Cryptography, part 2

CS5435: Security and Privacy (in the wild?)

Paul Grubbs

http://www.cs.cornell.edu/~paulgrubbs/

pag225 at cornell dot edu

Liberal borrowing from Ristenpart, Wisc CS642 and Mitchell, Boneh, Stanford CS 155

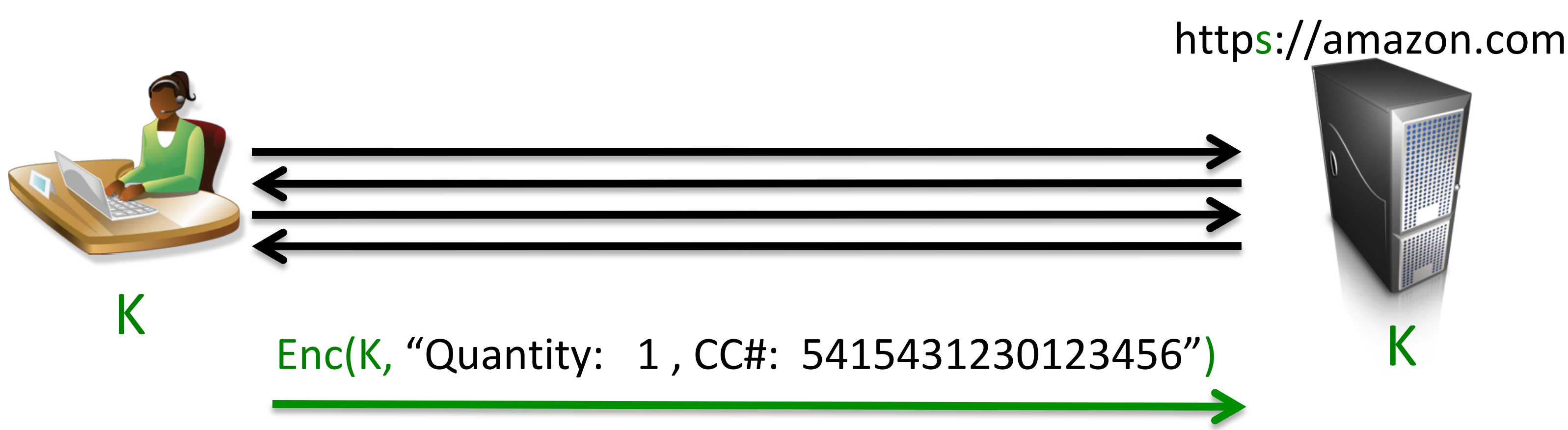
Today’s lecture

- Block cipher modes of operation
- Attacking insecure approaches
- Message authentication
  - fixing (some) problems
- Authenticated encryption
  - fixing (still more) problems

Recall setting

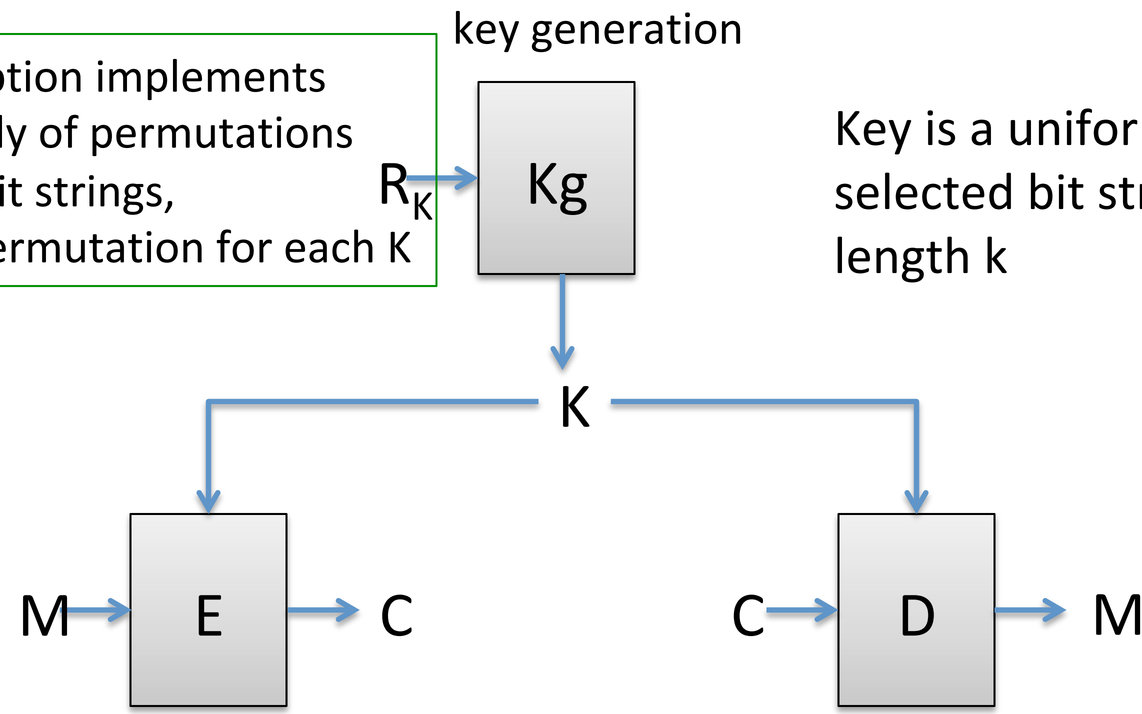
Two or more parties agree on a secret random value, want to keep communication secret.

Idea: use **symmetric encryption** to scramble messages, using shared random value



Block ciphers

Encryption implements
a family of permutations
on n bit strings,
one permutation for each K



Key is a uniformly
selected bit string of
length k

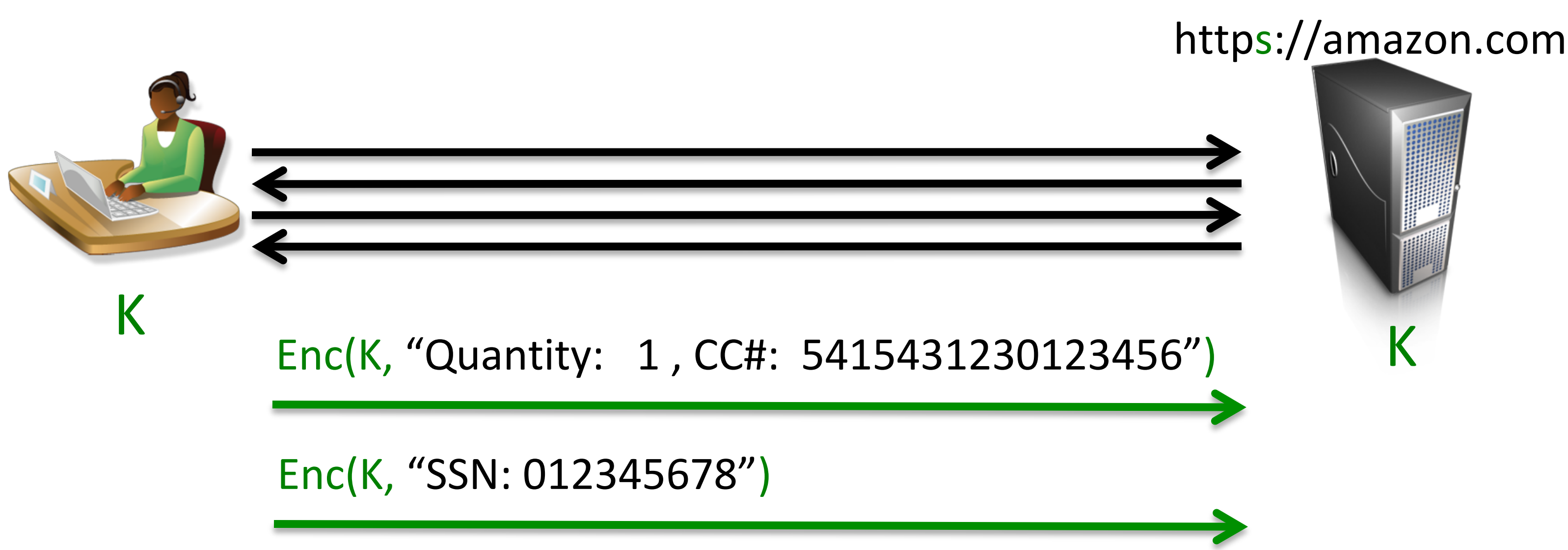
E/D: $\{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$

Security goal: E(K,M) is indistinguishable from random n-bit string for anyone without K (E is ***pseudorandom function/PRF***)

Are we done?

Unfortunately no − messages can be different sizes,
but block ciphers have fixed-length inputs and outputs!

Need ***mode of operation:***
fixed-length block cipher → variable-length encryption scheme.



