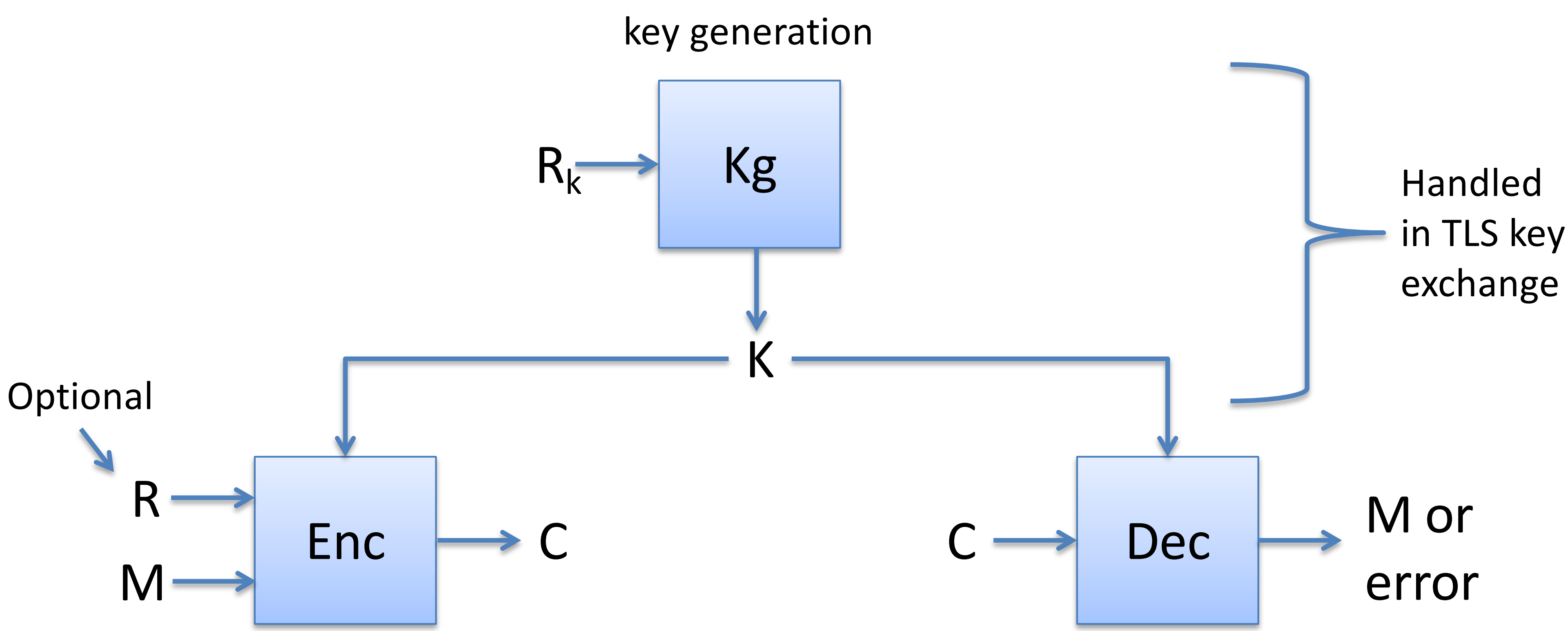
Symmetric encryption

CS642: Computer Security

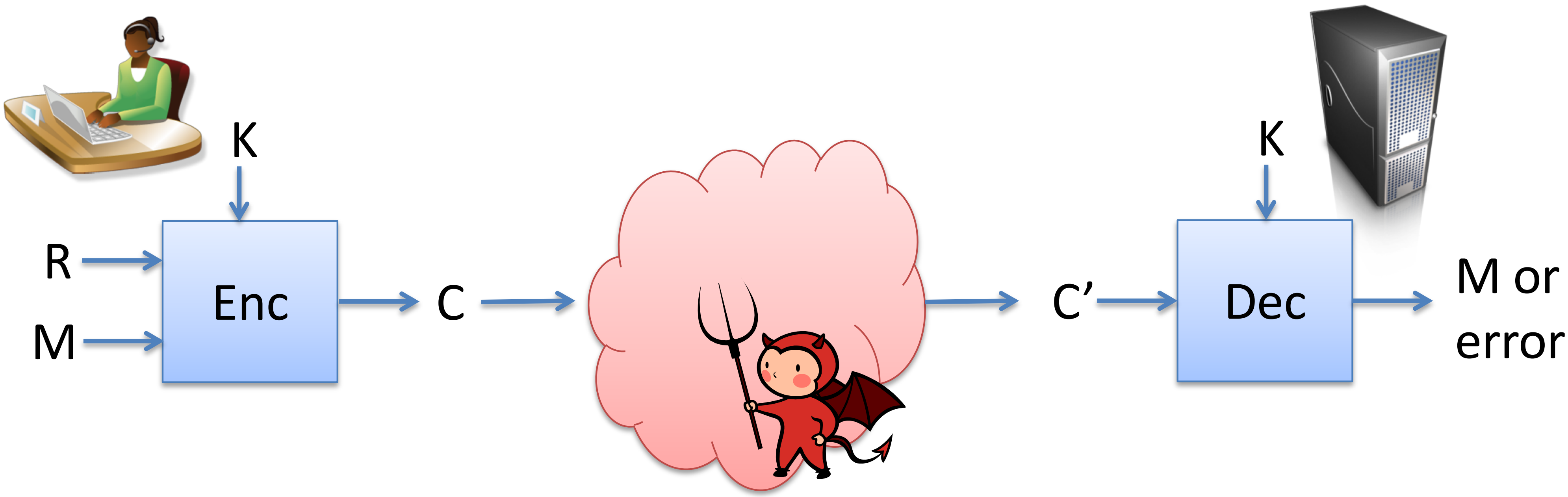
Symmetric encryption



C is a ciphertext

Correctness: D(K , E(K,M,R)) = M with probability 1 over randomness used

In TLS symmetric encryption underlies the Record Layer



What security properties do we need from symmetric encryption?

1) Confidentiality: should not learn any information about M
2) Authenticity: should not be able to forge messages

Often referred to as Authenticated Encryption security

Provable security cryptography

Supplement "design-break-redesign-break…" with a more mathematical approach

1. Design a cryptographic scheme
2. Provide proof that no one is able to break it

} Shannon 1949

<u>Formal definitions</u>

Scheme semantics

Security

<u>Security proofs</u>

Show it is mathematically impossible to break security

One-time pads

Fix some message length L

Kg: output random bit string K of length L

$E(K,M) = M \oplus K$ $\qquad$ $D(K,C) = C \oplus K$

Shannon's security notion

Def. A symmetric encryption scheme is perfectly secure if for all messages M,M' and ciphertexts C

$$\Pr[\, E(K,M) = C \,] = \Pr[\, E(K,M') = C \,]$$

where probabilities are over choice of K

In words:
each message is equally likely to map to a given ciphertext

In other words:
seeing a ciphertext leaks nothing about what message was encrypted

Does a substitution cipher meet this definition? No!

Shannon's security notion

Def. A symmetric encryption scheme is perfectly secure if for all messages M,M' and ciphertexts C

$$\Pr[\, E(K,M) = C \,] \;=\; \Pr[\, E(K,M') = C \,]$$

where probabilities are over choice of K

Thm. OTP is perfectly secure

For any C and M of length L bits

$$\Pr[\, K \oplus M = C \,] \;=\; 1 / 2^L$$

$$\Pr[\, K \oplus M = C \,] \;=\; \Pr[\, K \oplus M' = C \,]$$

Shannon's security notion

Def. A symmetric encryption scheme is perfectly secure if for all messages M,M' and ciphertexts C

