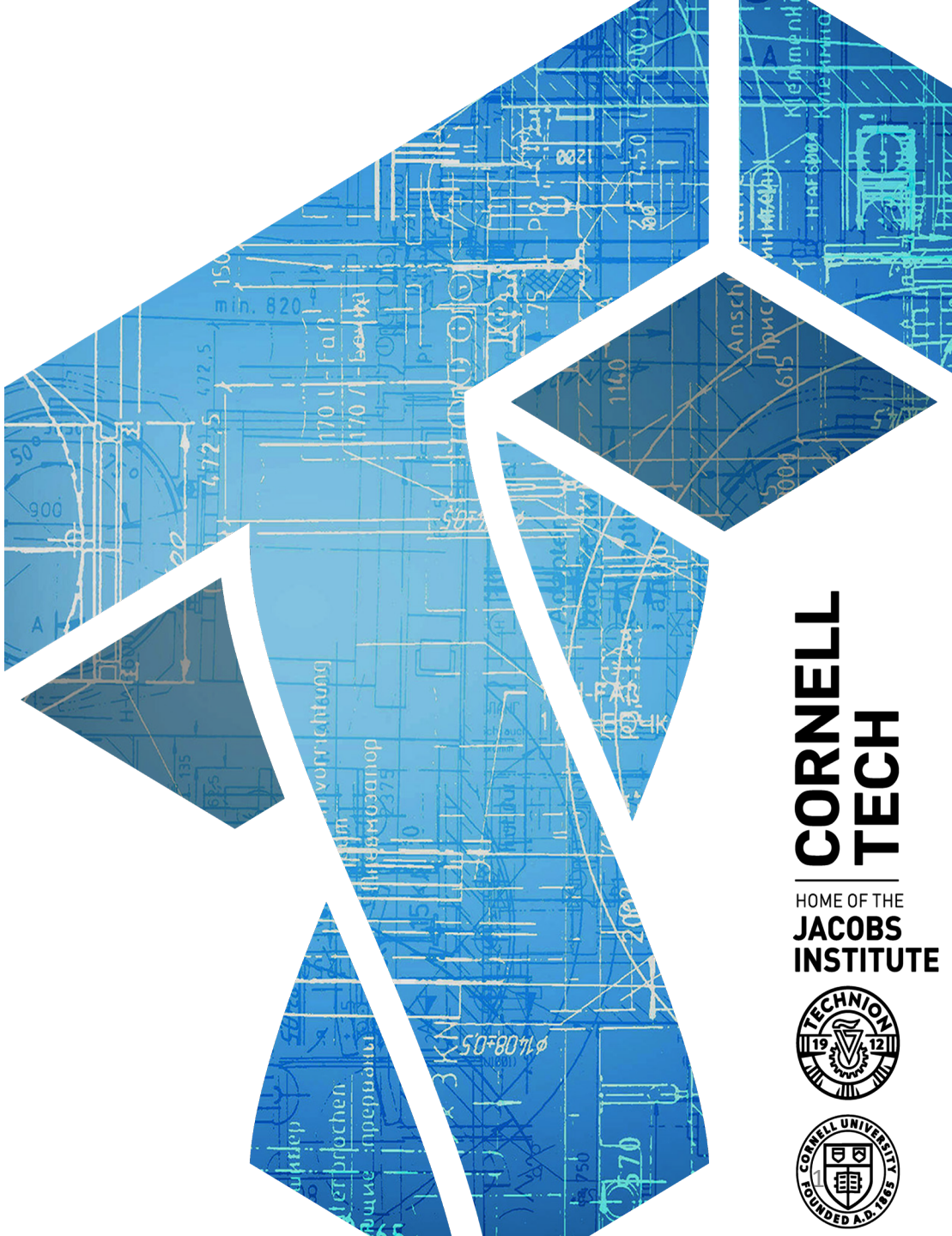
CS 5435: Cryptography part 3

Instructor: Tom Ristenpart

https://github.com/tomrist/cs5435-fall2019



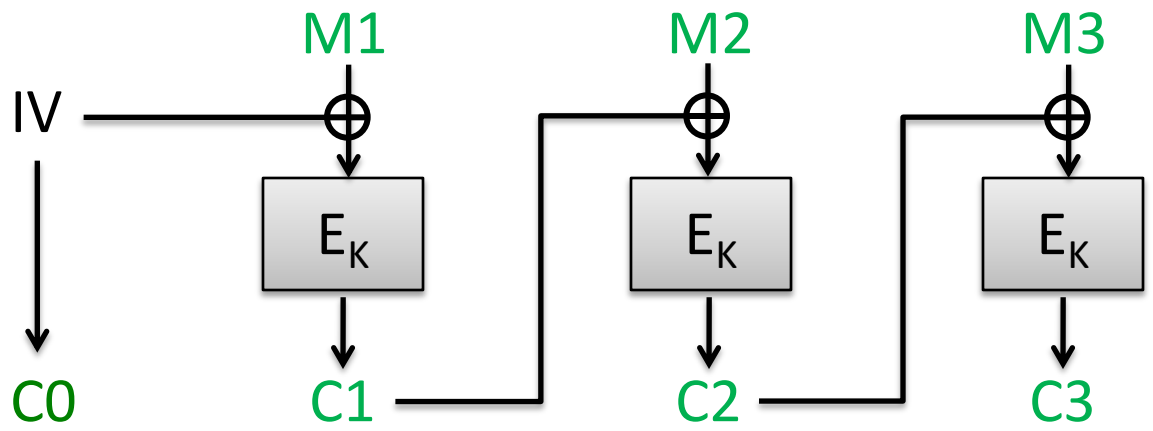
CORNELL TECH
HOME OF THE JACOBS INSTITUTE