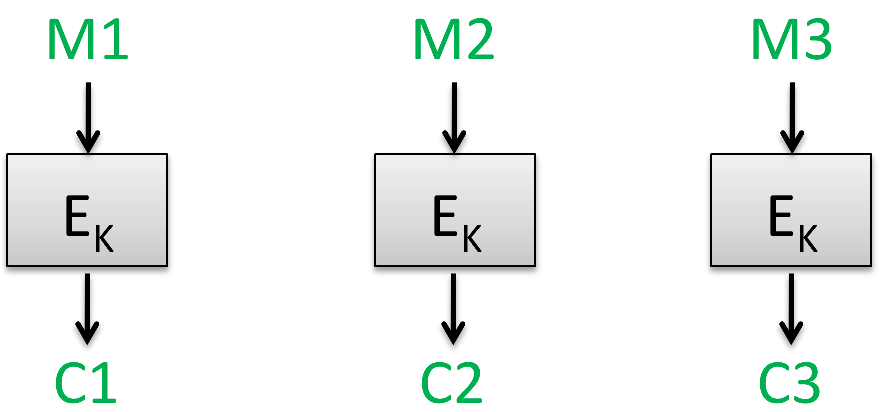
Block cipher modes of operation

Why don’t we apply BC on each (maybe padded) block?

Electronic codebook (ECB) mode

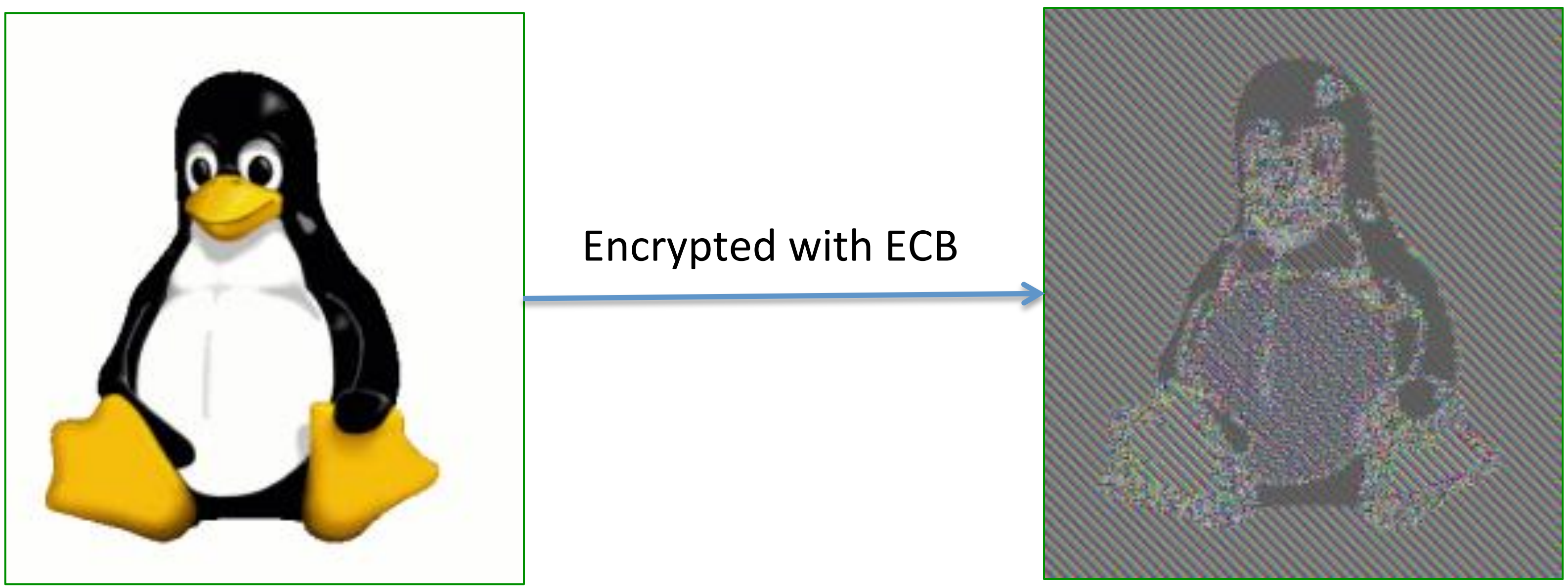
Pad message M to M1,M2,M3,... where each block Mi is n bits

Then:



ECB mode is a more complicated looking substitution cipher

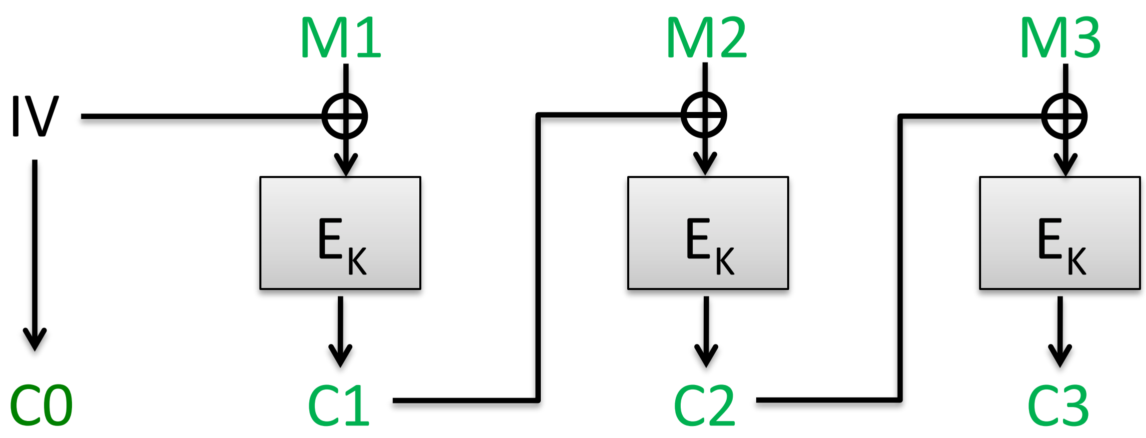
Recall our credit-card number example.
ECB: substitution cipher with alphabet n-bit strings instead of digits



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

CBC mode

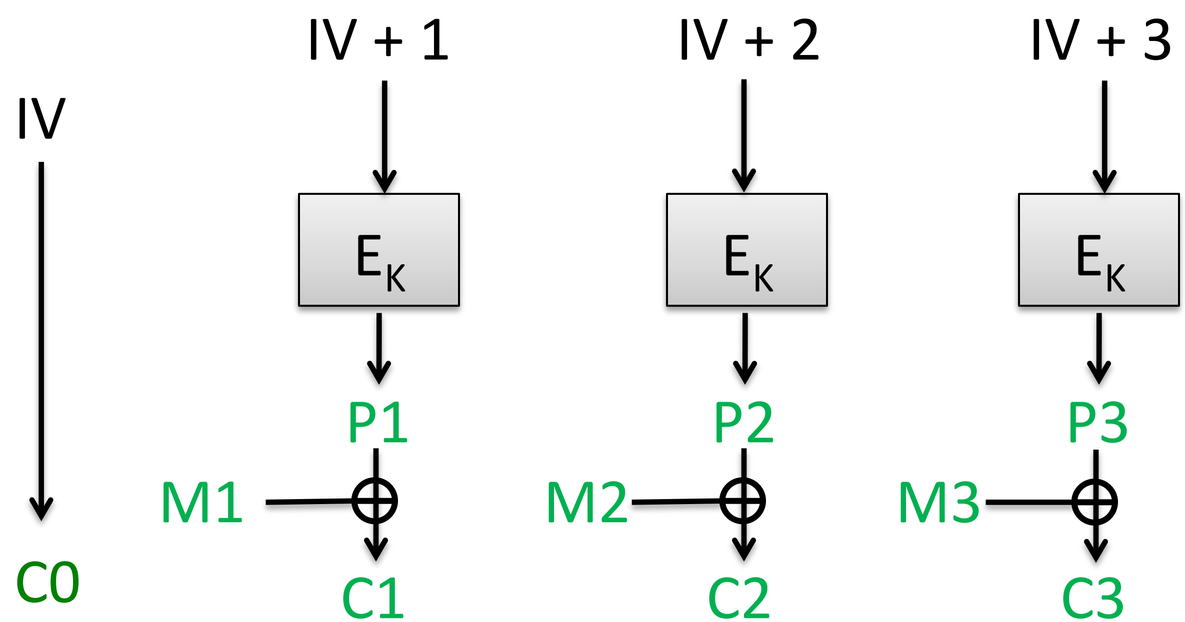
Ciphertext block chaining (CBC)
Pad message M to M1,M2,M3,... where each block Mi is n bits
Choose random n-bit string IV
Then:



How do we decrypt?

