Today in Cryptography (5830)

CBC mode
Padding oracle attacks

Recap: Block ciphers, length-preserving & length-extending encryption

Block cipher is a map $E : \{0,1\}^k \times \{0,1\}^n \dashrightarrow \{0,1\}^n$

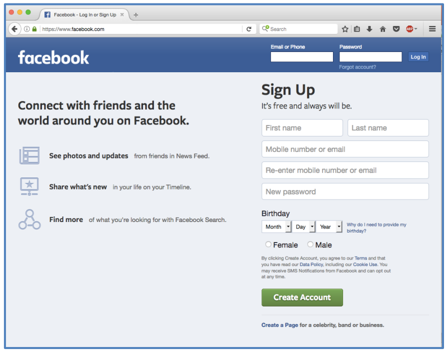
Length-preserving encryption
- Useful in practice in legacy settings
- Use Feistel networks to construct

(Randomized) length-extending encryption
- CTR mode
- CBC mode

Modes are ***not*** secure against chosen-ciphertext attacks

Today: chosen-ciphertext attacks break confidentiality

Session handling and login



GET /index.html

Set-Cookie: AnonSessID=134fds1431

Protocol is HTTPS. Elsewhere just HTTP

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

Cookie: AnonSessID=134fds1431

Set-Cookie: SessID=83431Adf

GET /account.html

Cookie: SessID=83431Adf

Security problems here?

