Cryptography Overview

Cryptography

- Is
  - A tremendous tool
  - The basis for many security mechanisms
- Is not
  - The solution to all security problems
  - Reliable unless
    - implemented properly
    - used properly
  - Something to try invent yourself unless
    - you spend a lot of time becoming an expert
    - you subject your design to outside review

Auguste Kerckhoffs

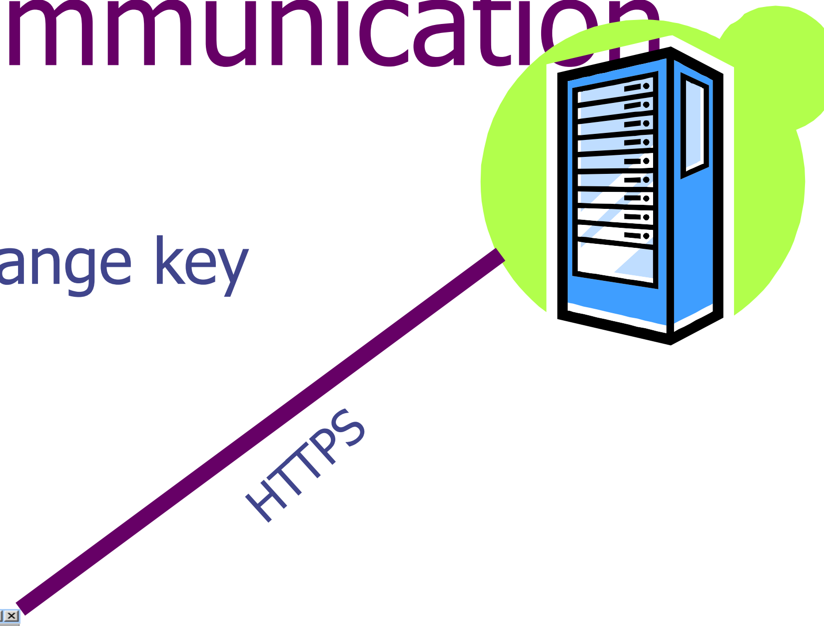
- A cryptosystem should be secure even if **everything** about the system, except the secret key,
is public knowledge.

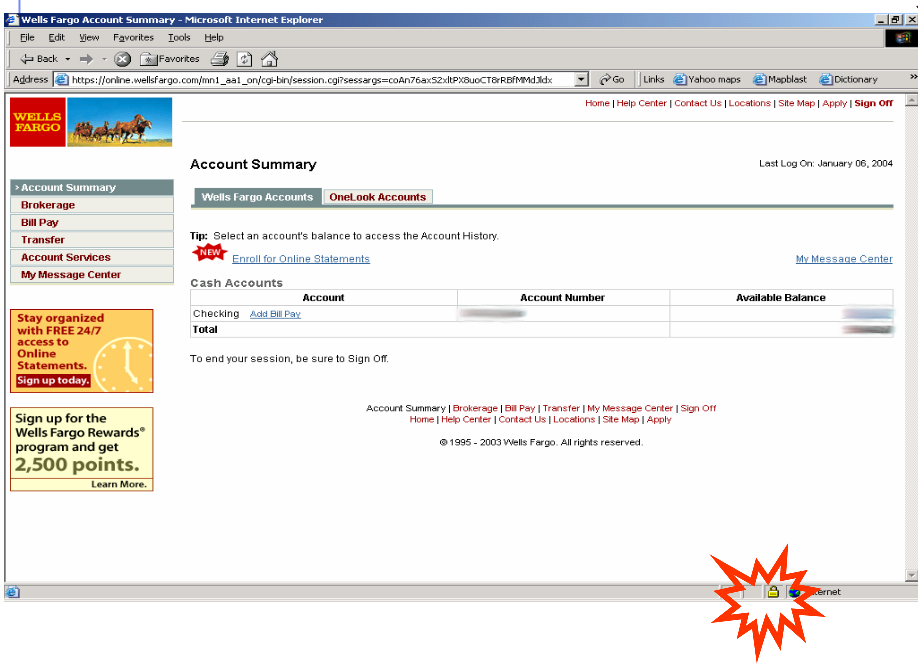
baptised as **Jean-Guillaume-Hubert-Victor-François-Alexandre-Auguste Kerckhoffs von Nieuwenhof**