Today’s lecture

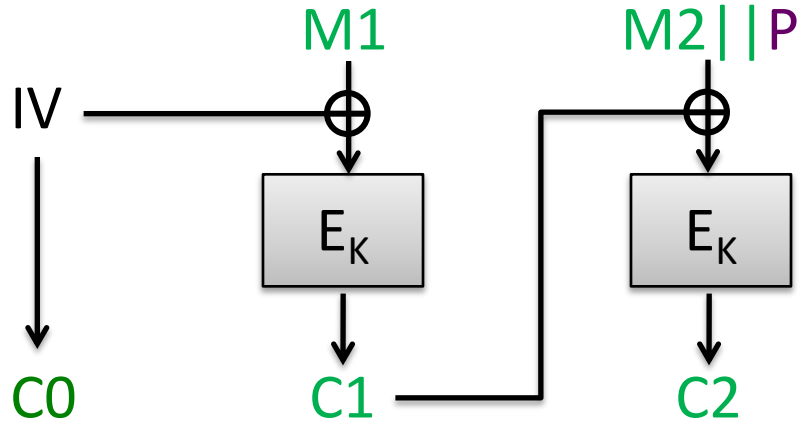
- Wrapping up CBC padding oracle attacks
- Message authentication
- Authenticated encryption

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:



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

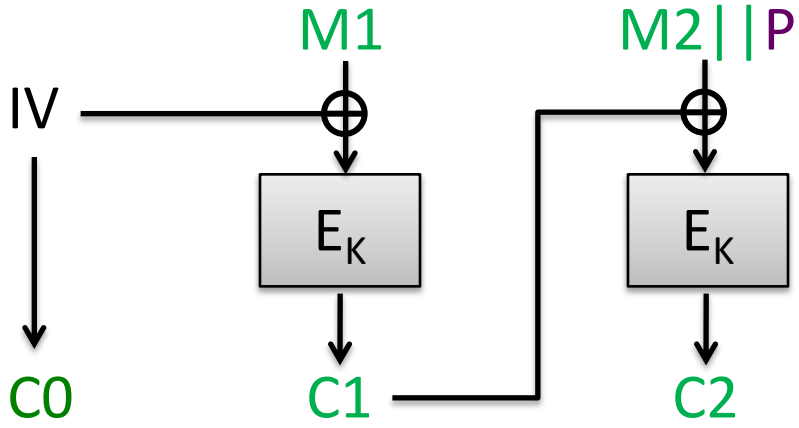
ok

C0, C1⊕1 , C2

error

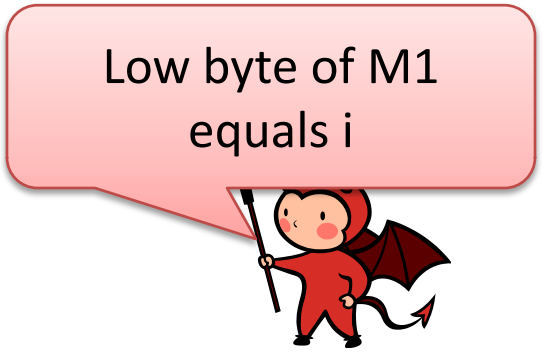
Dec(K, C')
M1'||M2'||P' = CBC-Dec(K,C')
If P' ≠ 0x00 then
Return error
Else
Return ok

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

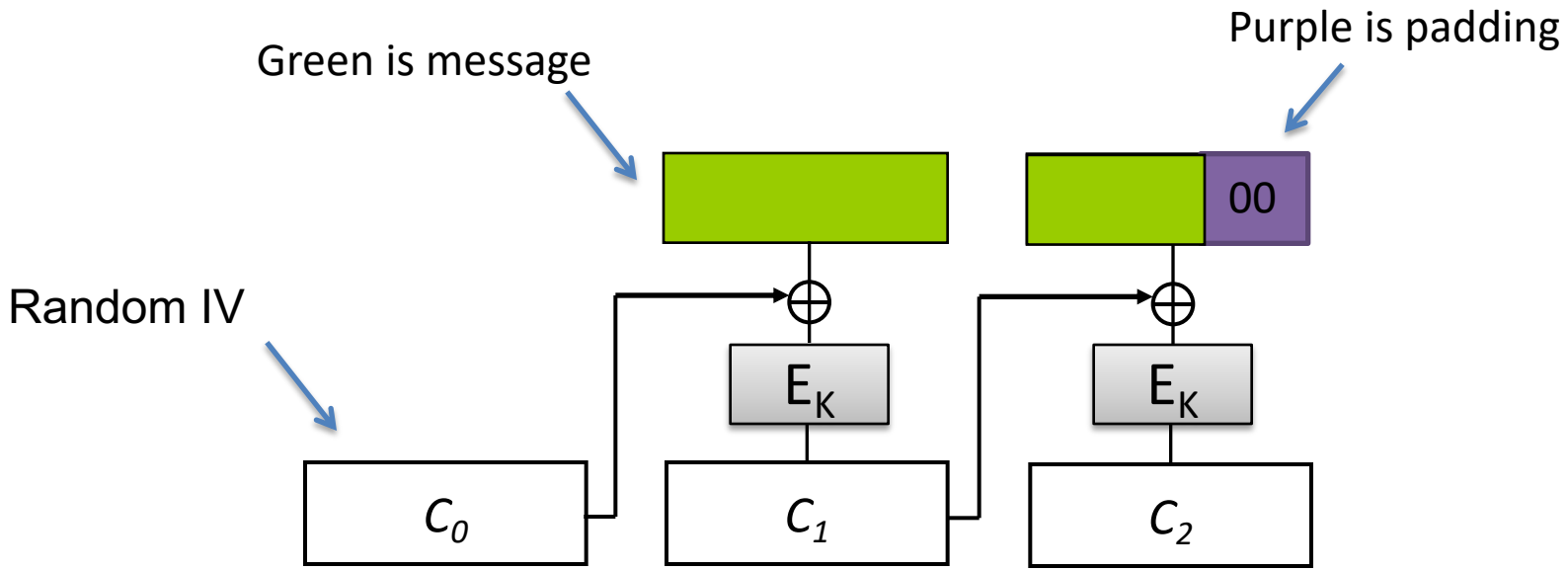
…

R , C0 ⊕ i , C1
ok

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

Dec(K, C')
M1'||M2'||P' = CBC-Dec(K,C')
If P' ≠ 0x00 then
    Return error
Else
    Return ok