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

Cookie: AnonSessID=134fds1431

Facebook.com

Set-Cookie: SessID=83431Adf

Secret key K_c only known to server

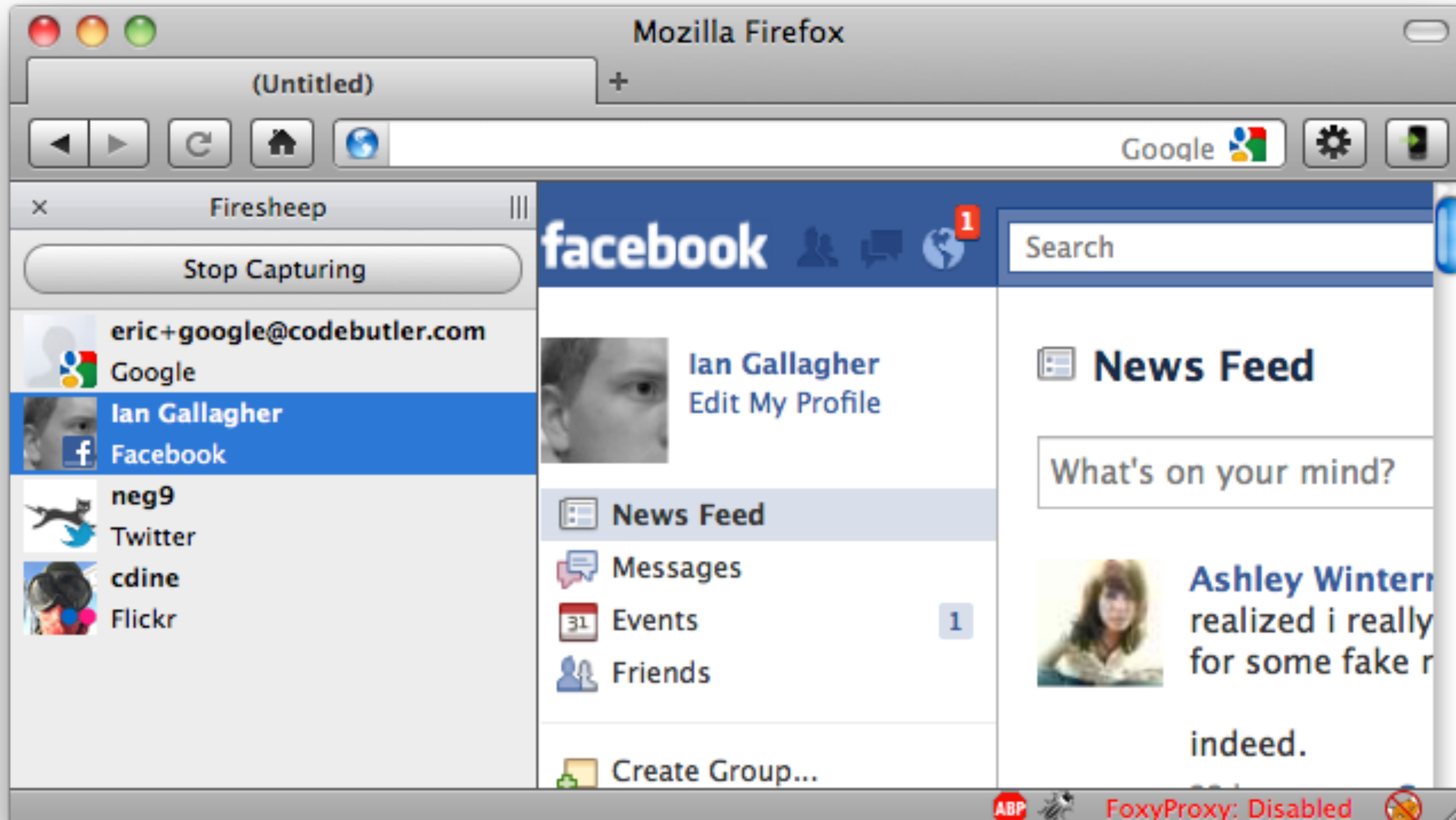
GET /account.html

Cookie: SessID=83431Adf

$$83431\text{Adf} = \text{CTR}[E](K_c, \text{“admin=0”})$$

- Network adversary can modify SessID ciphertext
- Malicious client can modify SessID ciphertext
- Network adversary can steal SessID

Session Hijacking



From http://codebutler.com/firesheep

Security problems here?

Facebook.com

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

Cookie: AnonSessID=134fds1431

Set-Cookie: SessID=83431Adf

Secret key K_c only known to server

GET /account.html

Cookie: SessID=83431Adf

Use HTTPS for all communications

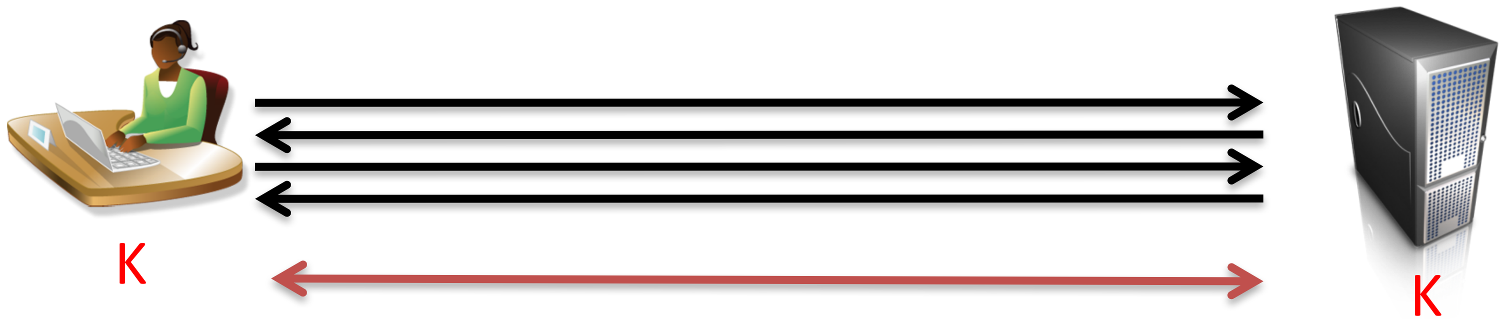
83431Adf = CTR[E](K_c, "admin=0")

All things still possible if HTTPS record layer encryption is bad!!!