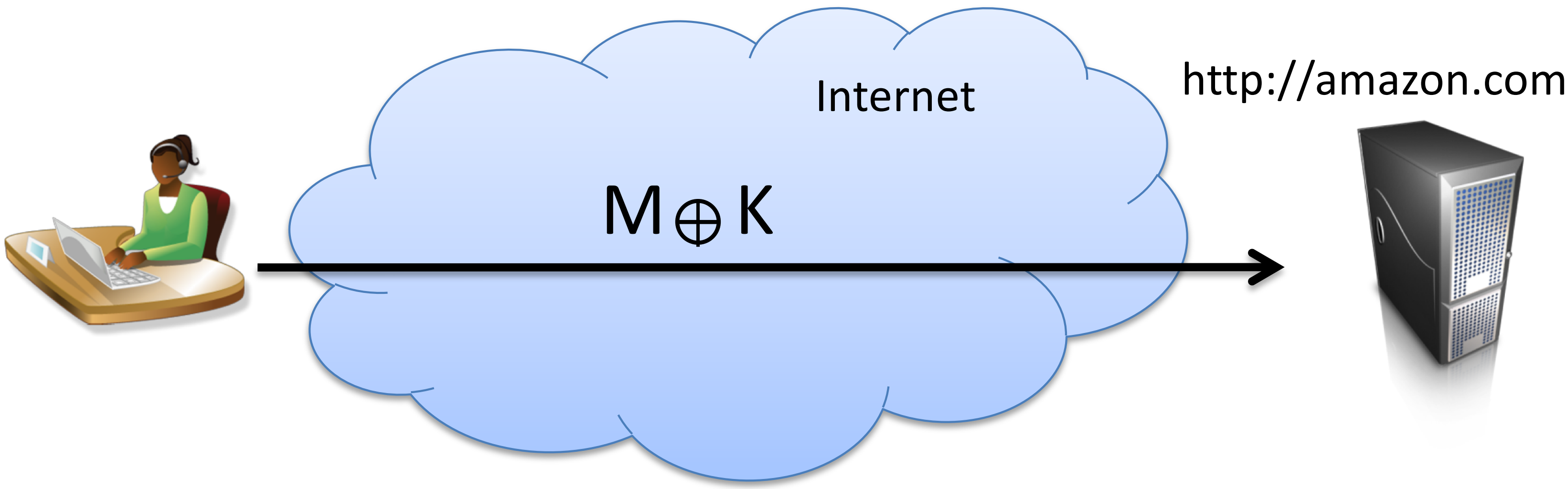
$$\Pr[\, E(K,M) = C \,] \;=\; \Pr[\, E(K,M') = C \,]$$

where probabilities are over choice of K

Thm. OTP is perfectly secure

Thm. Any perfectly secure scheme requires as many key bits as message bits.

Back to our application



Does OTP provide a secure channel?

Integrity easily violated

Reuse of K for messages M,M' leaks $M \oplus M'$

Encrypting same message twice under K leaks the message equality

K must be as large as message

Message length revealed

Cryptography as computational science

Use computational intractability as basis for confidence in systems

1. Design a cryptographic scheme
2. Provide proof that no attacker with limited computational resources can break it

} Goldwasser, Micali and Blum circa 1980's

Formal definitions

Scheme semantics

Security

Security proofs (reductions)

Breaking scheme

↓

Breaking assumptions

Example:

Attacker can not recover credit card

⇕

Can not factor large composite numbers

But no one knows how to do this. It's been studied for a very long time!

As long as assumptions holds we believe in security of scheme!

Provable security yields

1) well-defined assumptions and security goals

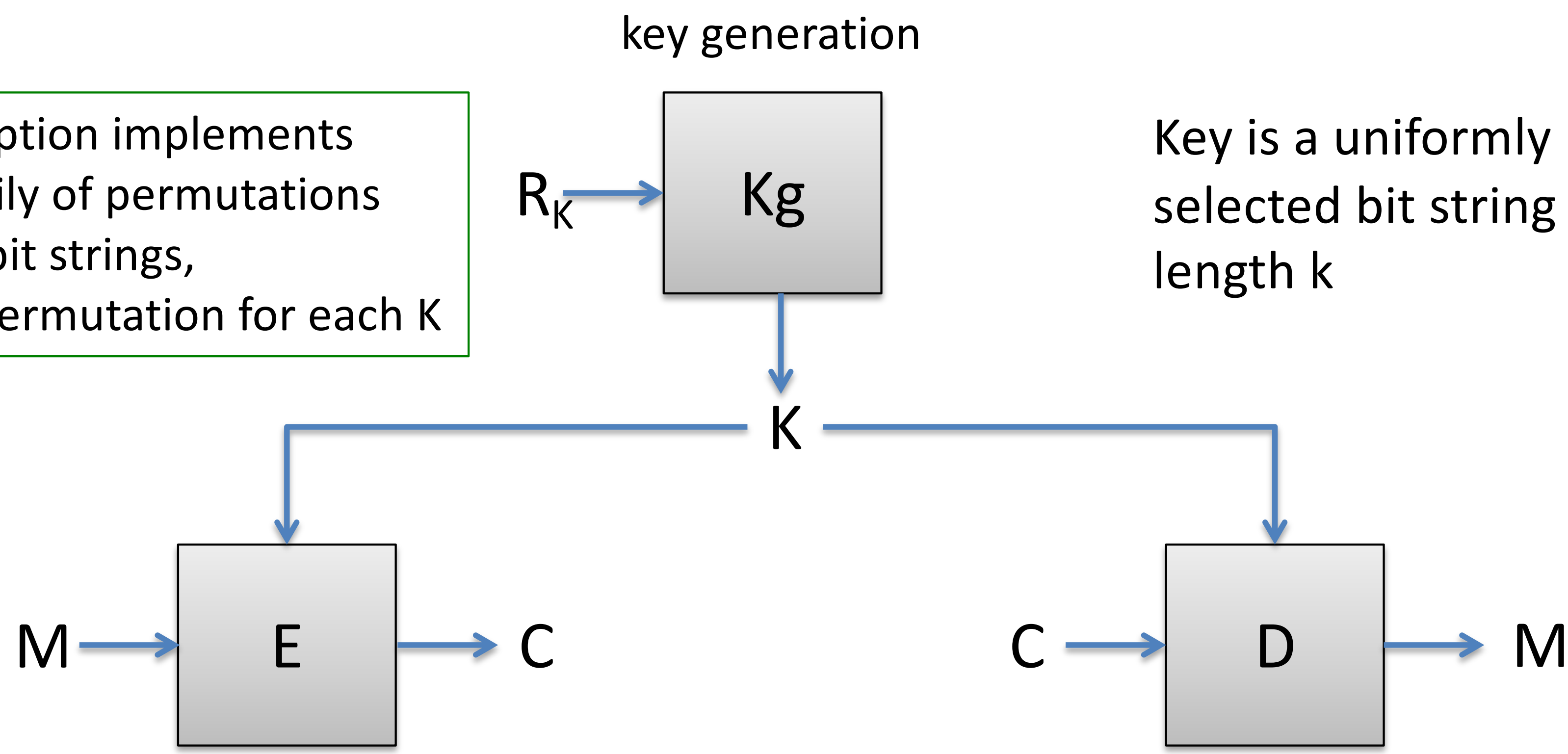
2) cryptanalysts can focus on assumptions and models

Typical assumptions

- Basic atomic primitives are hard to break:
  - Factoring of large composites intractable
  - RSA permutation hard-to-invert
  - Block ciphers (AES, DES) are good pseudorandom permutations (PRPs)
  - Hash functions are collision resistant

Confidence in atomic primitives is gained by cryptanalysis, public design competitions

Block ciphers



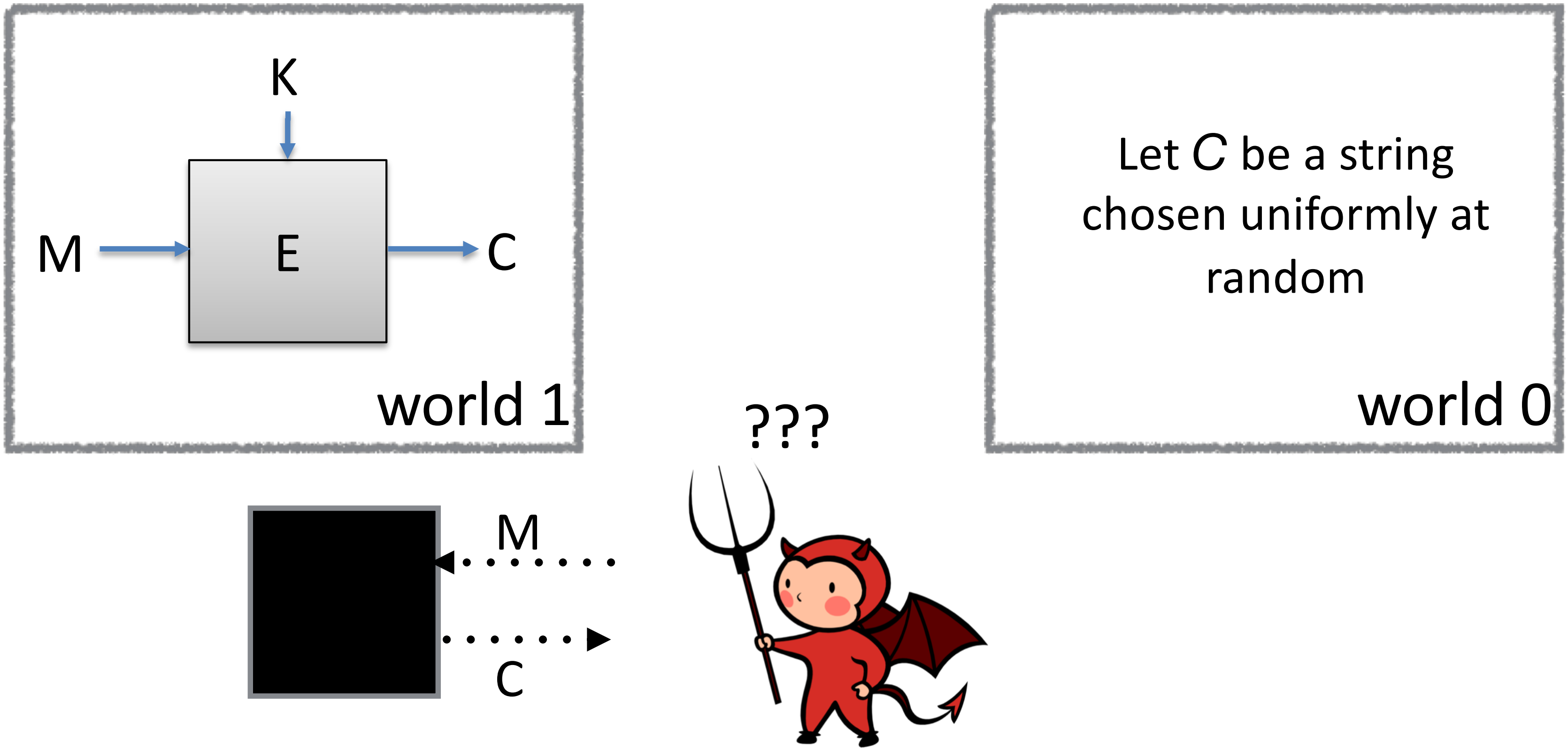
Encryption implements a family of permutations on n bit strings, one permutation for each K