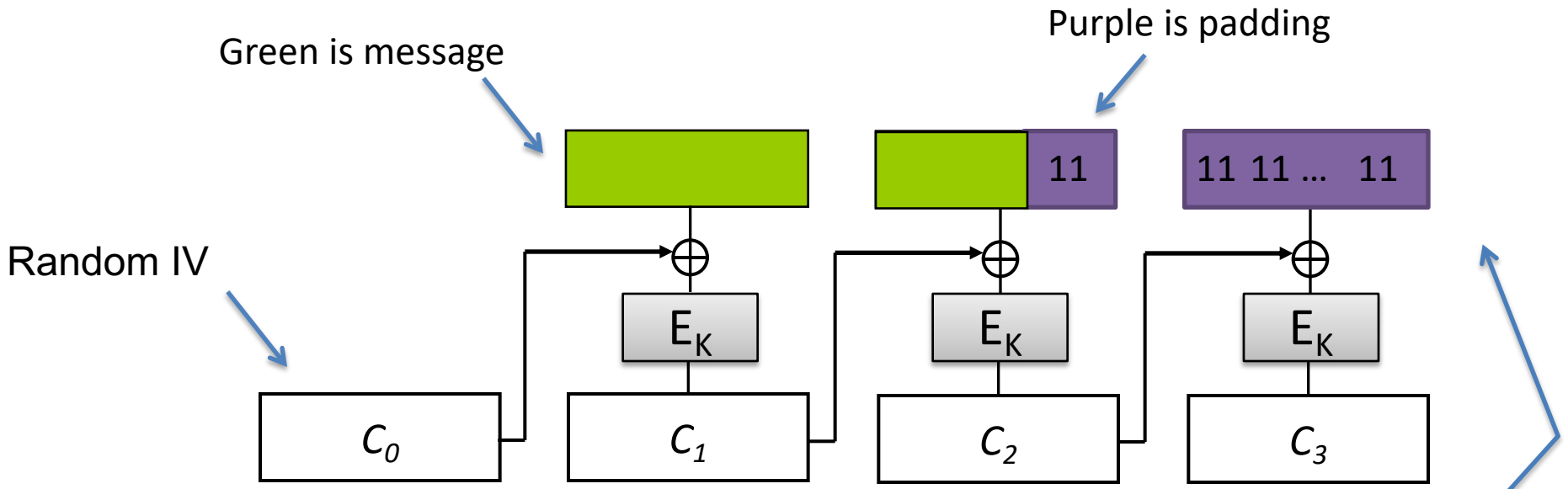
Padding for CBC Mode in TLS



Possible paddings in TLS:

00 01 01 02 02 02 etc.

Padding for CBC Mode in TLS

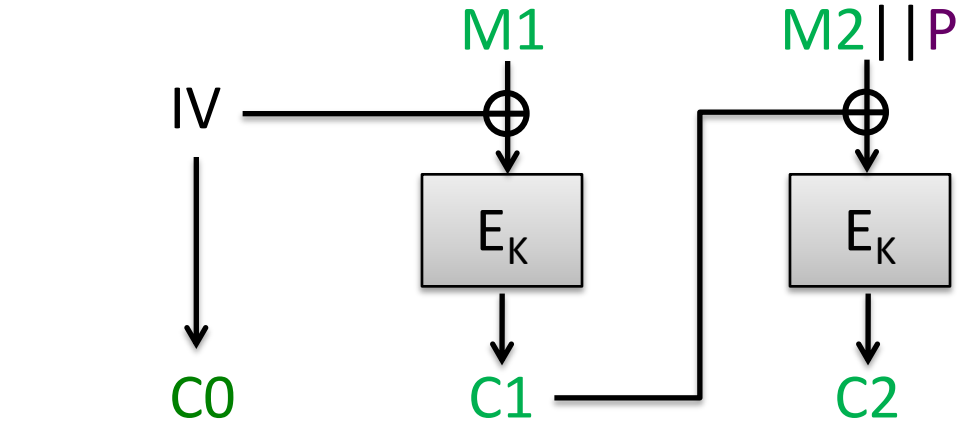


Possible paddings in TLS:

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 Called "traffic analysis attacks"