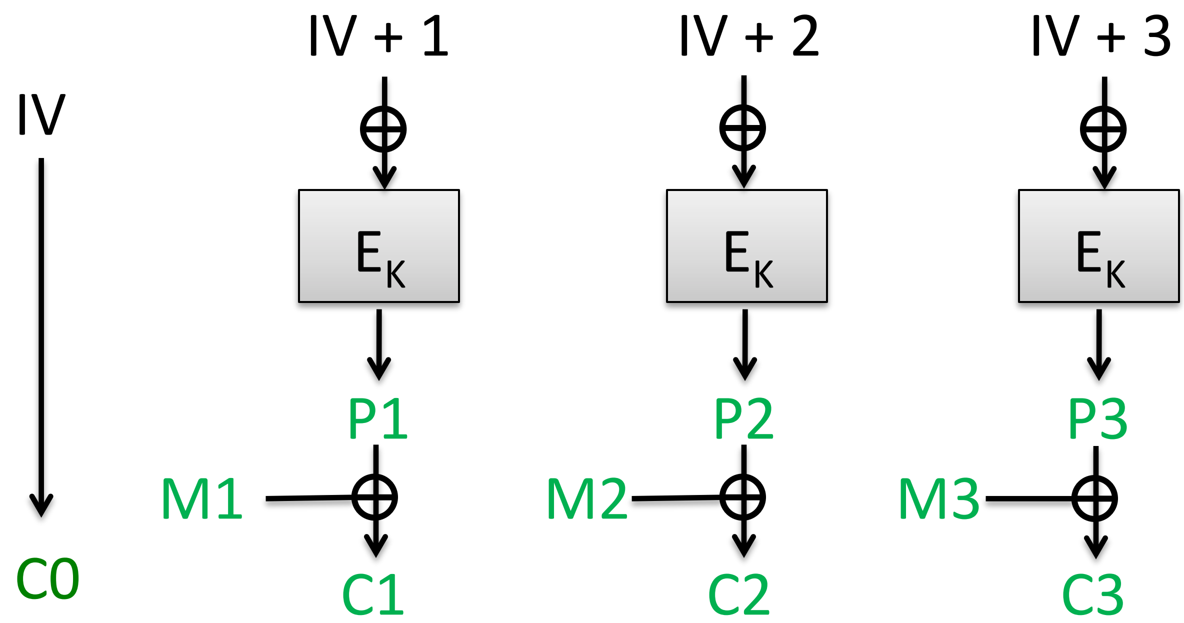
"OTP" encryption

Counter mode (CTR)
Pad message M to M1,M2,M3,... where each is n bits except last
Choose random n-bit string IV
Then:



Maybe use less than full n bits of P3

How do we decrypt?



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

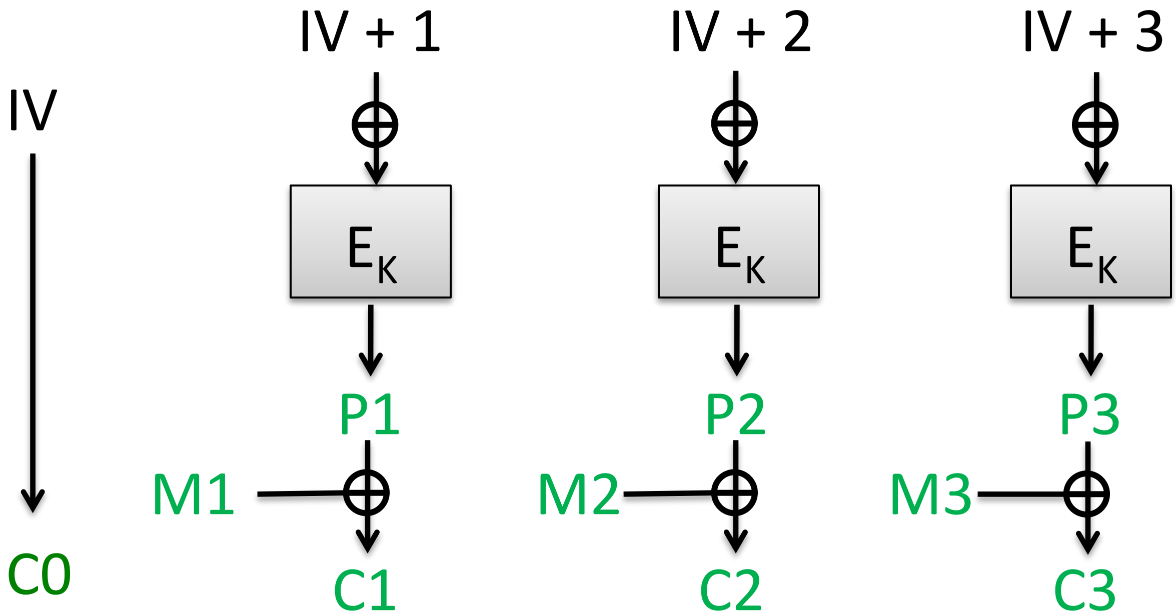
Implies attacker can break E, i.e. recover block cipher key

Can attacker learn M = M1,M2,M3 from C0,C1,C2,C3?

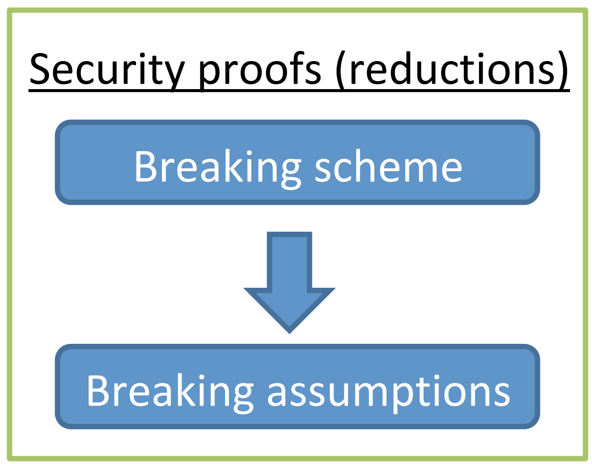
Implies attacker can invert the block cipher without knowing K

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

Implies attacker can break PRF security of E



<u>Theorem (informal).</u>
Let A be a successful, efficient attacker against security of CTR mode. Then there exists a PRF adversary B against E that is efficient and successful.



Attacker can not break CTR confidentiality

Attacker can break CTR confidentiality

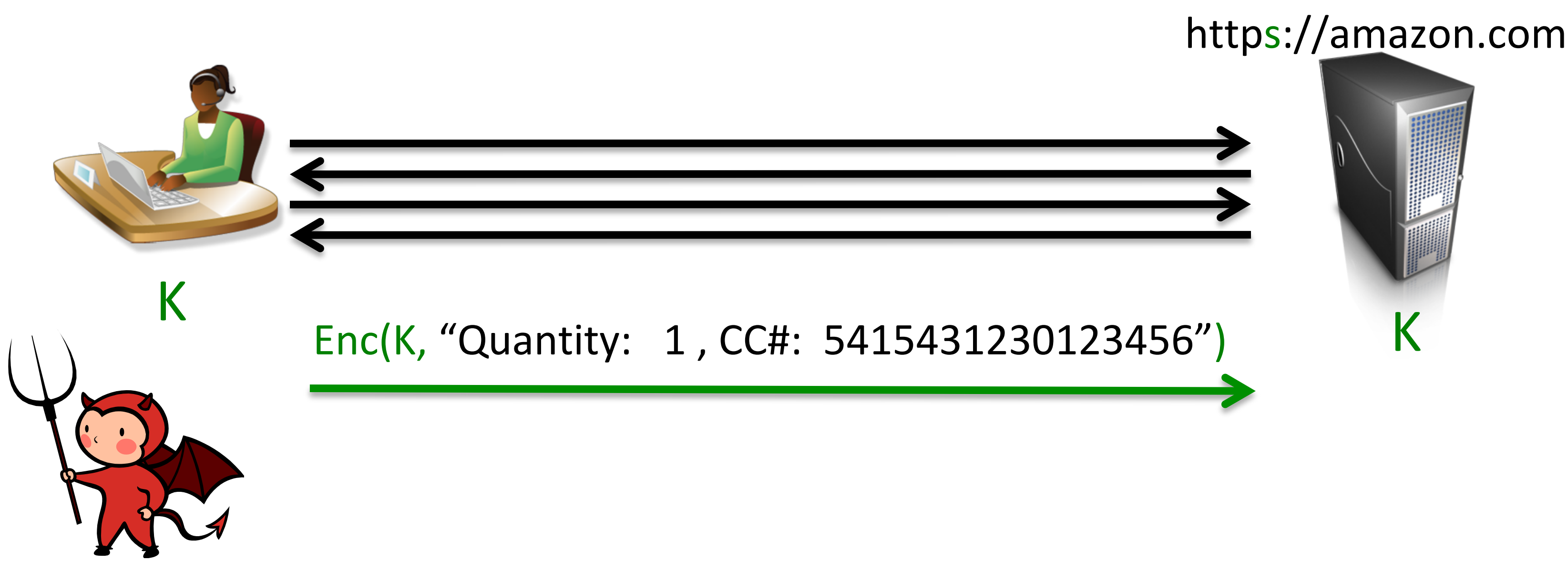
Can not break E PRF sense

Can break E in PRF sense

Reduces analysis now to E and to security definition / model

Are we done?