- Network adversary can modify SessID ciphertext
- Malicious client can modify SessID ciphertext
- Network adversary can steal SessID

How TLS works (high level view)

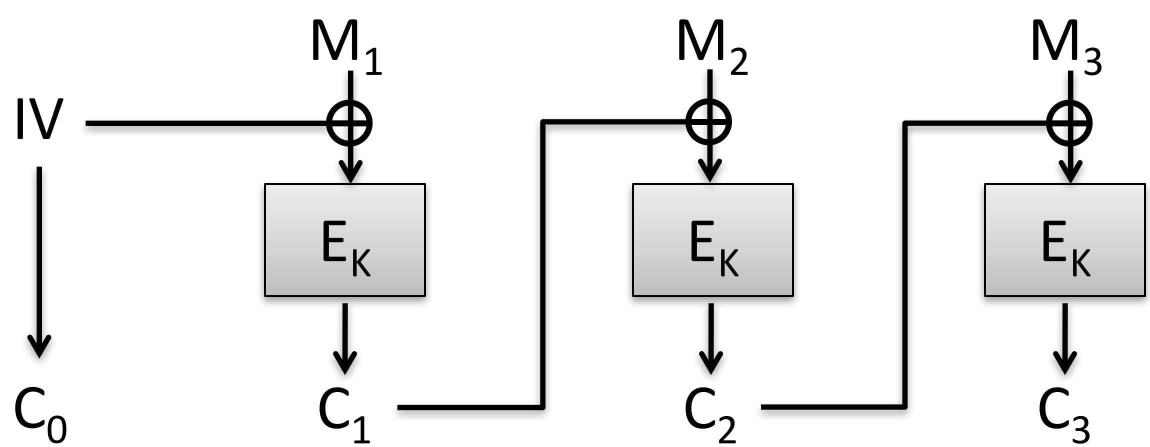
https://facebook.com



Step 1:
Key exchange protocol to share secret K

Step 2:
Send data via secure channel

Let’s suppose TLS uses just CBC-mode for symmetric encryption
M = “Cookie: SessID=83431Adf”



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 (up to some large max length)

- Padding checks often give rise to padding oracle attacks as we will see

PKCS #7 Padding

$$\text{PKCS\#7-Pad}(M) \quad = \quad M \,||\, \underbrace{P \,||\, \ldots \,||\, P}_{\text{P repetitions of byte encoding number of bytes padded}}$$

Possible paddings:

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

For block length of 16 bytes, never need more than 16 bytes of padding (10 10 … 10)

Always pad, even if M is multiple of n bytes. Why?