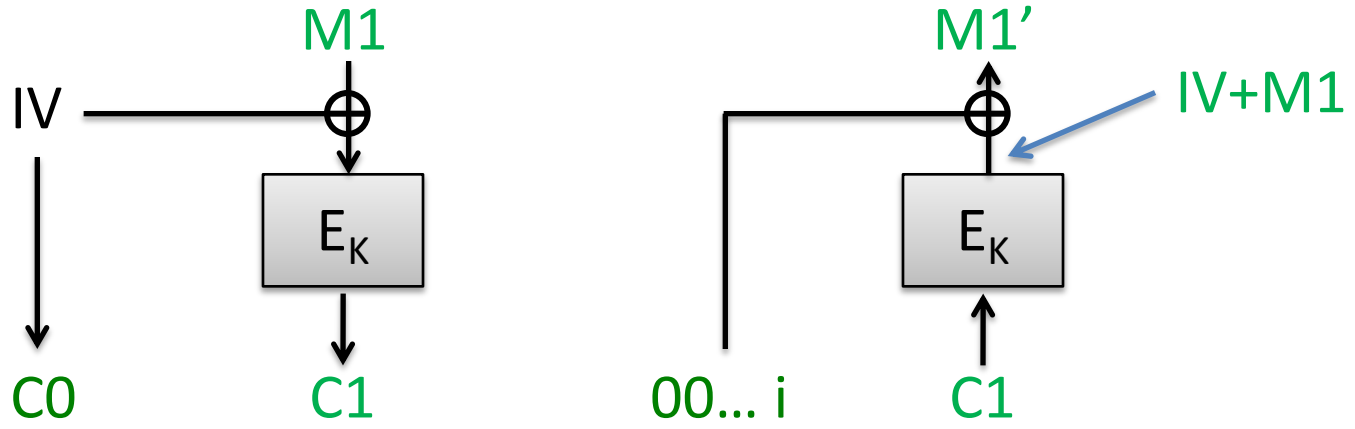
Vaudenay's padding oracle attack



Goal:
Decrypt entire plaintext

Dec(K, C')
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 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

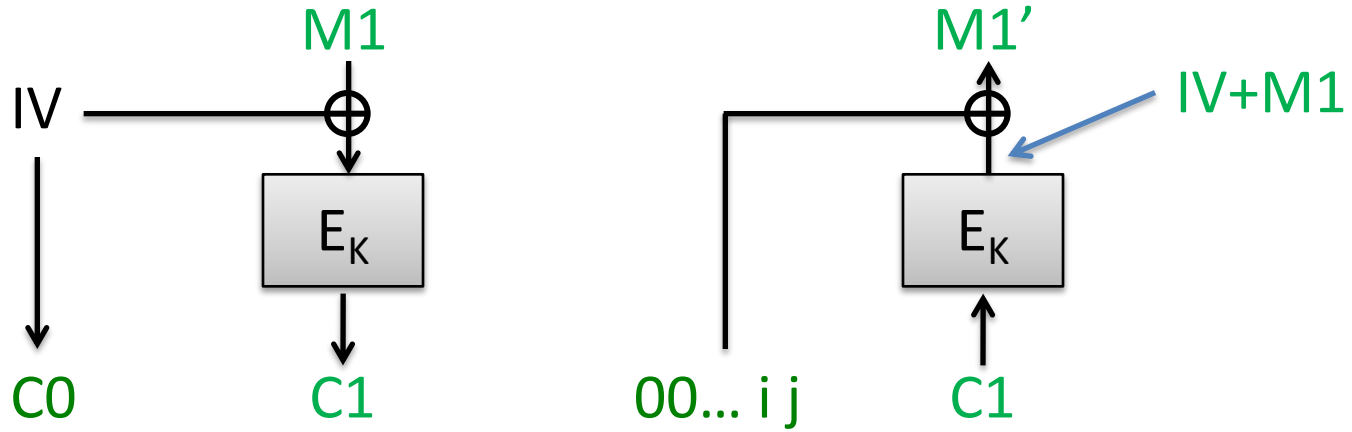
00…02 , C1
error

…

00… i, C1
ok

Dec(K, C')
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

Dec(K, C')
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

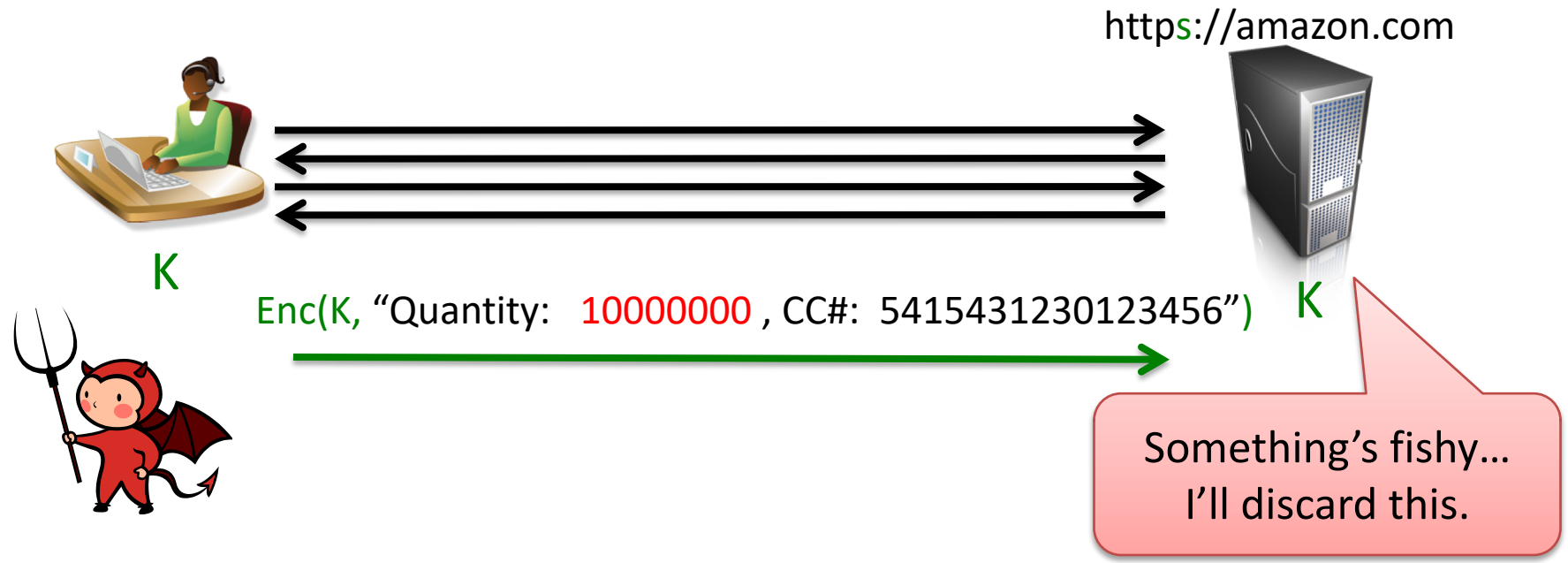
- Take five minutes and discuss with your neighbor…
  - Are padding oracles possible against CTR mode?
  - Are padding oracles possible against ECB mode?
  - How would you prevent padding oracle attacks? Cryptographic countermeasure or “system-level”?