Key is a uniformly selected bit string of length k

$E: \{0,1\}^k \times \{0,1\}^n \text{ -> } \{0,1\}^n$

Security goal: $E(K,M)$ is indistinguishable from a random n-bit string for anyone that doesn't know K

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



Can adversary distinguish between World 0 and World 1?

If this holds for all polynomial time adversaries, then *E* is called a secure pseudorandom function (PRF)

block cipher security

Data encryption standard (DES)

Originally called Lucifer

- team at IBM
- input from NSA
- standardized by NIST in 1976

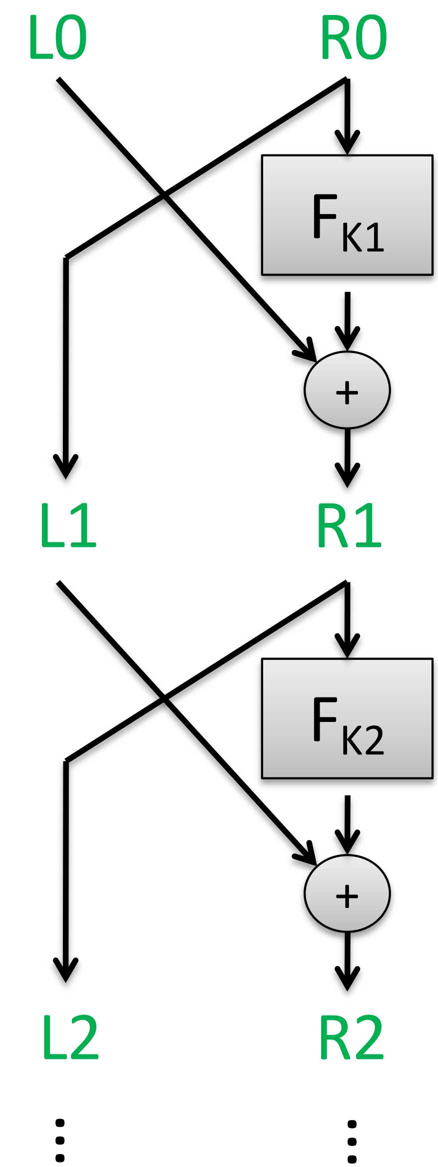
$n = 64$
$k = 56$

Number of keys:
72,057,594,037,927,936

Split 64-bit input into L0,R0 of 32 bits each

Repeat Feistel round 16 times

Each round applies function F using separate round key



Best attacks against DES

Attack	Attack type	Complexity	Year
Biham, Shamir	Chosen plaintexts, recovers key	2^{47} plaintext, ciphertext pairs	1992
DESCHALL	Unknown plaintext, recovers key	$2^{56/4}$ DES computations 41 days	1997
EFF Deepcrack	Unknown plaintext, recovers key	~4.5 days	1998
Deepcrack + DESCHALL	Unknown plaintext, recovers key	22 hours	1999

- DES is still used in some places
- 3DES (use DES 3 times in a row with more keys) expands keyspace and still used widely in practice

Advanced Encryption Standard (AES)

Response to 1999 attacks:

- NIST has design competition for new block cipher standard
- 5 year design competition
- 15 designs, Rijndael design chosen

Advanced Encryption Standard (AES)

Rijndael (Rijmen and Daemen)

n = 128
k = 128, 192, 256

Number of keys for k=128:
340,282,366,920,938,463,463,374,607,431,768,211,456

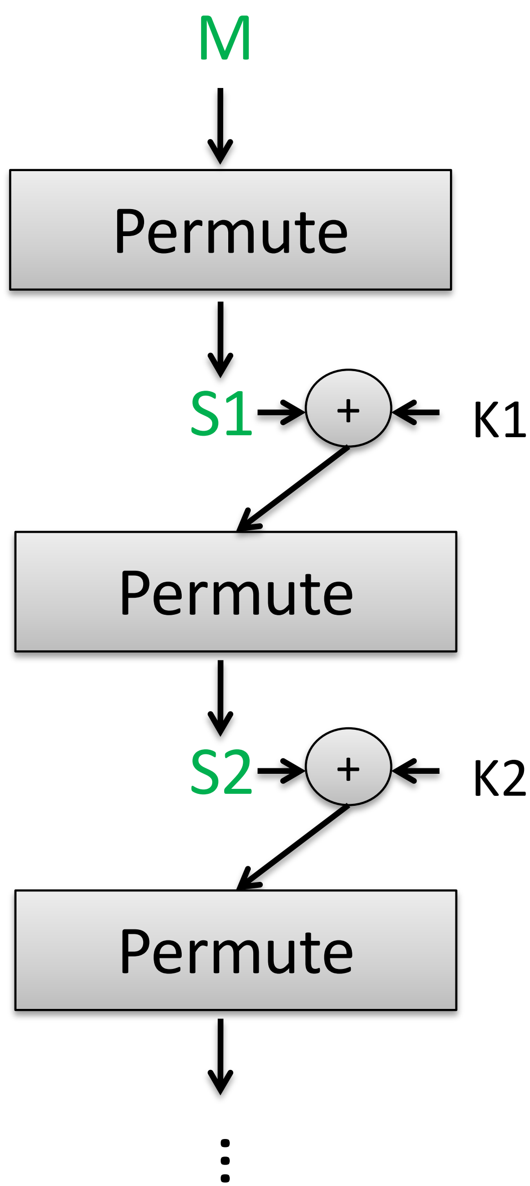
Substitution-permutation design.
For k=128 uses 10 rounds of:

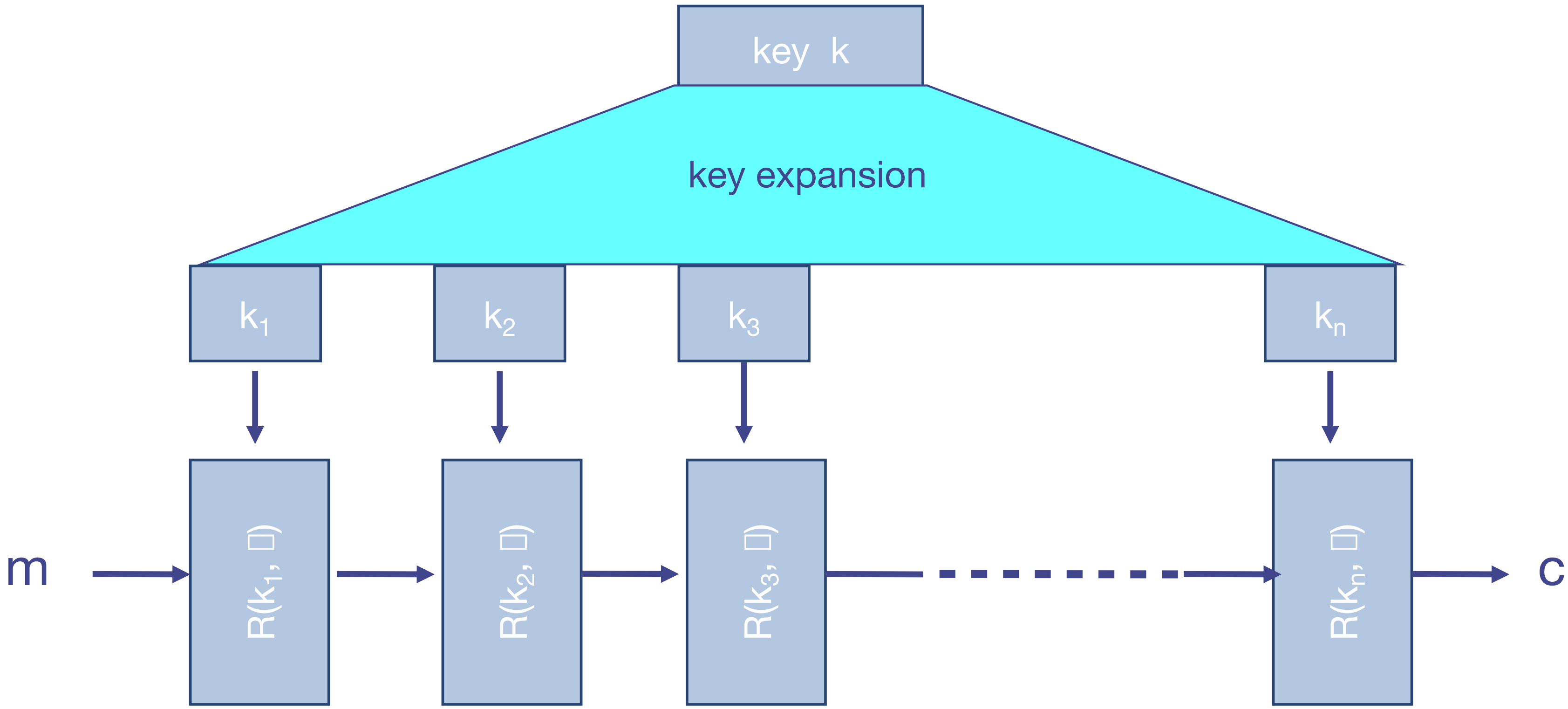
1) Permute:

SubBytes (non-linear S-boxes)
ShiftRows + MixCols (invertible linear transform)

2) XOR in a round key derived from K

(Actually last round skips MixCols)

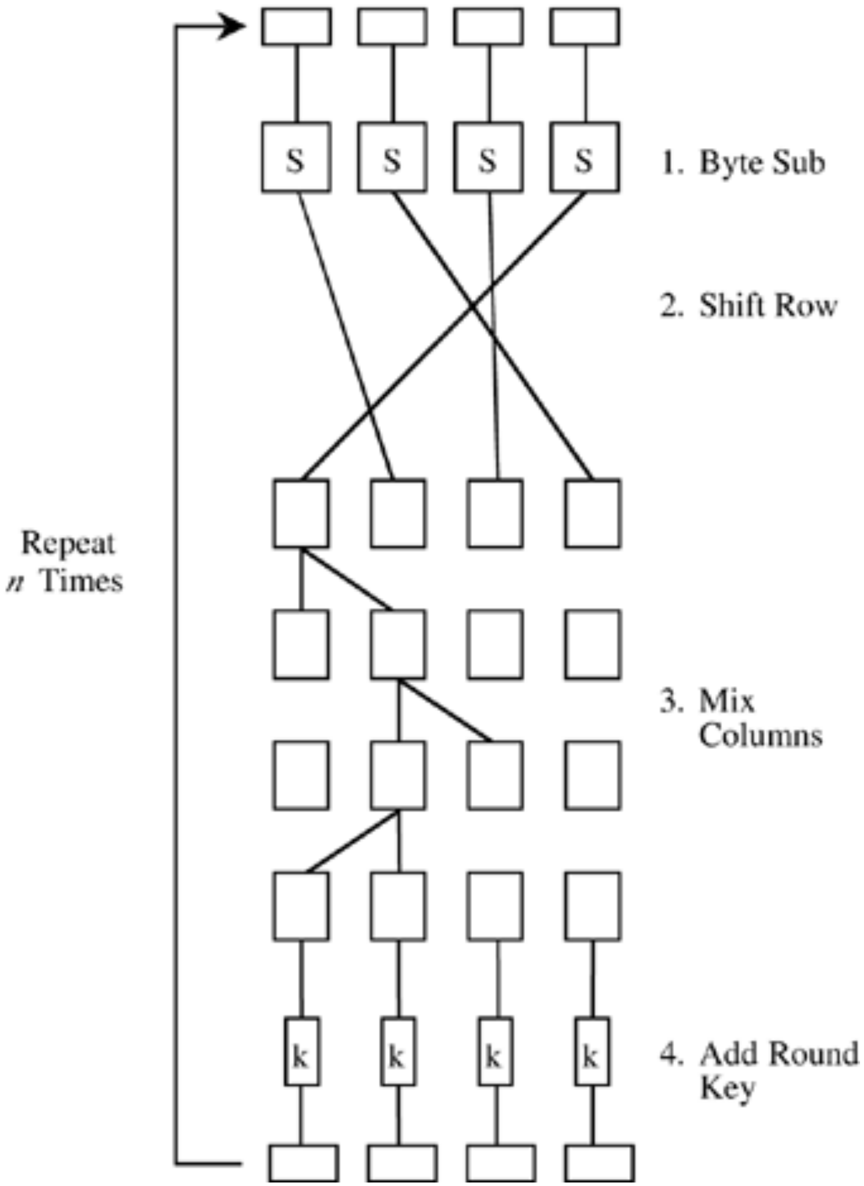




R(k,m): round function
AES-128 n=10

building a block cipher

[slide credit: Dan Boneh, CS155]



Designing good block ciphers is a dark art

Must resist subtle attacks: differential attack, linear attacks, others

Chosen through public design contests

Use build-*break*-build-*break* iteration

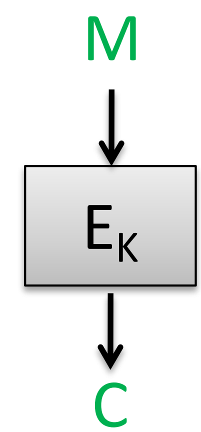
aes round function

Best attacks against AES

Attack	Attack type	Complexity	Year
Bogdanov, Khovratovich, Rechberger	chosen ciphertext, recovers key	$2^{126.1}$ time + some data overheads	2011

- Brute force requires time 2^{128}
- Approximately factor 4 speedup

Are block ciphers good for record layers?



Functional limitations:

- Only encrypt messages that fit in n bits

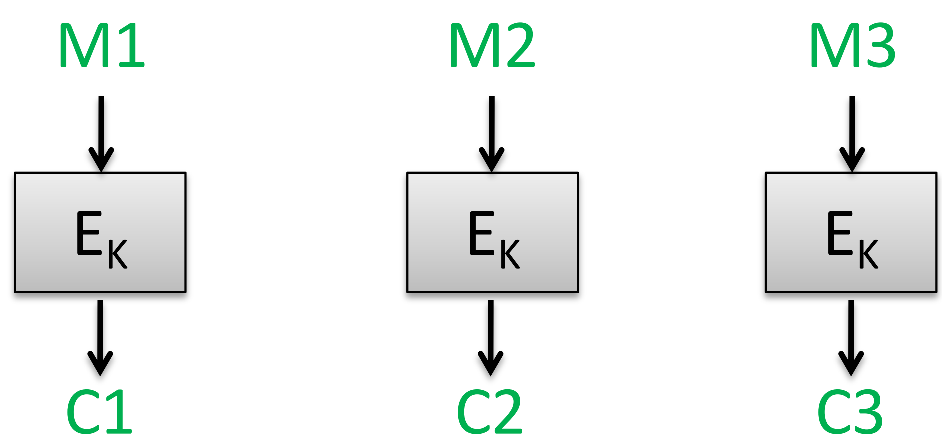
Security limitations:

- Confidentiality: M = M' => E(K,M) = E(K,M')
- Authenticity: any C of length n is valid ciphertext

Block cipher modes of operation

How can we build an encryption scheme for arbitrary message spaces out of a block cipher?