Decryption

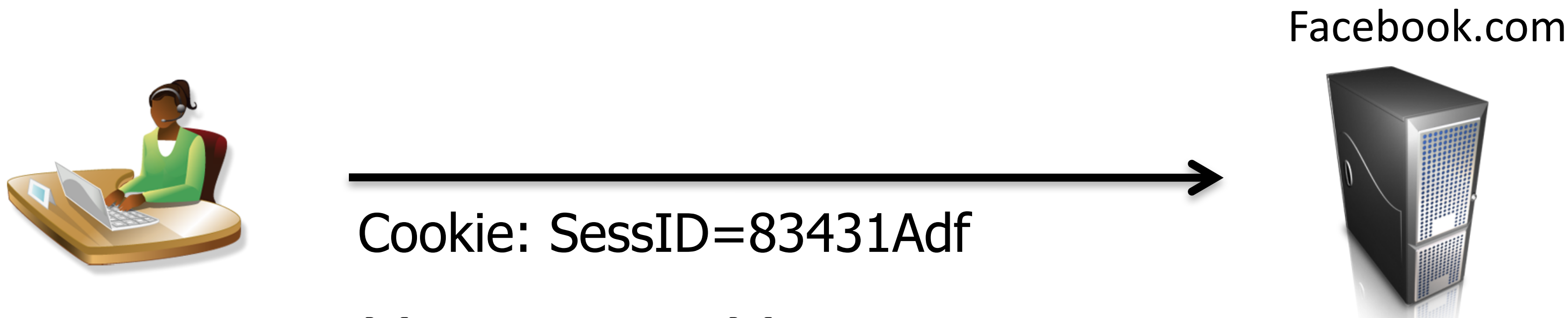
<u>Dec(K, C)</u>

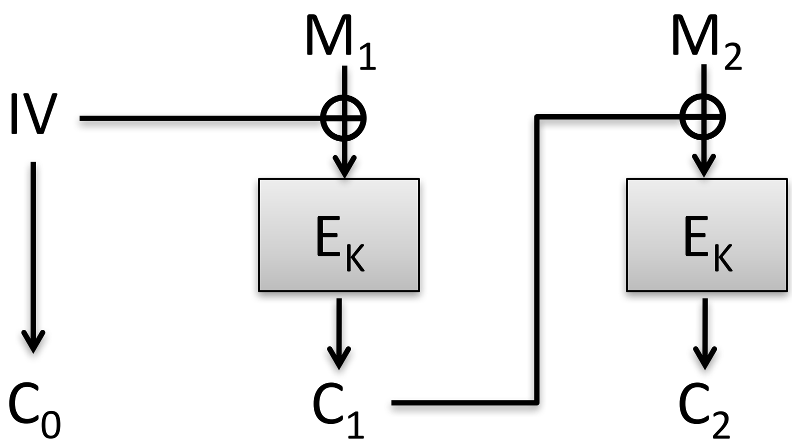
```
Dec( K, C )
M_1 || … || M_L = CBC-Dec(K,C)
P = RemoveLastByte(M_L)
while i < int(P):
      P' = RemoveLastByte(M_L)
      If P' != P then Return error
      i = i + 1
Return ok
```

“Ok” is a stand-in for some other behavior:

- Passing data to application layer (web server)
- Returning other error code (if data is junk)

Security problems here?





M_1 = “ SessID=83431Adf”
M_2 = 10 10 10 … 10

<u>Dec(K, C)</u>
M_1 || … || M_L = CBC-Dec(K,C)
P = RemoveLastByte(M_L)
while i < int(P):
    P’ = RemoveLastByte(M_L)
    If P’ != P then Return error
    i = i + 1
Return ok

Attacker gets C_0, C_1, C_2
Attacker can send to server any ciphertexts it wants and see return values

- Modify bits of any block
- Reorder blocks
- Make up new blocks

PKCS #7 padding oracles

$M_1[16] = i \oplus 01$

Adversary obtains ciphertext $C = C_0, C_1, C_2$

Let R be arbitrary n bits

$R, C_0, C_1 \rightarrow$

$\leftarrow$ error

$R, C_0 \oplus 1, C_1 \rightarrow$

$\leftarrow$ error

$R, C_0 \oplus 2, C_1 \rightarrow$

$\leftarrow$ error

…

$R, C_0 \oplus i, C_1 \rightarrow$

$\leftarrow$ ok

<u>Dec(K, C)</u>

$M_1 \,||\, \ldots \,||\, M_L$ = CBC-Dec(K,C)

P = RemoveLastByte(M_L)

while i < int(P):

    P’ = RemoveLastByte(M_L)

    If P’ != P then Return error

    i = i + 1

Return ok

<u>Why?</u>

$C_0[16] \oplus X_1[16] = M_1[16]$

$C_0[16] \oplus i \oplus X_1[16] = 01$

Implies: $M_1[16] \oplus i = 01$

Actually, it could be that:

$M_1[16] \oplus i = 02$

Implies that $M_1[15] = 02$

We can rule out with an additional query

PKCS #7 padding oracles

$M_1[15] = i \oplus 02$

R, C_0, C_1 →

← error

$R, C_0 \oplus (1||j), C_1$ →

← error

$R, C_0 \oplus (2||j), C_1$ →

← error

…

$R, C_0 \oplus (i||j), C_1$ →

← ok

Adversary obtains ciphertext
$C = C0,C1,C2$
Let R be arbitrary n bits

Set $j = i \oplus 01 \oplus 02$

<u>Dec(K, C)</u>

$M_1 || \ldots || M_L = \text{CBC-Dec}(K,C)$
$P = \text{RemoveLastByte}(M_L)$
while $i < \text{int}(P)$:
    $P' = \text{RemoveLastByte}(M_L)$
    If $P' \mathrel{!=} P$ then Return error
    $i = i + 1$
Return ok

History of padding oracle attacks

Author(s)	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

Good write-up of ASP.net vulnerability + exploit:
https://www.troyhunt.com/fear-uncertainty-and-and-padding-oracle/

Active chosen-ciphertext attacks can break confidentiality

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

- Next lecture: adding authentication mechanisms to prevent chosen-ciphertext attacks