We need new primitives

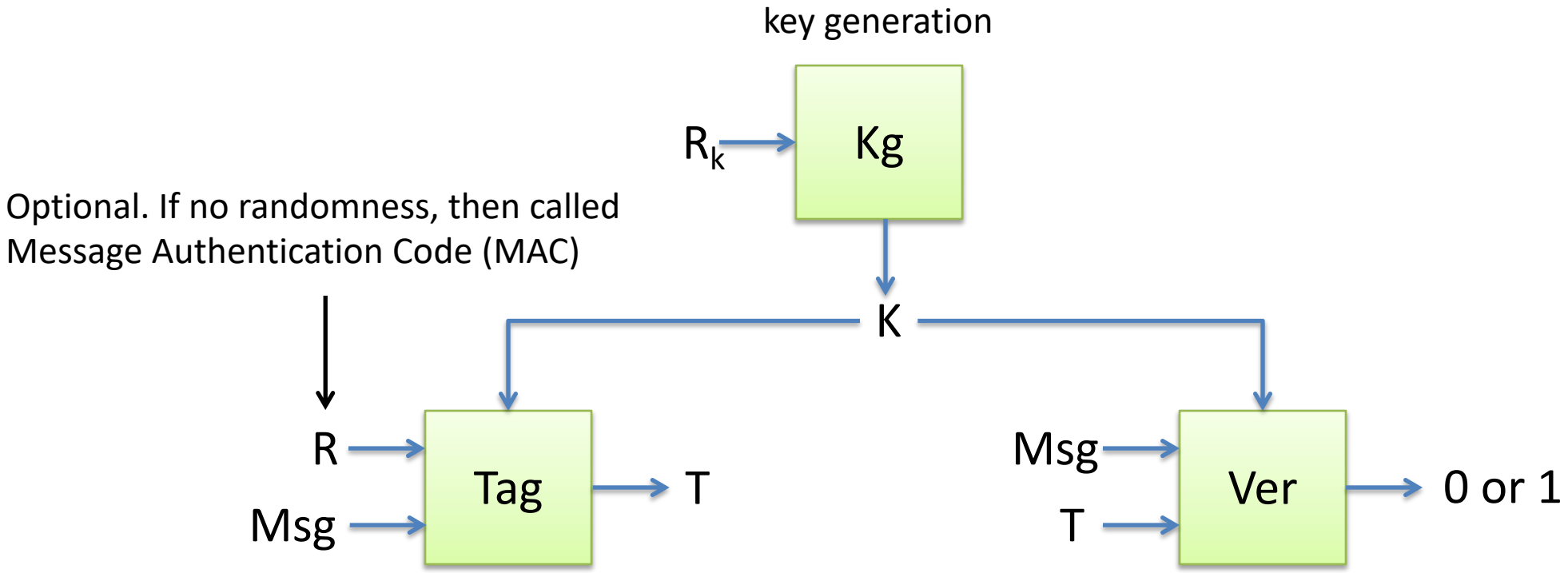
Message authentication makes malicious modifications detectable

Goal is ***authenticated encryption*** (AE): Hide message and detect modifications

Can build by combining encryption with message authentication scheme

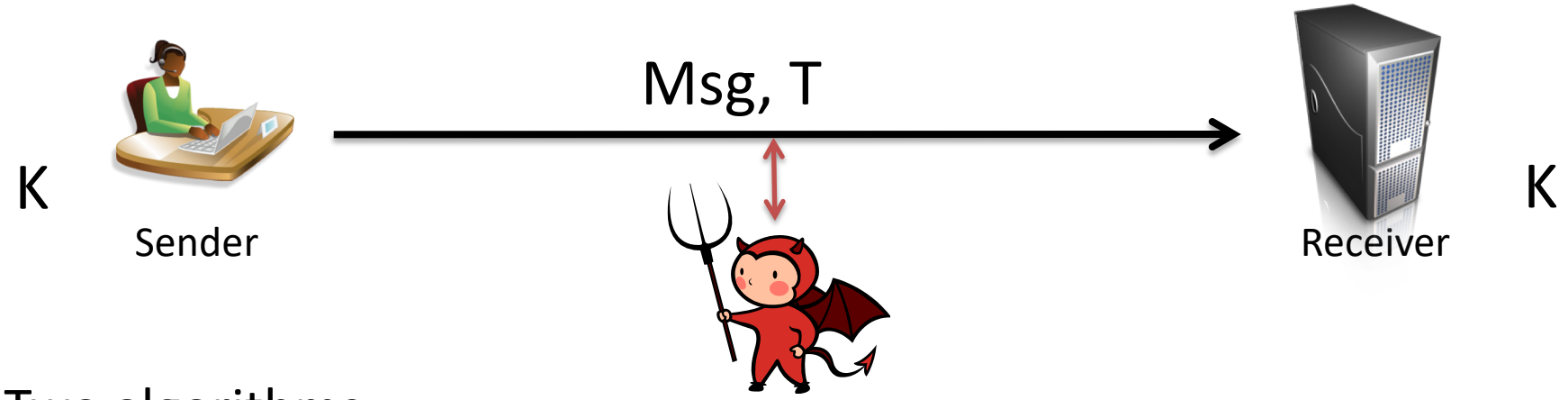


Message authentication



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

Message authentication



Two algorithms:

(1) Tag(K,Msg) outputs a tag T

(2) Verify(K,Msg,T) outputs 0/1 (invalid / valid)

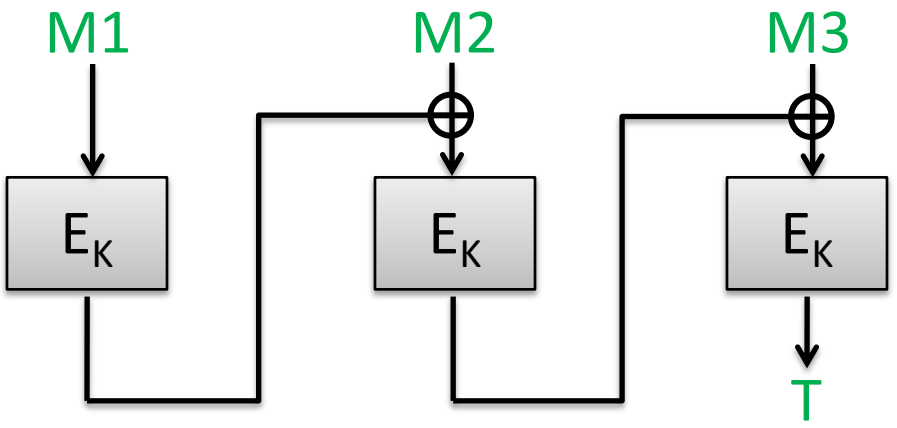
Security: No computationally efficient attacker can forge tags for a new message even when attacker gets

$$(Msg_1 , T_1) , (Msg_2 , T_2), \dots , (Msg_q , T_q)$$

for messages of his choosing and reasonably large q.