Still no! Why? Attacker can change message…

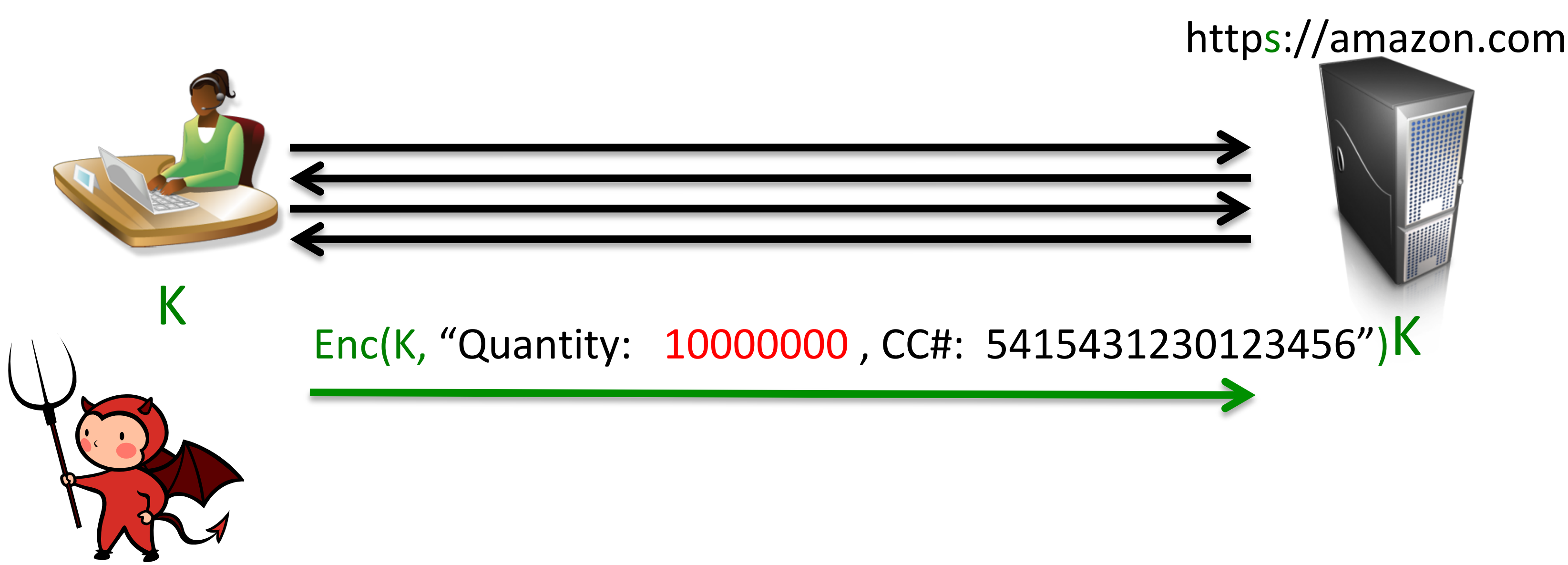


Are we done?

Still no! Why? Attacker can change message…

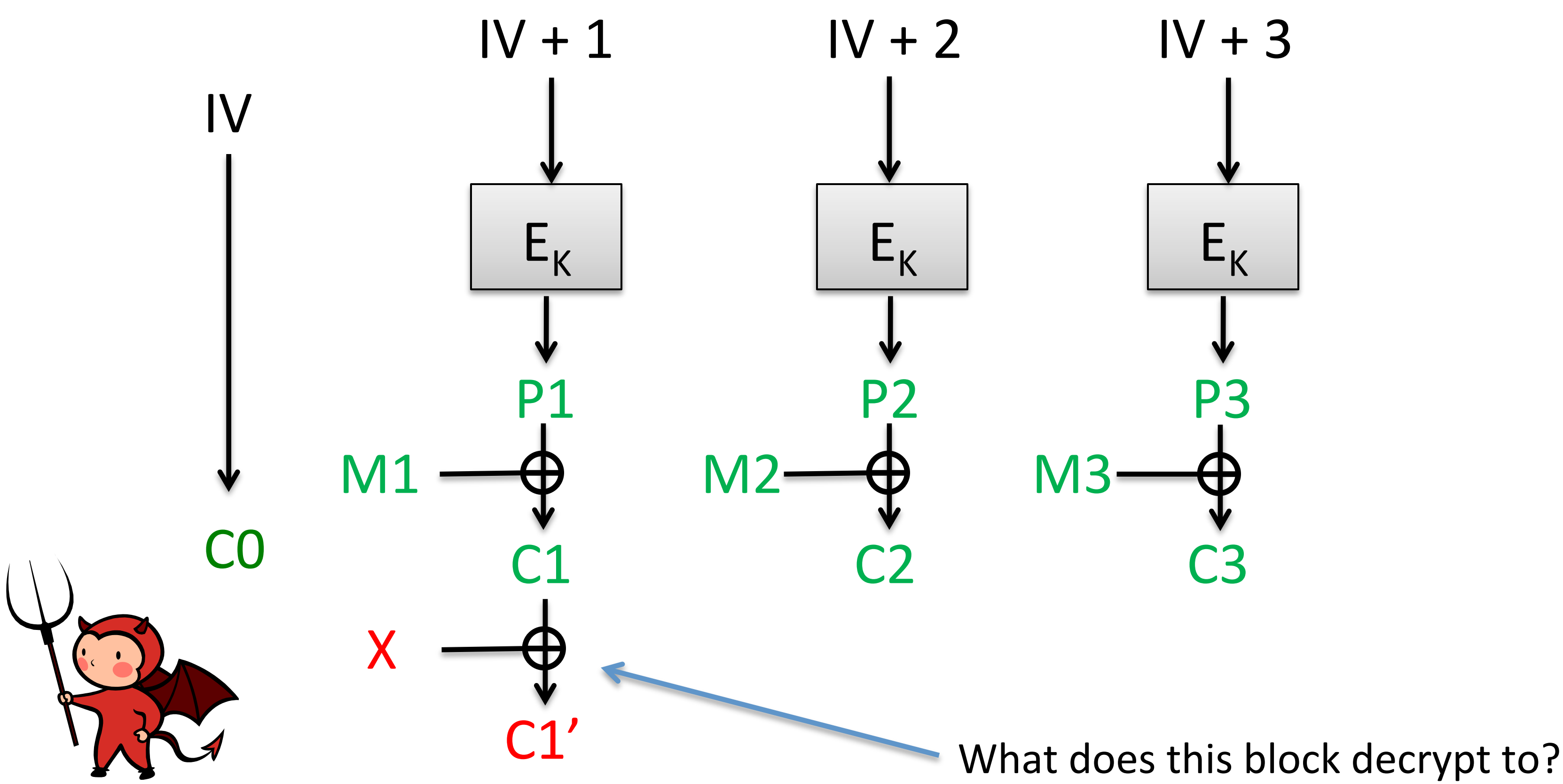
Need to prevent modifications of message in transit!

How can attacker modify messages for CBC and CTR mode?

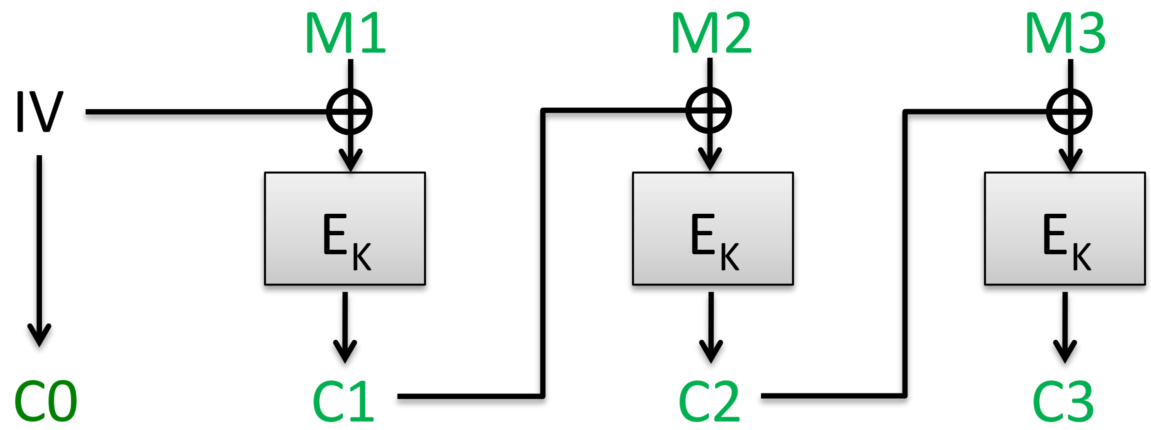


CTR mode malleability

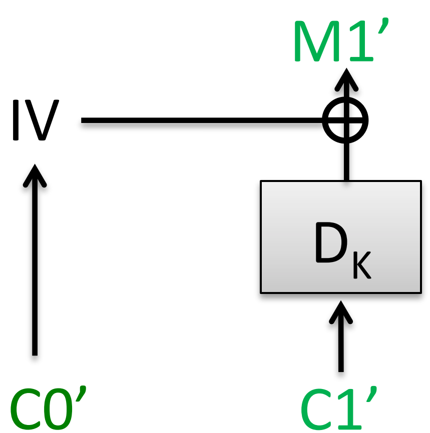
Change message contents via XOR



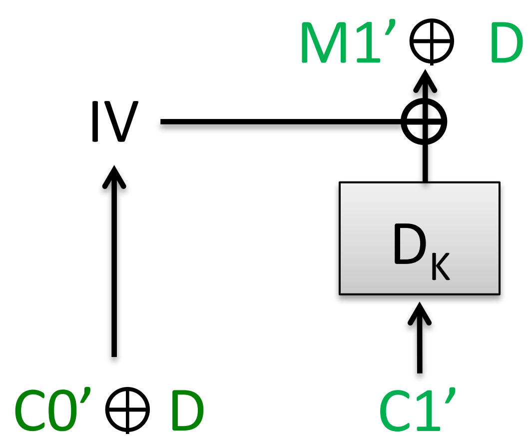
Active security of CBC mode



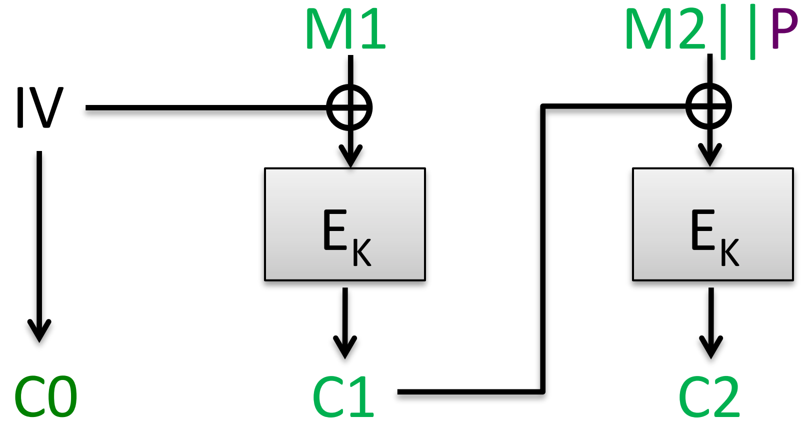
What about forging a message? Pick any C0’, C1’ …



