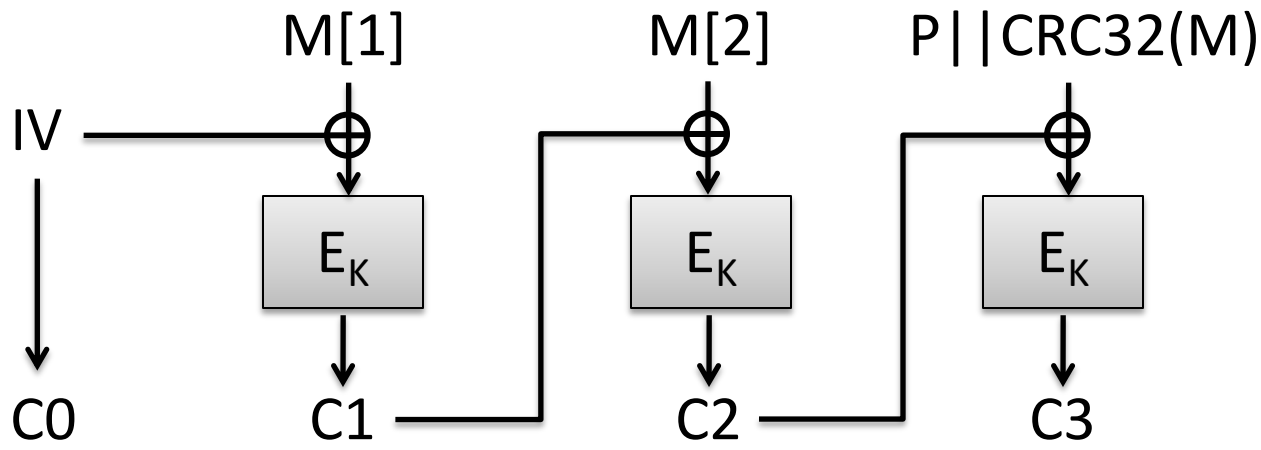
Attack	Description	Year
Vaudenay	10's of chosen ciphertexts, recovers message bits from a ciphertext. Called "padding oracle attack"	2001
Canvel et al.	Shows how to use Vaudenay's ideas against TLS	2003
Degabriele, Paterson	Breaks IPsec encryption-only mode	2006
Albrecht et al.	Plaintext recovery against SSH	2009
Duong, Rizzo	Breaking ASP.net encryption	2011
Jager, Somorovsky	XML encryption standard	2011
Duong, Rizzo	"Beast" attacks against TLS	2011
AlFardan, Paterson	Attack against DTLS	2012
AlFardan, Paterson	Lucky 13 attack against DTLS and TLS	2013
Albrecht, Paterson	Lucky microseconds against Amazon's s2n library	2016

Non-cryptographic checksums?



CRC32(M) is cyclic redundancy code checksum.
Probabilistically catches random errors
Decryption rejects if checksum is invalid

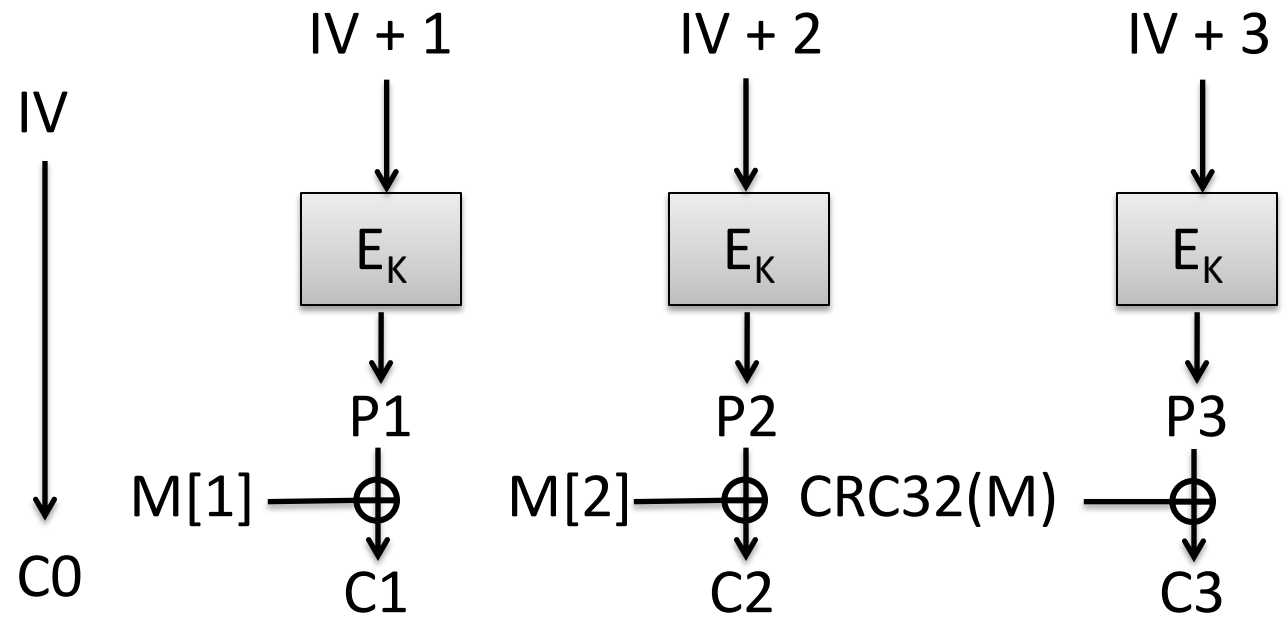
Non-cryptographic checksums?



CRC32(M) is cyclic redundancy code checksum.
Probabilistically catches random errors
Decryption rejects if checksum is invalid

Wagner sketched partial chosen plaintext, chosen ciphertext attack
(see Vaudenay 2002 paper)

Non-cryptographic checksums?



Can simply maul message and CRC32 checksum to ensure correctness

None of these modes secure for general-purpose encryption

- CTR mode and CBC mode fail in presence of active attacks
  - Cookie example
  - Padding oracle attacks

- Need authentication mechanisms to help prevent chosen-ciphertext attacks

Brief digression: need for per-message randomness

- CTR mode uses a per-message random IV
- Deterministic symmetric encryption:
  - Enc(K,M) = Enc(K,M') iff M = M'
    - In other words, ciphertexts leak (at least) plaintext equality
  - ECB mode is an old, deprecated example. Leaks *much more* than plaintext equality
  - Other examples in wider use that leak just if M = M':
    - format-preserving encryption (FPE), synthetic IV mode (SIV), …
  - Must be very careful using these, as repetitions can be useful for ***frequency analysis***