CBC-MAC

Split Msg into blocks M1, M2, M3

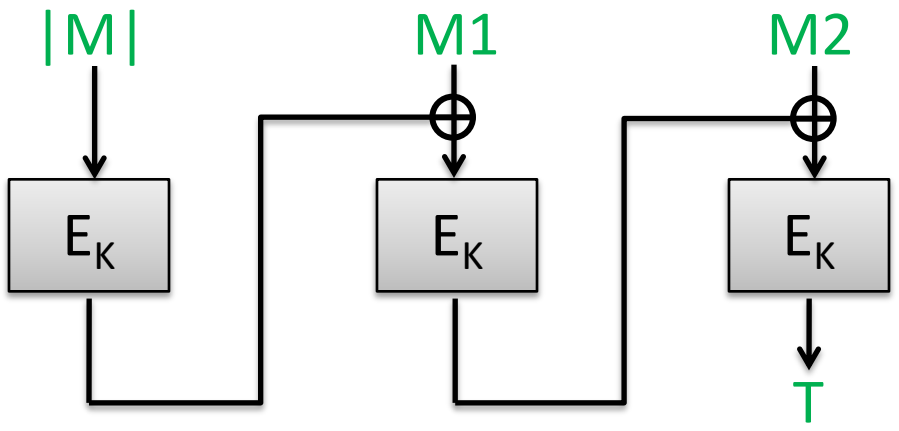


Turns out this is secure MAC

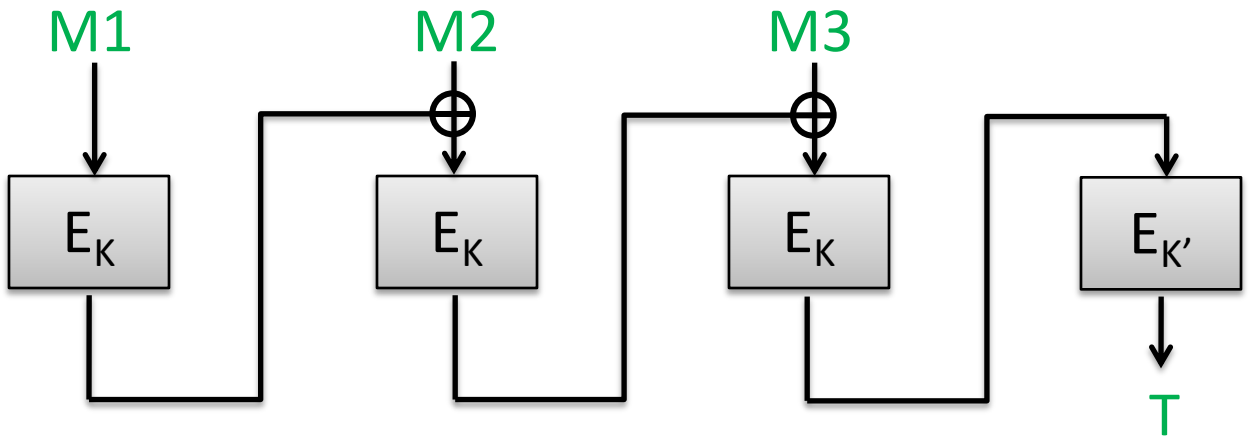
if K used only on same-length messages

Variable-message-length CBC-MAC

- Prepend message length

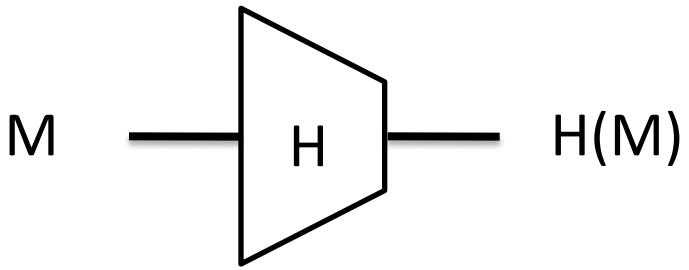


- Encrypted CBC-MAC



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

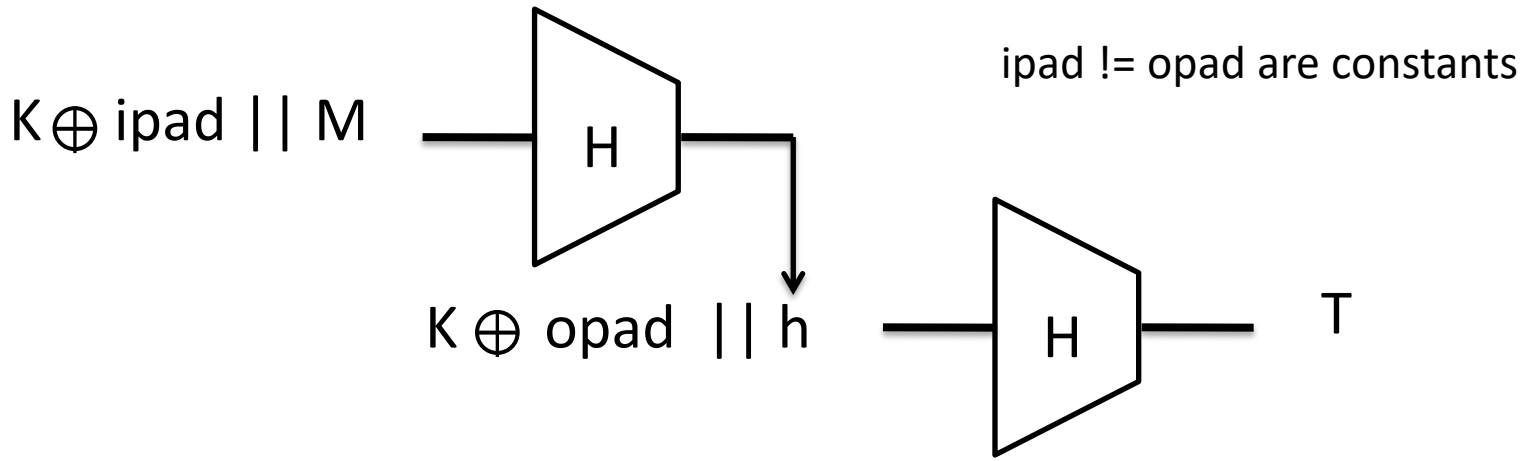
Some security goals:

- collision resistance: can't find M != M' such that H(M) = H(M')
- "behaves like" random function: H(M) is uniformly distributed bit string for all M
  - Sometimes called the random oracle model (ROM)

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:

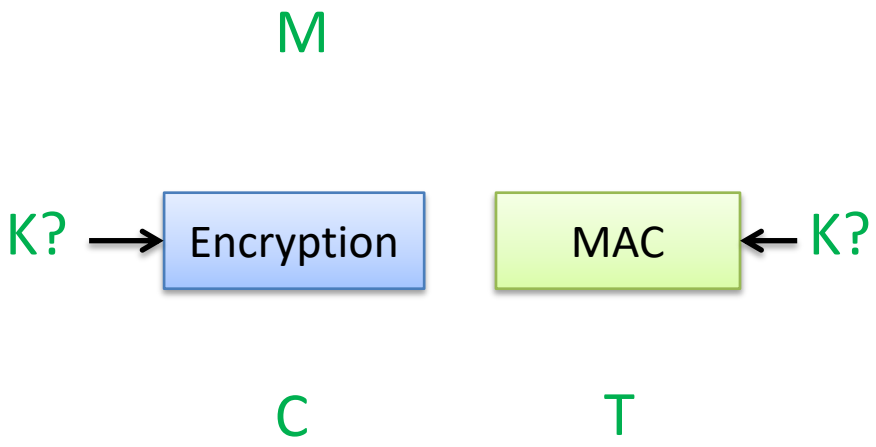


To verify a M,T pair, check if HMAC(K,M) = T

Unforgeability holds if H behaves like a random function

Build *authenticated encryption…?*

- Recall that our goal is authenticated encryption
- Want to combine some encryption scheme with a MAC – how do we do this? Any ideas?