Better yet
for any D:



Padding oracle attack



Assume that M1||M2 has length 2n-8 bits

P is one byte of padding that must equal 0x00

Adversary obtains Ciphertext C0,C1,C2

C0 , C1 , C2

ok

C0, C1⊕1 , C2

error

Dec(K, C')

M1'||M2'||P' = CBC-Dec(K,C')

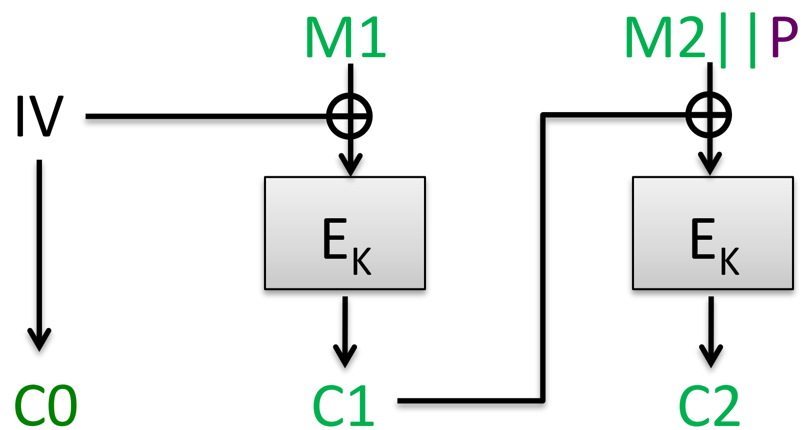
If P' ≠ 0x00 then

  Return error

Else

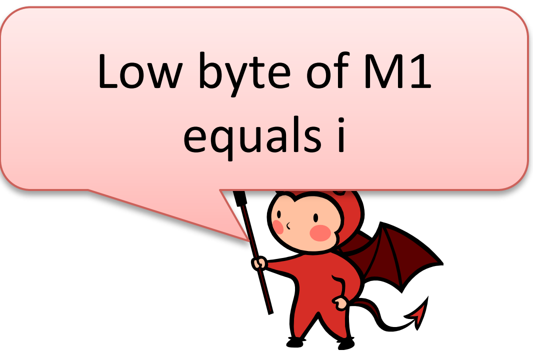
  Return ok

Padding oracle attack



Assume that M1||M2 has length 2n-8 bits

P is one byte of padding that must equal 0x00



R, C0 , C1 →
← error

R , C0 ⊕ 1 , C1 →
← error

R , C0 ⊕ 2 , C1 →
← error

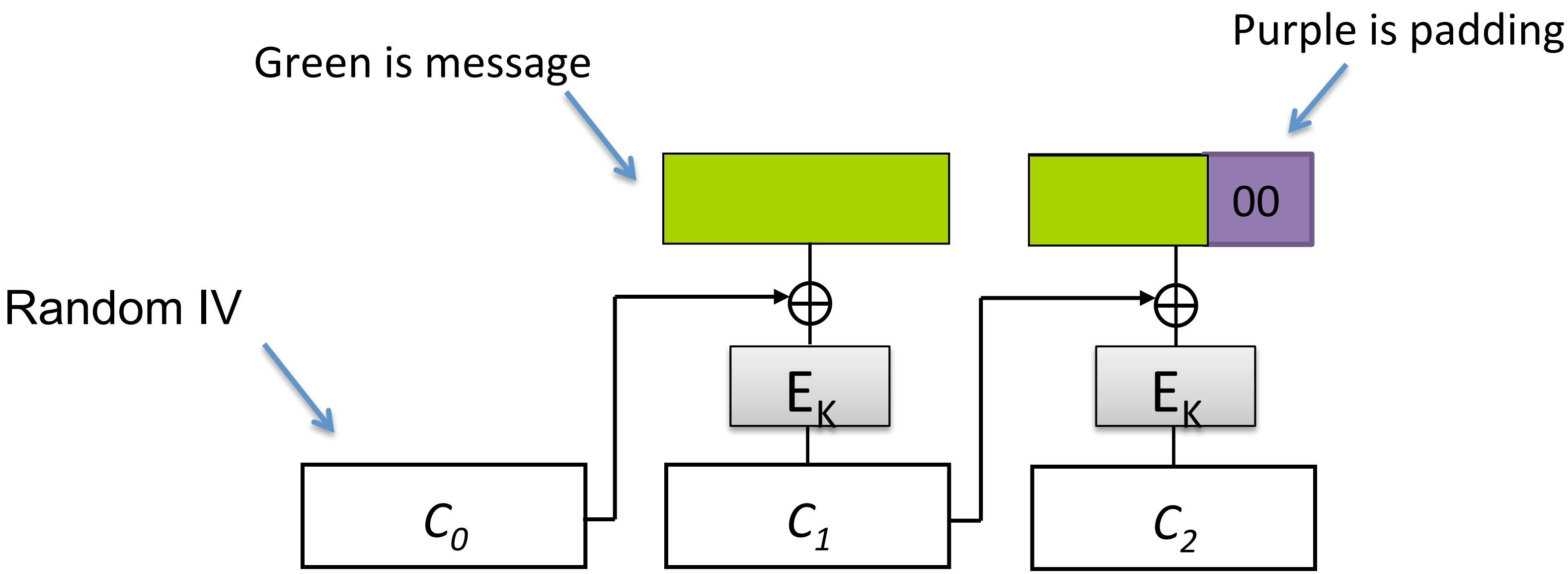
…

R , C0 ⊕ i , C1 →
← ok

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

<u>Dec(K, C’)</u>
M1’||M2’||P’ = CBC-Dec(K,C’)
If P’ ≠ 0x00 then
    Return error
Else
    Return ok

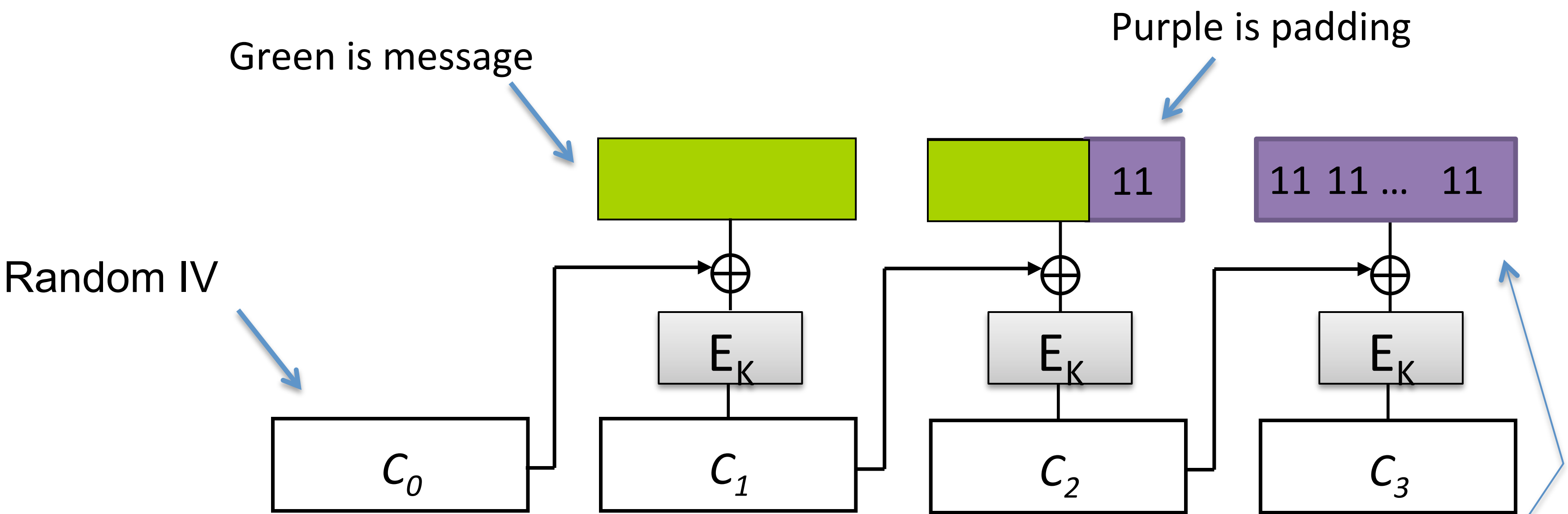
Padding for CBC Mode in TLS



Possible paddings in TLS:

00   01 01   02 02 02   etc.