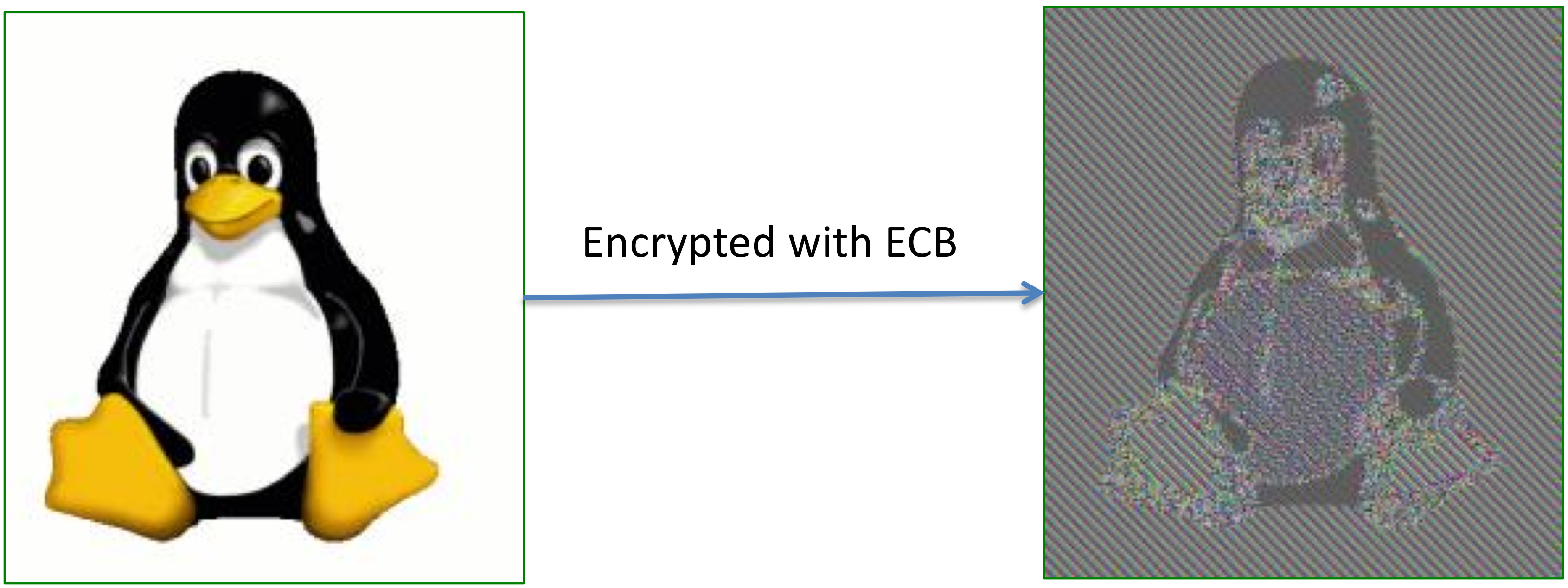
Electronic codebook (ECB) mode
Pad message M to M1,M2,M3,... where each block Mi is n bits
Then:



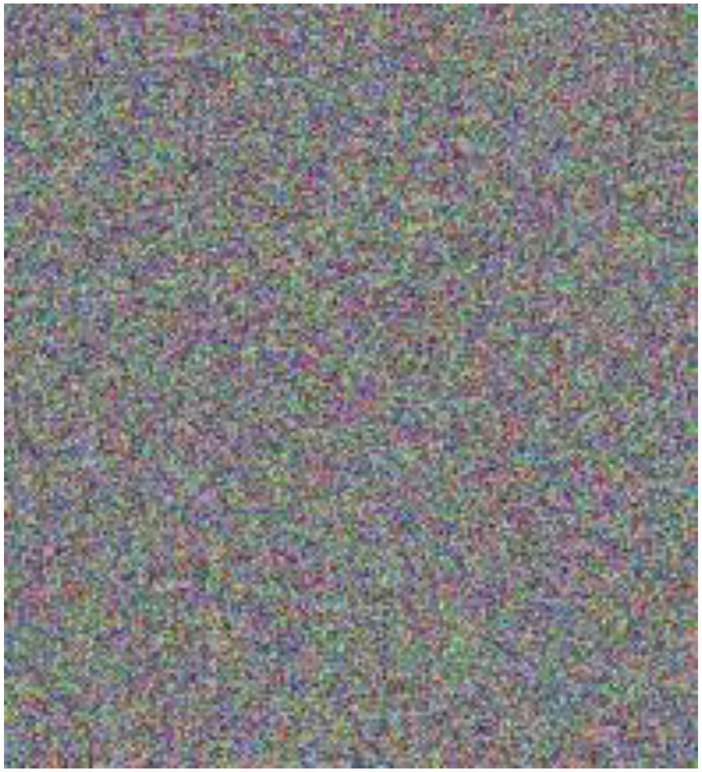
ECB mode is a more complicated looking substitution cipher

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



CTR, GCM, any randomized mode

secure modes

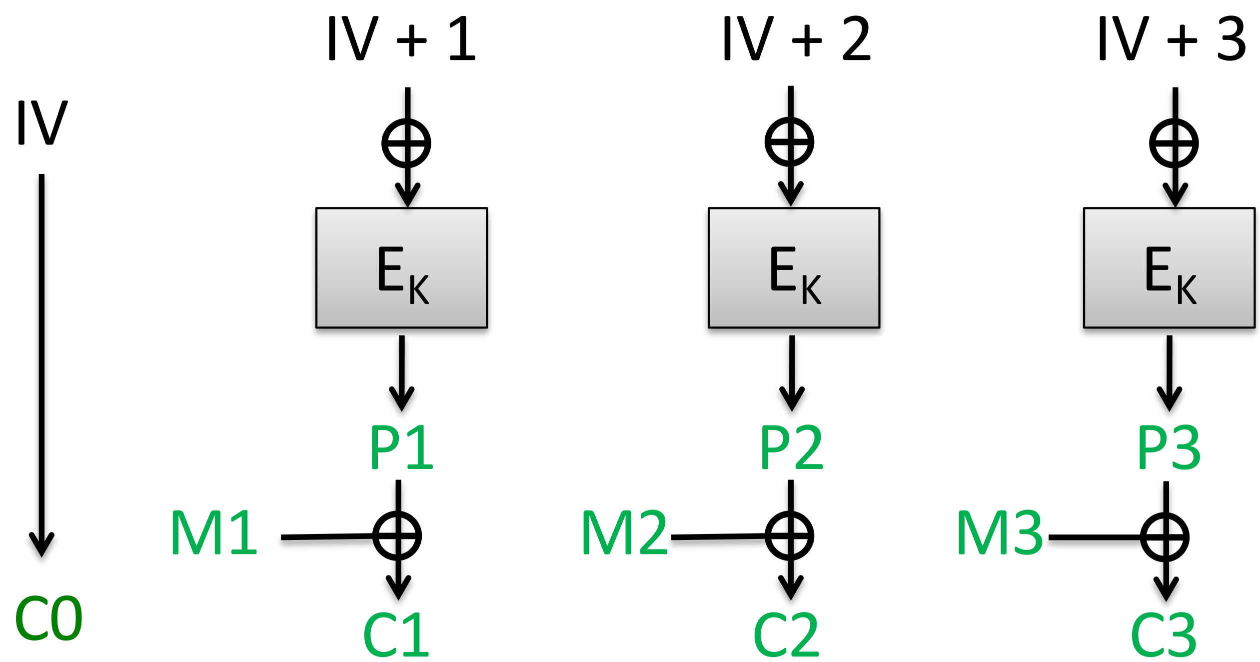
OTP-like encryption using block cipher

Counter mode (CTR)

Pad message M to M1,M2,M3,... where each is n bits except last

IV := rand()

Then:



Maybe use less than full n bits of P3

How do we decrypt?