Build a new scheme from CBC and HMAC

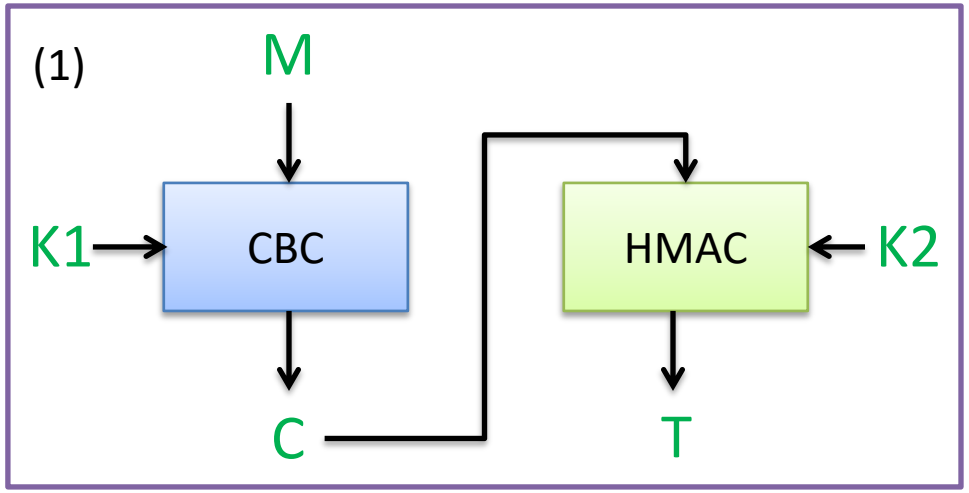
Kg outputs CBC key K1 and HMAC key K2

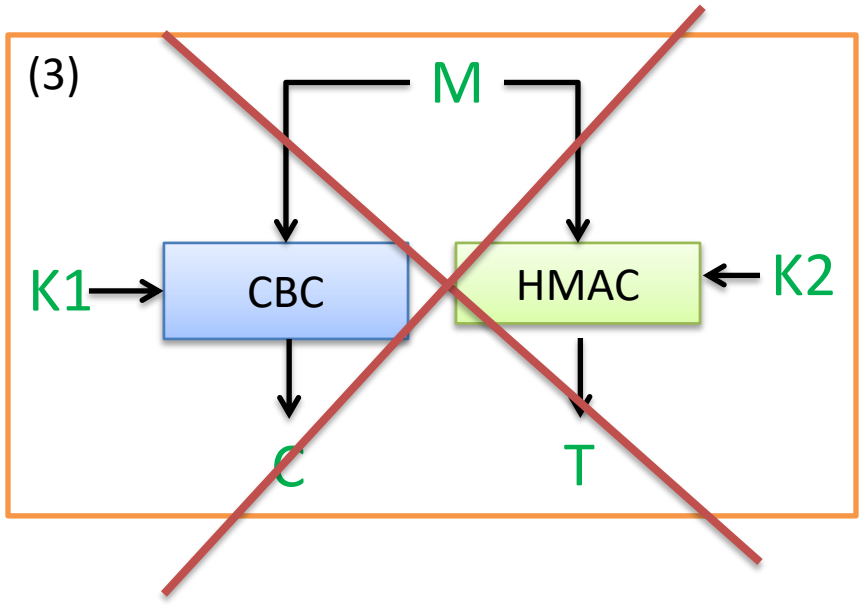
Several ways to combine:

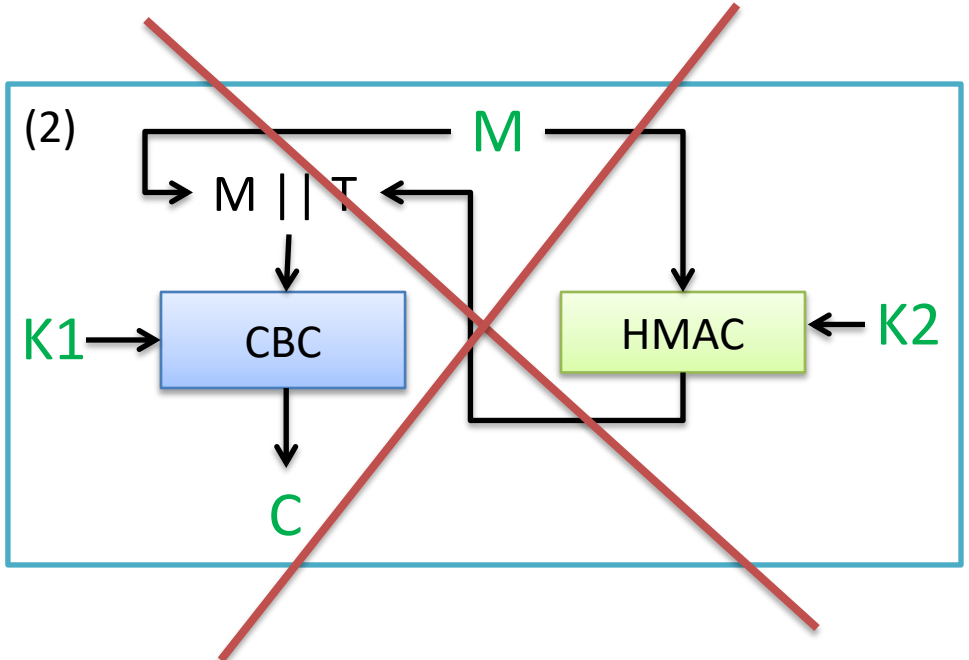
(1) encrypt-then-mac

(2) mac-then-encrypt

(3) encrypt-and-mac

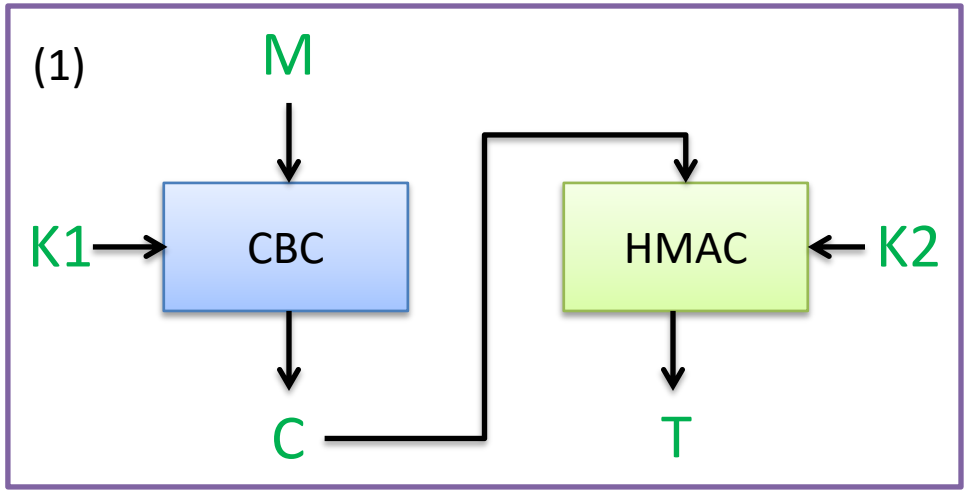






Build a new scheme from CBC and HMAC
Kg outputs CBC key K1 and HMAC key K2

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

Dedicated authenticated encryption schemes

Attack	Inventor(s)	Notes
OCB (Offset Codebook)	Rogaway	One-pass mode and fastest
AES-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

Other considerations in AE:

robustness & IV misuse, deterministic AE, associated data, …

Next time: asymmetric cryptography

So far we've seen how to encrypt using shared key

We'll introduce asymmetric cryptography next lecture