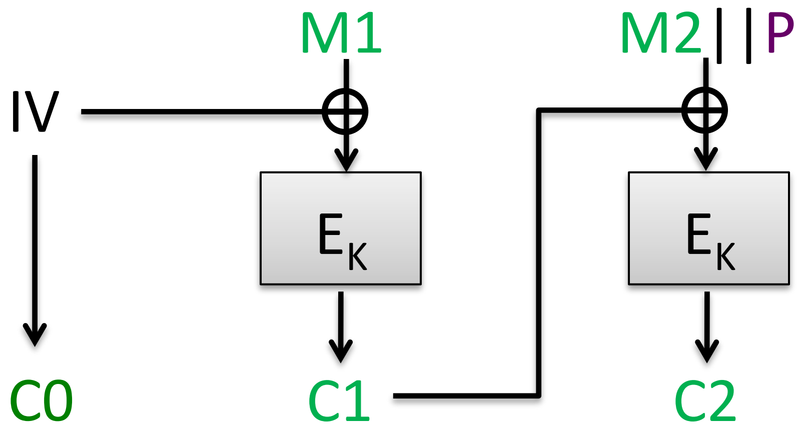
Padding for CBC Mode in TLS



Possible paddings in TLS:
00 01 01 02 02 02 etc.

"Lengths longer than necessary might be desirable to frustrate attacks on a protocol that are based on analysis of the lengths of exchanged messages." RFC 5246

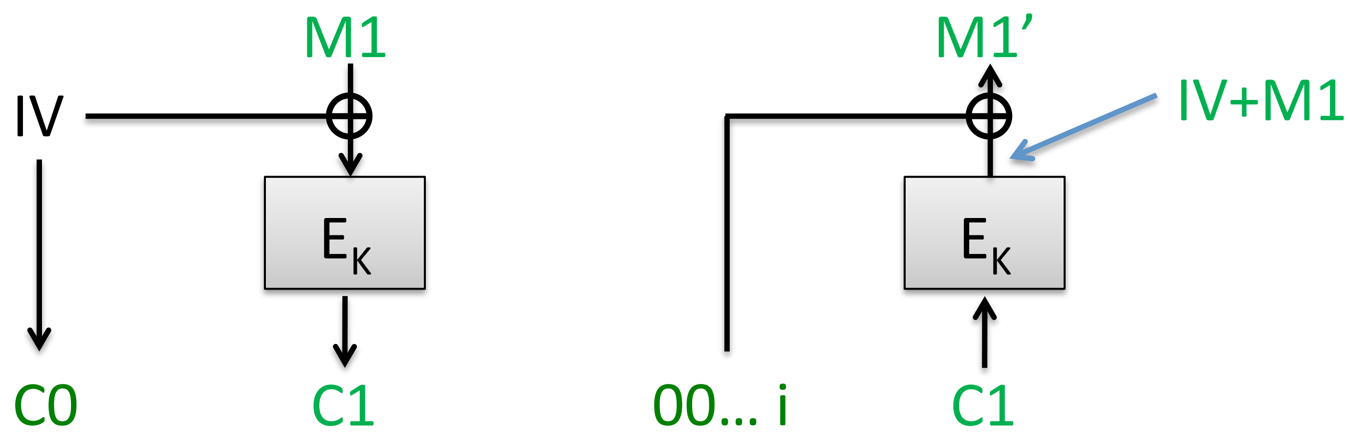
Vaudenay’s padding oracle attack



Goal:
Decrypt entire plaintext

I see this topic in your future…

Vaudenay's padding oracle attack



We know that:

00 = i + IV[n] + M1[n]

Or do we? Could be:

01 = i + IV[n] + M1[n]

01 = IV[n-1] + M1[n-1]

Easy to exclude other cases

00…00, C1

error

00…01 , C1

error

00…02 , C1

error

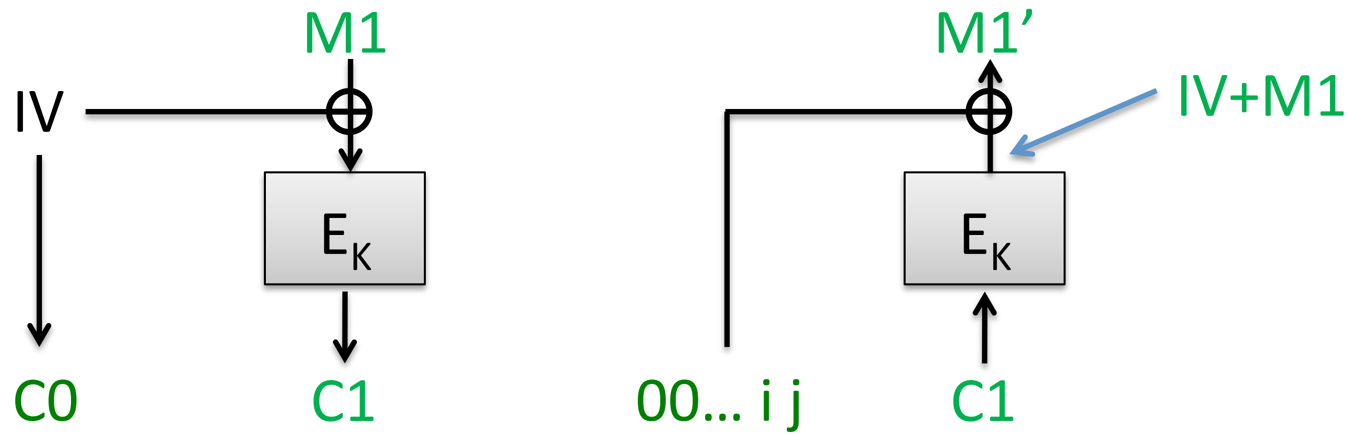
…

00… i, C1

ok

<u>Dec(K, C')</u>
M1' = CBC-Dec(K,C')
(X,plen) <- lastbyte(M1')
For i = 0 to padlen do
    (X,plen') <- lastbyte(X)
    If plen' != plen
        Return **Error**
Return Ok

Vaudenay's padding oracle attack



We know M1[n]. Let's get second to last byte.

Solve j to make M1'[n] = 01
01 = j + IV[n] + M1[n]

Know that:
01 = i + IV[n-1] + M1[n-1]

Repeat for all n bytes

00...00 j , C1 →
← error

00...01 j, C1 →
← error

00...02 j , C1 →
← error

...

00...i j, C1 →
← ok

<u>Dec(K, C')</u>
M1' = CBC-Dec(K,C')
(X,plen) <- lastbyte(M1')
For i = 0 to padlen do
    (X,plen') <- lastbyte(X)
    If plen' != plen
        Return **Error**
Return Ok

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

In-class exercise

- Take five minutes and discuss with your neighbor…
  - Are padding oracles possible against CTR mode?
  - Same question, but for ECB (this is subtle…)
  - How would you prevent padding oracle attacks? Cryptographic countermeasure or “system-level”?

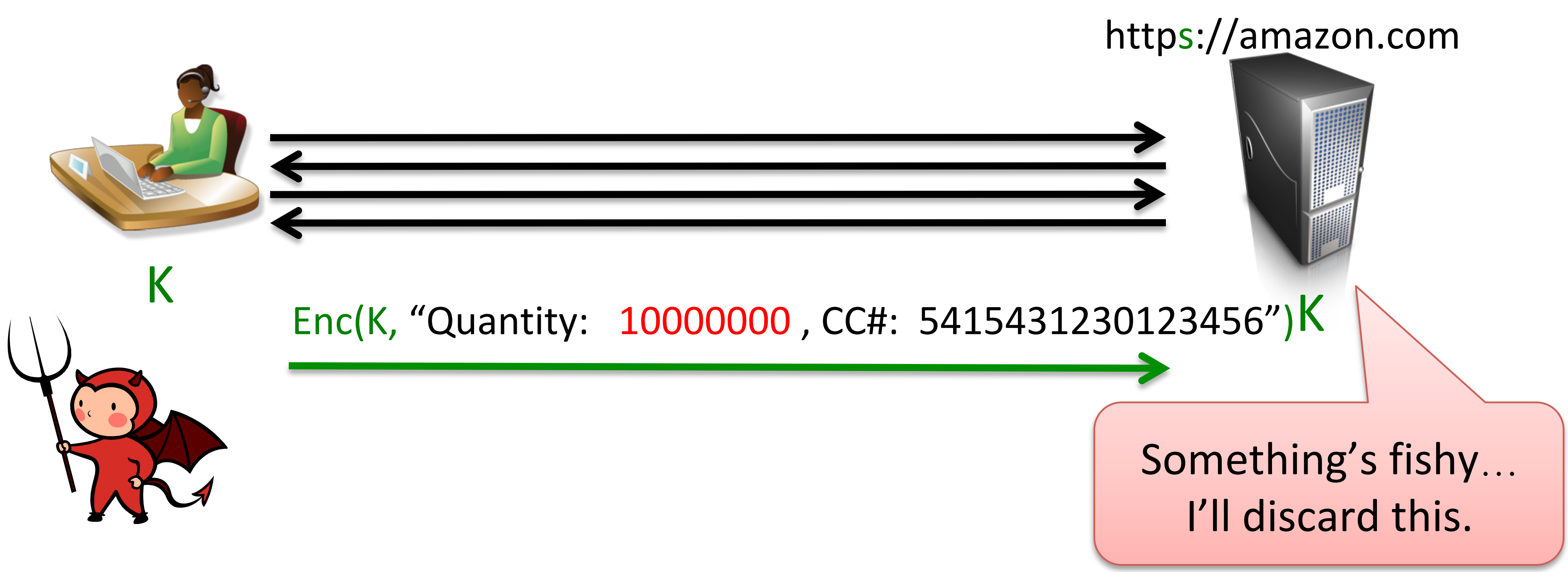
Are we done?