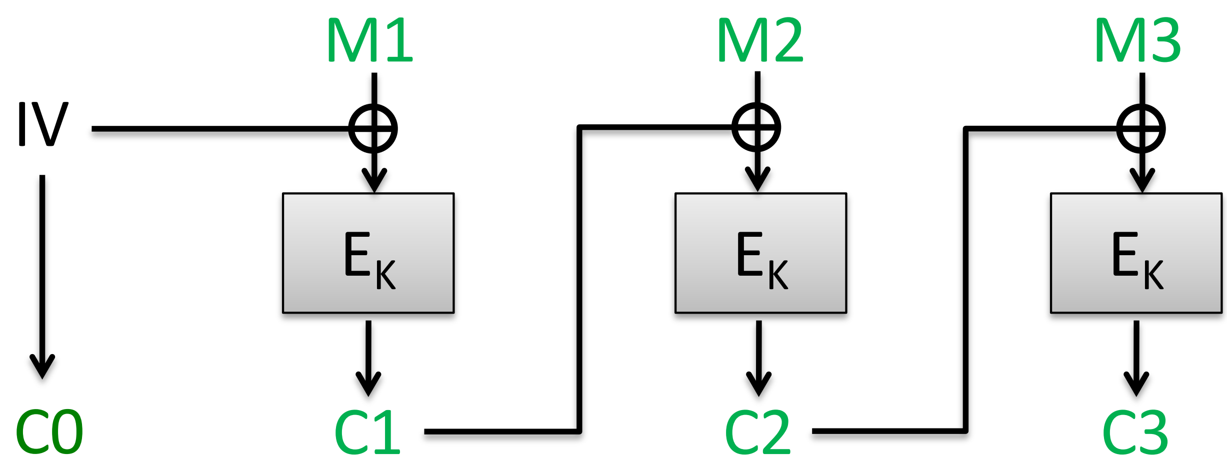
Another option: CBC mode

Ciphertext block chaining (CBC)

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

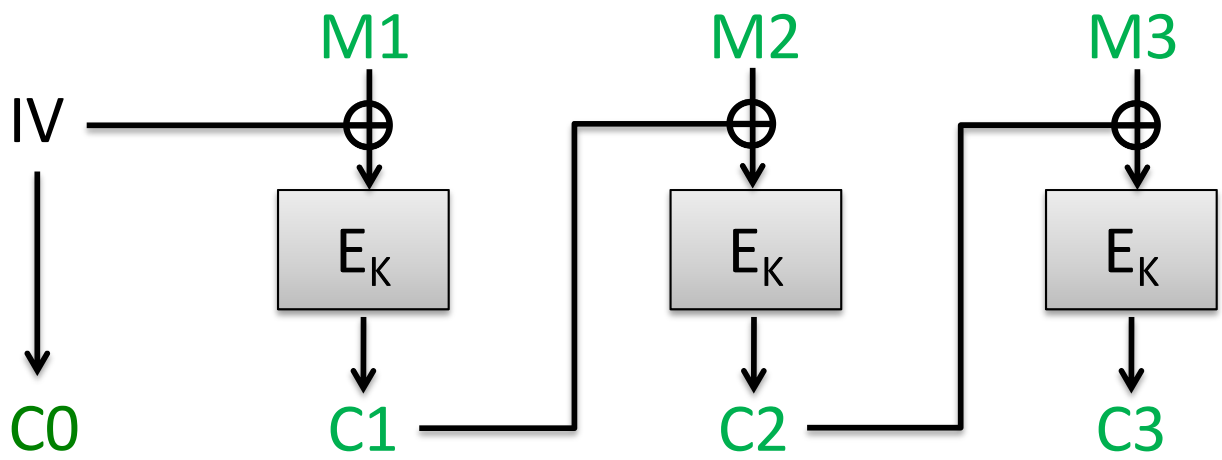
Choose random n-bit string IV

Then:



How do we decrypt?

Security of CBC mode



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?

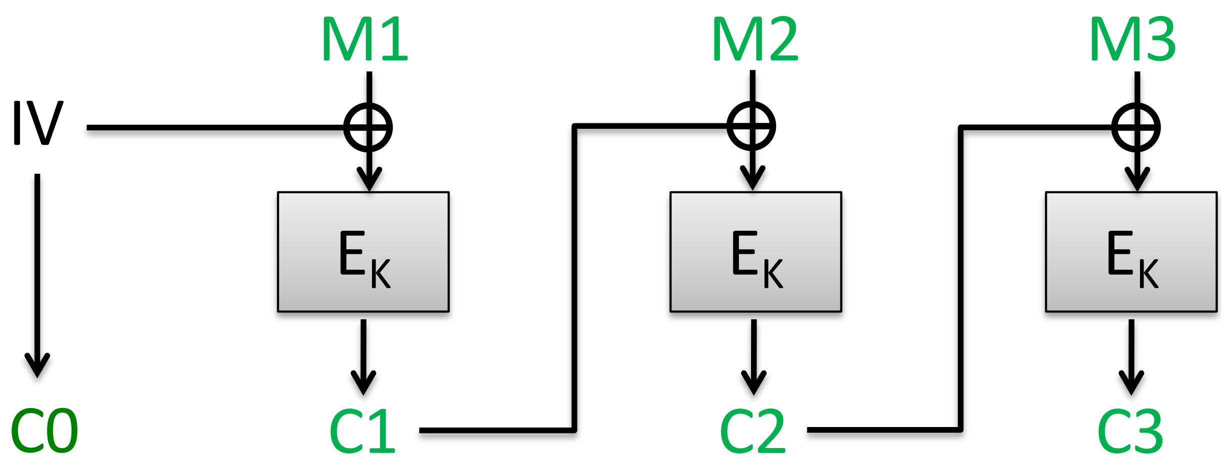
Implies attacker can invert the block cipher without knowing K

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

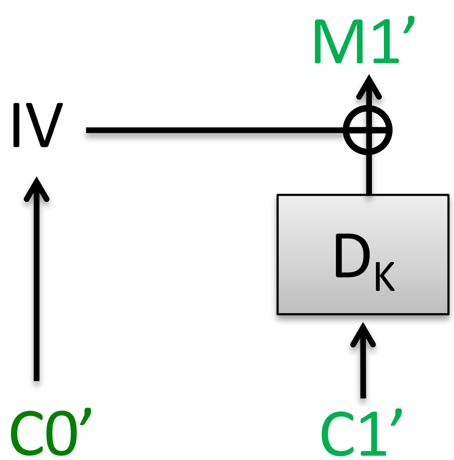
Implies attacker can break PRF security of E

Passive adversaries cannot learn anything about messages

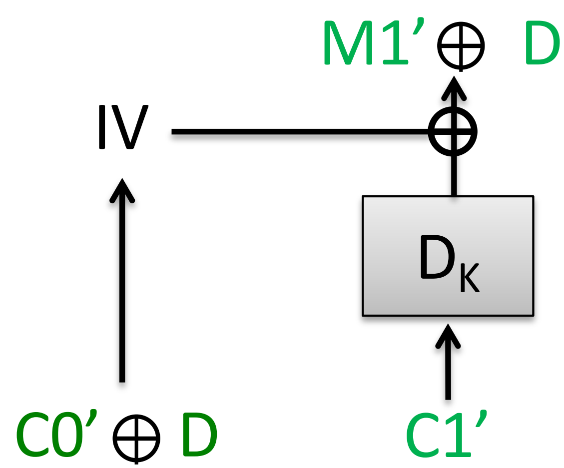
Active security of CBC mode



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



Better yet for any D:



Cutting and Pasting CBC Messages

- Consider the encrypted message

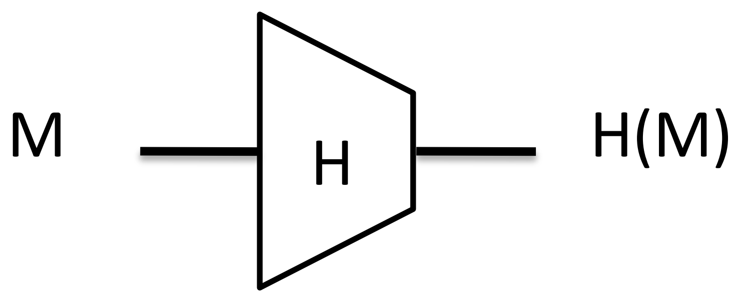
  IV, C1, C2, C3, C4, C5
- The shortened message IV, C1, C2, C3, C4 appears valid
- The truncated message C2, C3, C4, C5 is valid: C2 acts as the IV.
- Even C2, C3, C4 is valid, and will decrypt properly.
- Any subset of a CBC message will decrypt cleanly.
- If we snip out blocks, leaving IV, C1, C4, C5, we only corrupt one block of plaintext.
- Conclusion: if you want message integrity, you have to do it yourself.

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

Hash functions and message authentication

Hash function H maps arbitrary bit string (message) to fixed length string of size m (a digest)



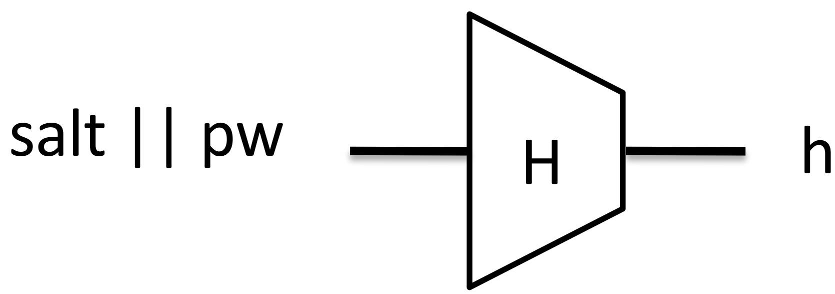
MD5: m = 128 bits
SHA-1: m = 160 bits
SHA-256: m = 256 bits

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)

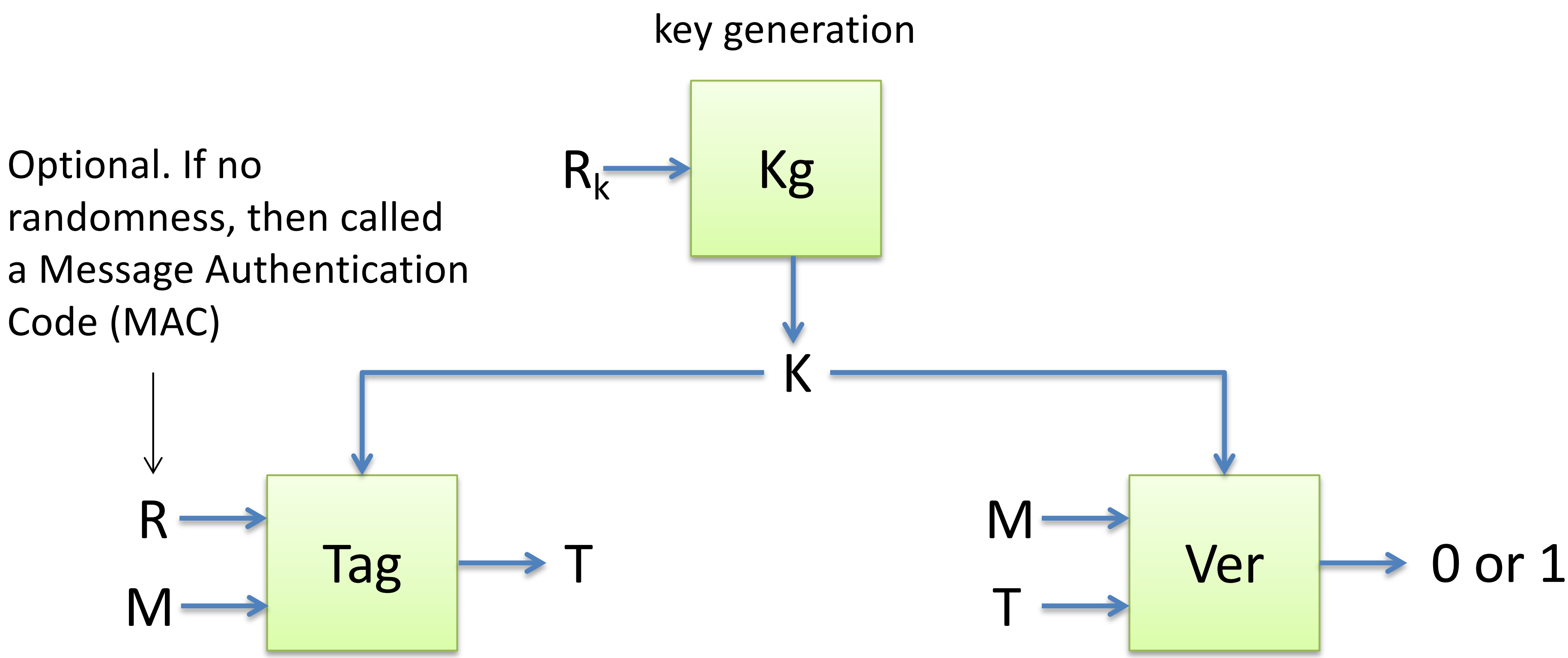
Hash function application example

Password hashing. Choose random salt and store (salt,h) where:



The idea: Attacker, given (salt,h), should not be able to recover pw

Message authentication



C is a ciphertext

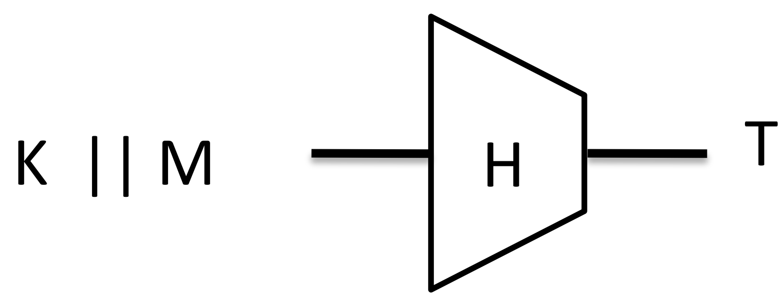
Correctness: Ver(K , Tag(K,M,R)) = 1 with probability 1 over randomness used

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

Attempt 1

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

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



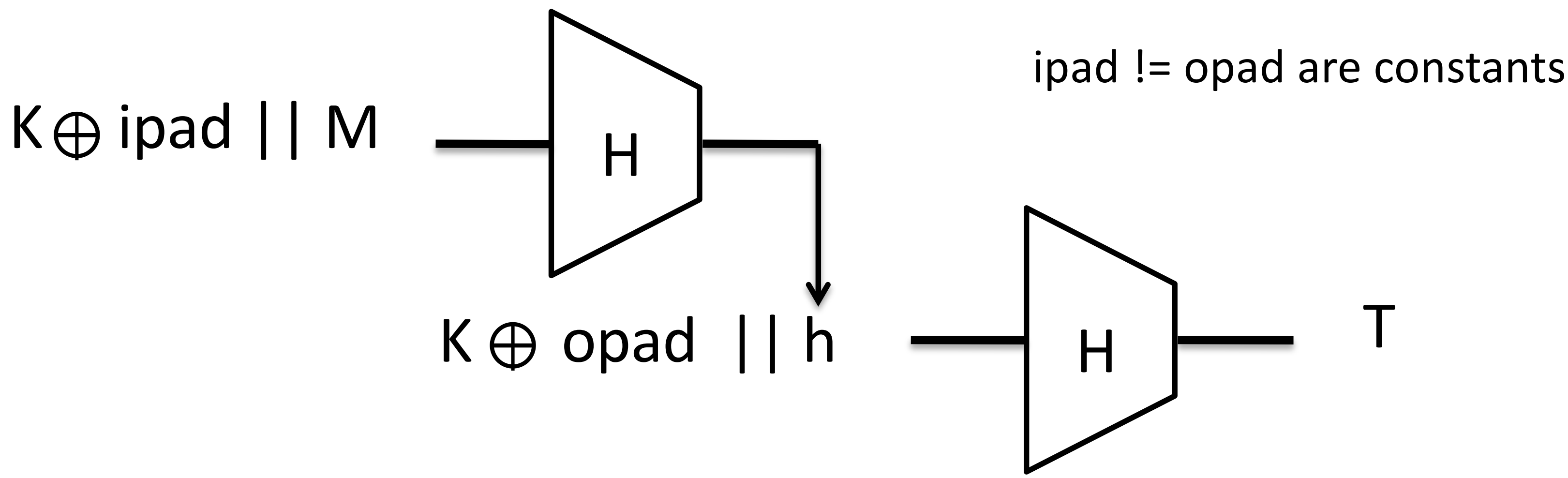
To verify a M,T pair, check if HMAC(K,M) = T

But: what if I want to append: HMACK(K, M||M') by continuing hash

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:

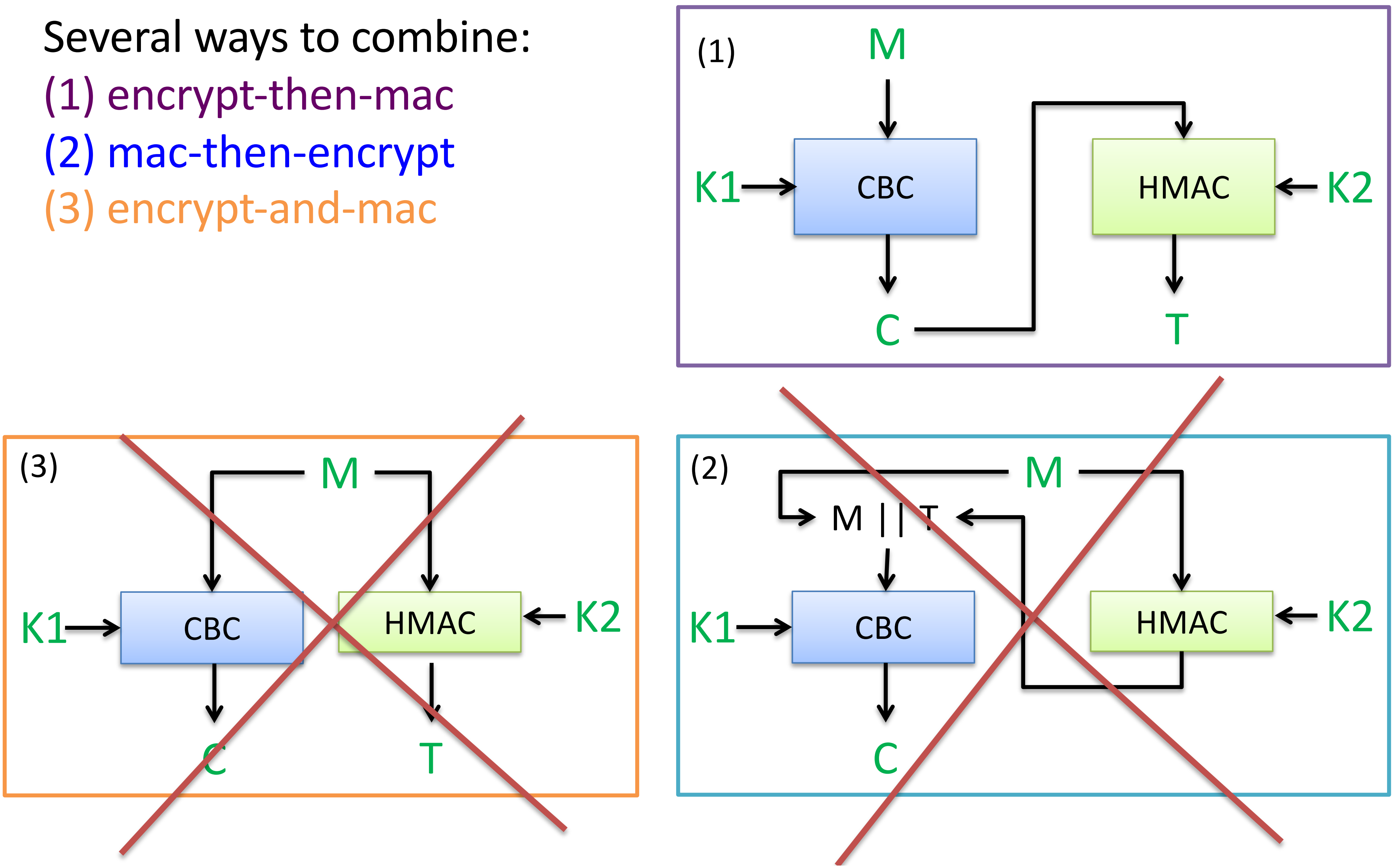


To verify a M,T pair, check if HMAC(K,M) = T

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

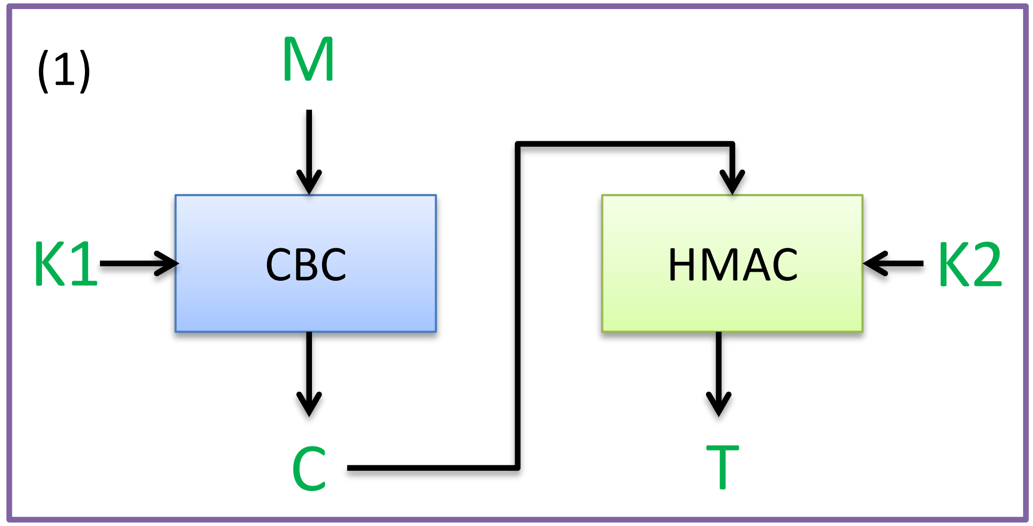
Build a new scheme from CBC and HMAC
Kg outputs CBC key K1 and HMAC key K2

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

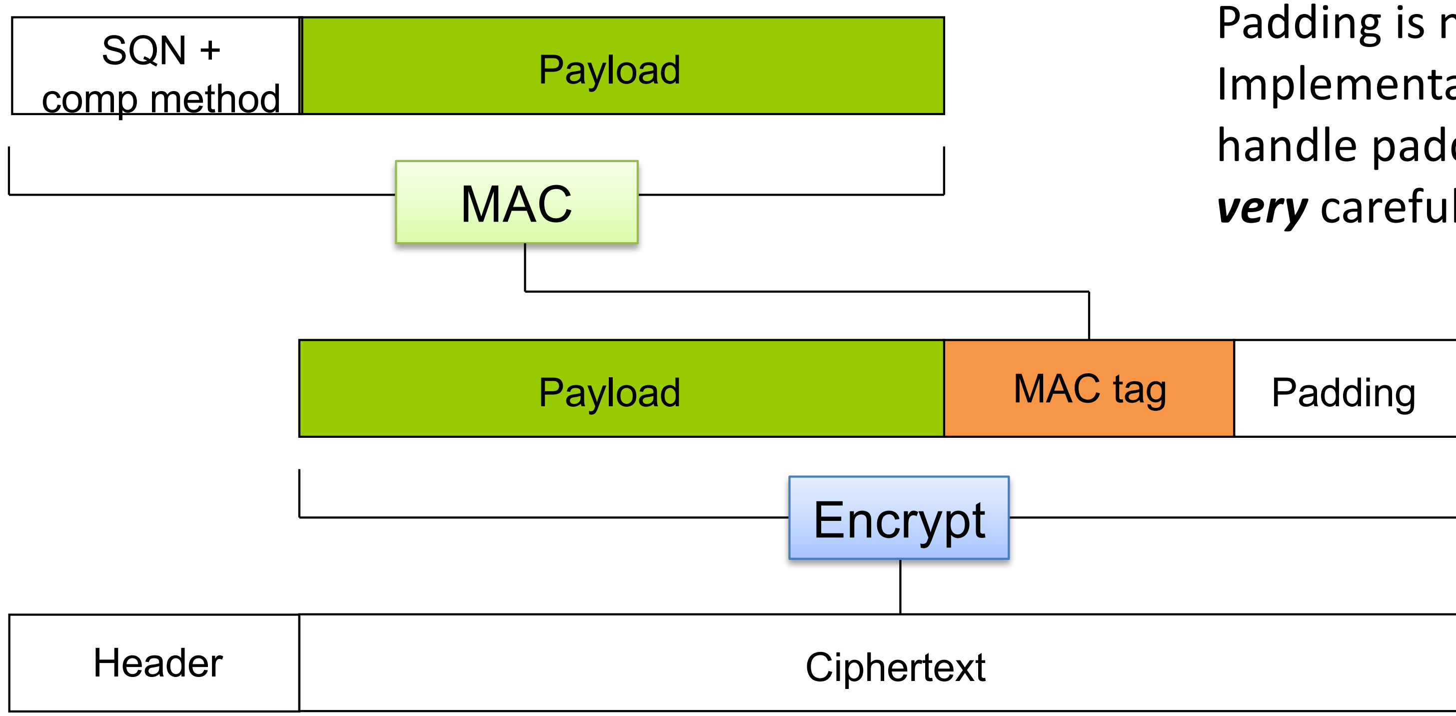
Several ways to combine:
(1) encrypt-then-mac
(2) mac-then-encrypt
(3) encrypt-and-mac



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

https://www.iacr.org/archive/crypto2001/21390309.pdf

TLS record protocol: MAC-Encode-Encrypt (MEE)



Padding is not MAC'd. Implementations must handle padding checks ***very*** carefully.

MAC	HMAC-MD5, HMAC-SHA1, HMAC-SHA256
Encrypt	CBC-AES128, CBC-AES256, CBC-3DES, RC4-128

Dedicated authenticated encryption schemes

Attack	Inventors	Notes
OCB (Offset Codebook)	Rogaway	One-pass
GCM (Galios Counter Mode)	McGrew, Viega	CTR mode plus specialized MAC
CWC	Kohno, Viega, Whiting	CTR mode plus Carter-Wegman MAC
CCM	Housley, Ferguson, Whiting	CTR mode plus CBC-MAC
EAX	Wagner, Bellare, Rogaway	CTR mode plus OMAC

Symmetric Encryption Advice

Never use CTR mode or CBC mode by themselves

Passive security is almost never good enough!!

Encrypt-then-MAC better than MAC-then-Encrypt, Encrypt and MAC

Dedicated modes that have been analyzed thoroughly are also good