Message authentication prevents this by making modifications detectable.

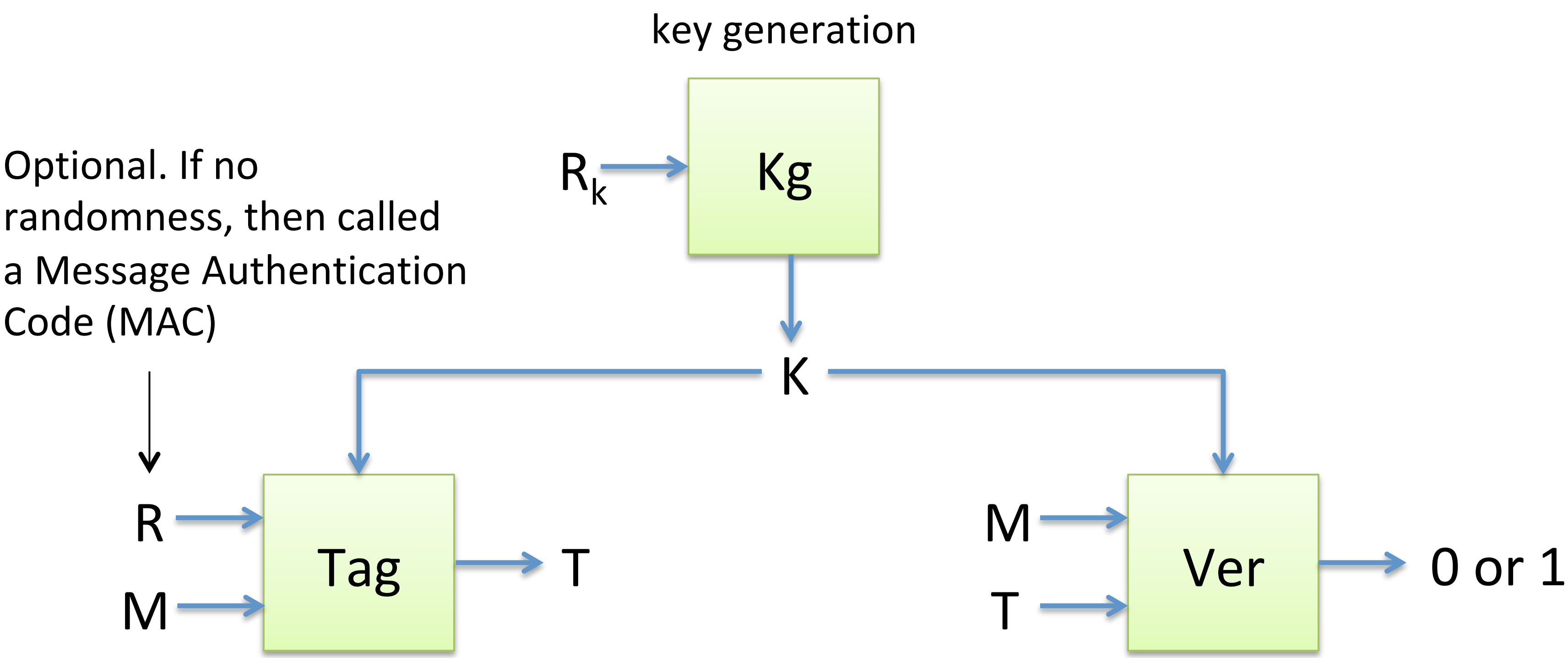
Goal is ***Authenticated Encryption*** (AE):

Hide message and detect modifications

Can build by combining encryption with a symmetric authentication primitive



Message authentication

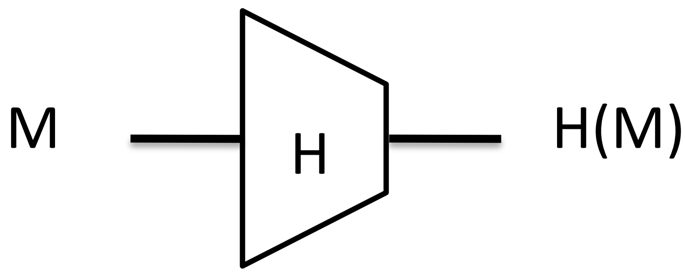


Correctness: Ver(K , Tag(K,M,R)) = 1 with probability 1 over randomness used

Unforgeability: Attacker can't find M',T such that V(K,M',T) = 1

Hash functions and message authentication

Hash function H maps arbitrary bit string to fixed length string of size m



~~MD5: m = 128 bits~~
~~SHA-1: m = 160 bits~~
SHA-256: m = 256 bits

Some security goals:

- collision resistance: can’t find M != M’ such that H(M) = H(M’)
- preimage resistance: given H(M), can’t find M
- second-preimage resistance: given H(M), can’t find M’ s.t. H(M’) = H(M)

Birthday paradox

- Anyone heard of this?
- Generic upper bound on collision resistance of hash function

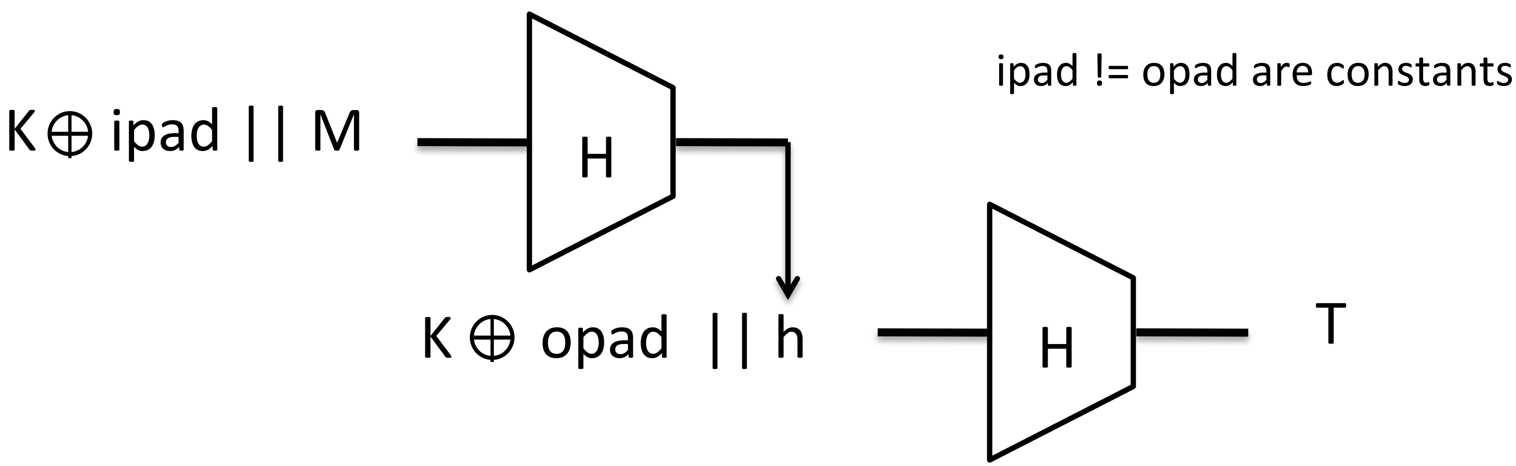
<u>Thm</u>: For random function of output size N,
need ~ square root N inputs to find a collision

I see this topic in your future…

Message authentication with HMAC

Use a hash function H to build MAC.
Kg outputs uniform bit string K

Tag(K,M) = HMAC(K,M) defined by:

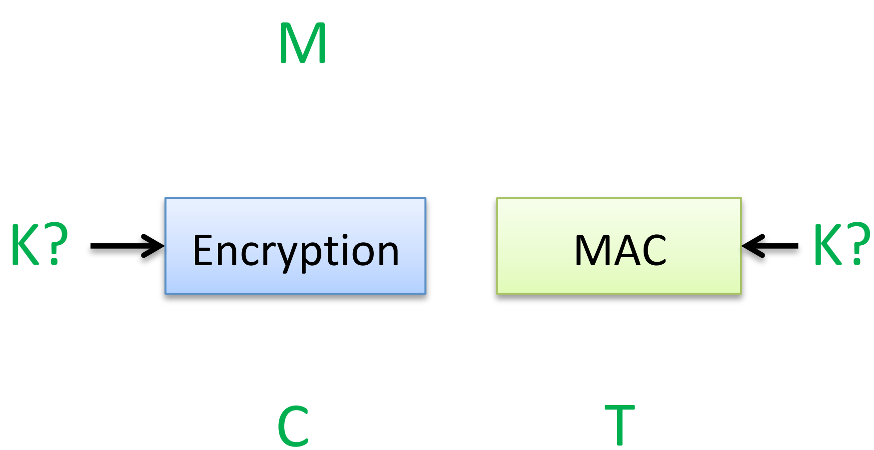


To verify a M,T pair, check if HMAC(K,M) = T

Unforgeability holds if H is a secure PRF when so-keyed

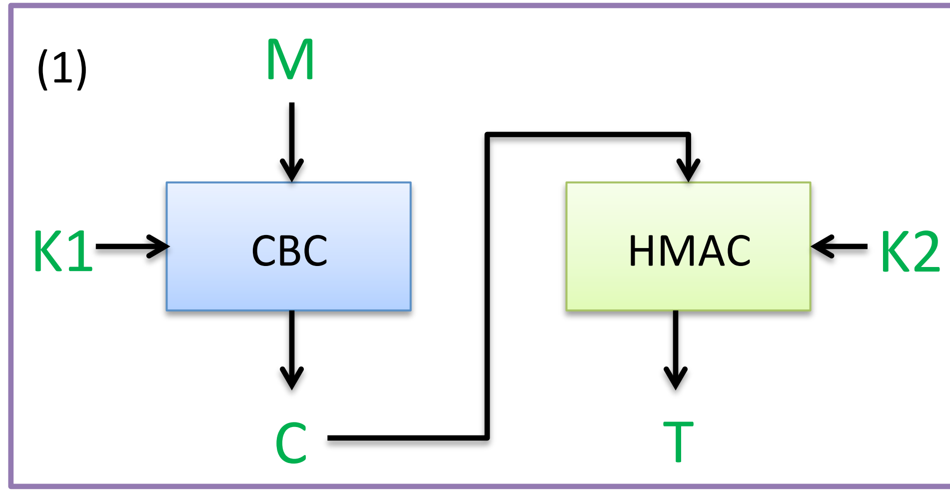
Build ***authenticated encryption***…?

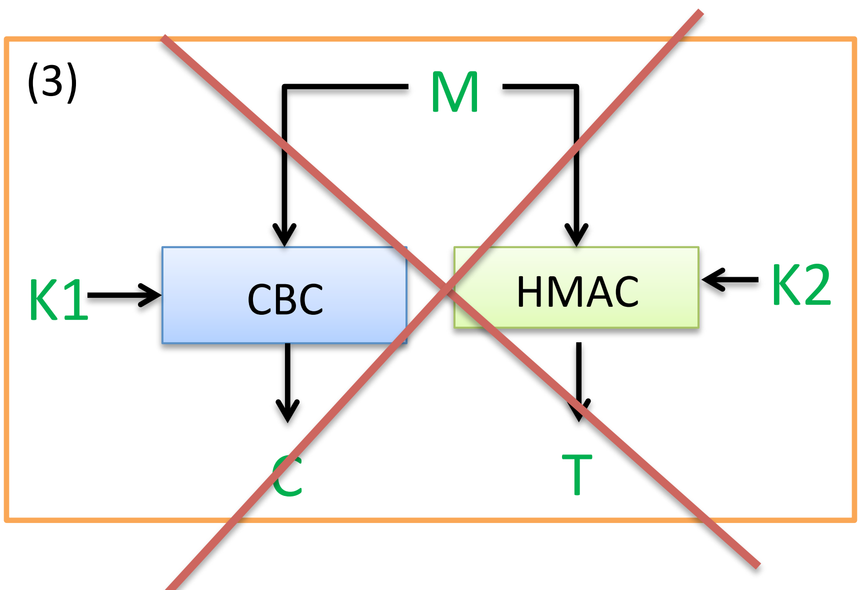
- Recall that our goal is a single "thing" giving both secrecy+authenticity
- Want to combine some encryption scheme with a MAC – how do we do this? Any ideas?

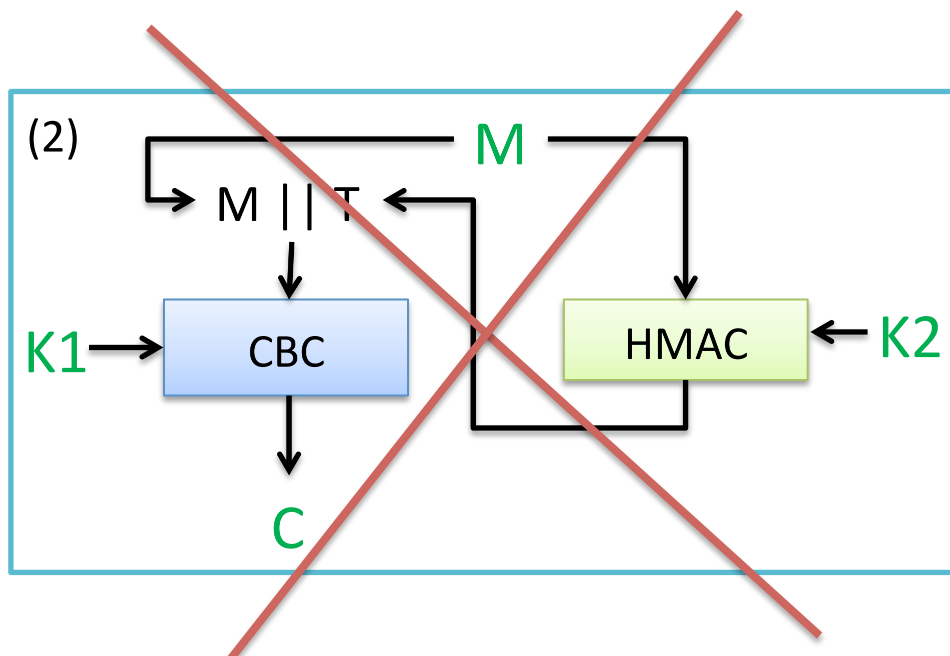


Build a new scheme from CBC and HMAC
Kg outputs CBC key K1 and HMAC key K2

Several ways to combine:
(1) encrypt-then-mac
(2) mac-then-encrypt
(3) encrypt-and-mac

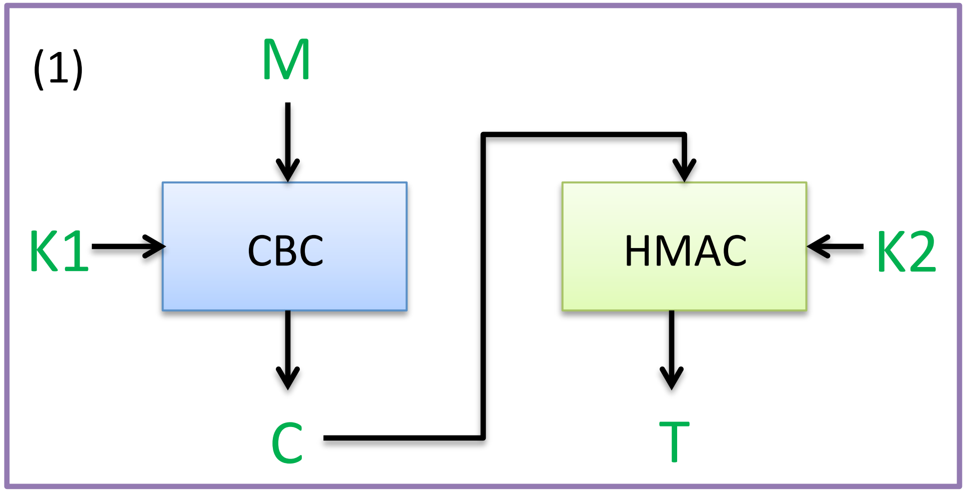






Build a new scheme from CBC and HMAC
Kg outputs CBC key K1 and HMAC key K2

Several ways to combine:
(1) encrypt-then-mac
(2) mac-then-encrypt
(3) encrypt-and-mac



Thm. If encryption scheme provides confidentiality against passive attackers and MAC provides unforgeability, then Encrypt-then-MAC provides secure authenticated encryption

Are we done?

It's circa 2002 in crypto research,
and we're in decent shape…

Still no!

Why? Many reasons:

efficiency: latency, code size…

better security: randomness reuse…

Fundamental reason why we're not done:

crypto is *hard to get right*, people make mistakes often.

Need to build crypto that is hard to misuse.

Build dedicated AE schemes!

Dedicated authenticated encryption schemes

Attack	Inventors	Notes
OCB (Offset Codebook)	Rogaway	One-pass
GCM (Galois Counter Mode)	McGrew, Viega	CTR mode plus Carter-Wegman MAC
ChaCha20/ Poly1305	Bernstein	“essentially” CTR mode plus special Carter-Wegman MAC
CCM	Housley, Ferguson, Whiting	CTR mode plus CBC-MAC
EAX	Wagner, Bellare, Rogaway	CTR mode plus OMAC

Dedicated authenticated encryption schemes

Attack	Inventors	Notes
OCB (Offset Codebook)	Rogaway	One-pass
GCM (Galois Counter Mode)	McGrew, Viega	CTR mode plus Carter-Wegman MAC
ChaCha20/ Poly1305		plus special
CCM	Housley, Ferguson, Whiting	CTR mode plus CBC-MAC
EAX	Wagner, Bellare, Rogaway	CTR mode plus OMAC

OCB was second AE scheme published.
By most accounts, still the best.
Nobody uses it because of patents!

Today's lecture

- Block cipher modes of operation
  - ECB, CTR, CBC
- Attacking insecure approaches
  - malleability, padding oracle
- Message authentication
  - HMAC
- Authenticated encryption
  - OCB, GCM