Goal 1:secure communication

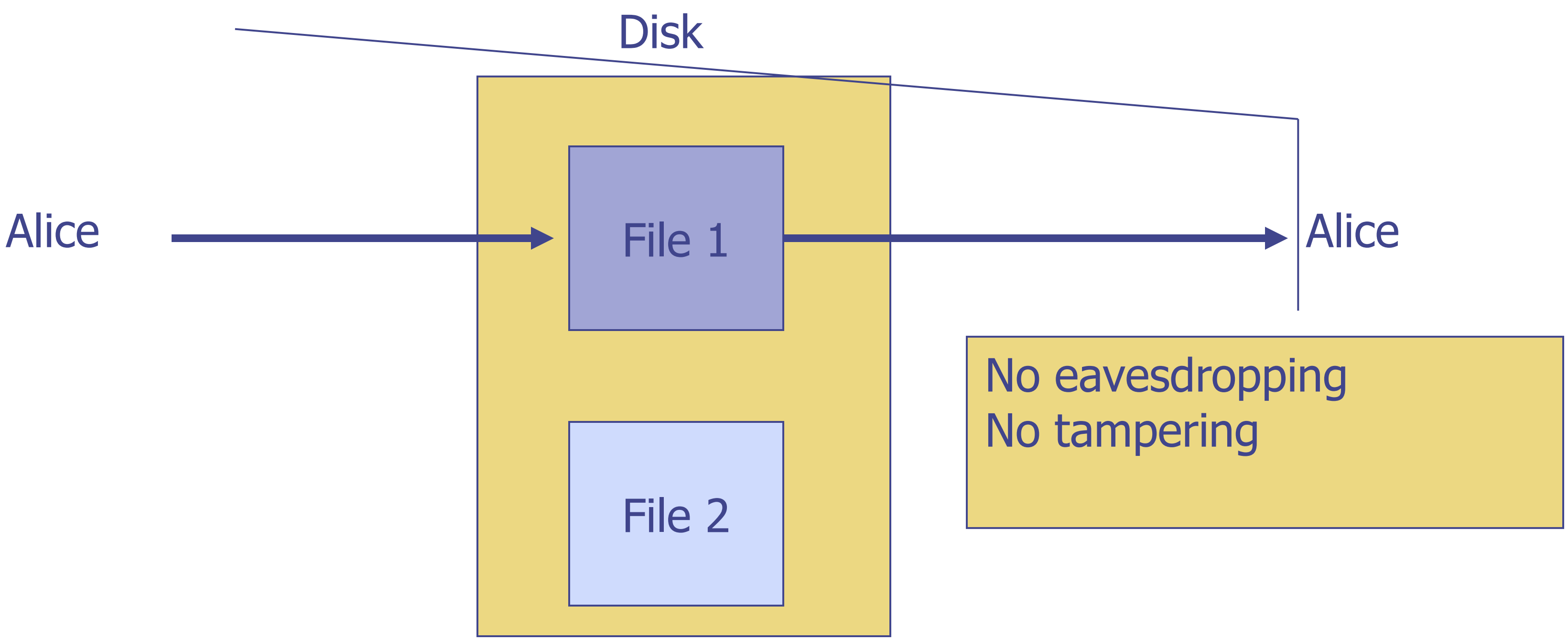
Step 1: Session setup to exchange key

Step 2: encrypt data





Goal 2: Protected files



Analogous to secure communication:

Alice today sends a message to Alice tomorrow

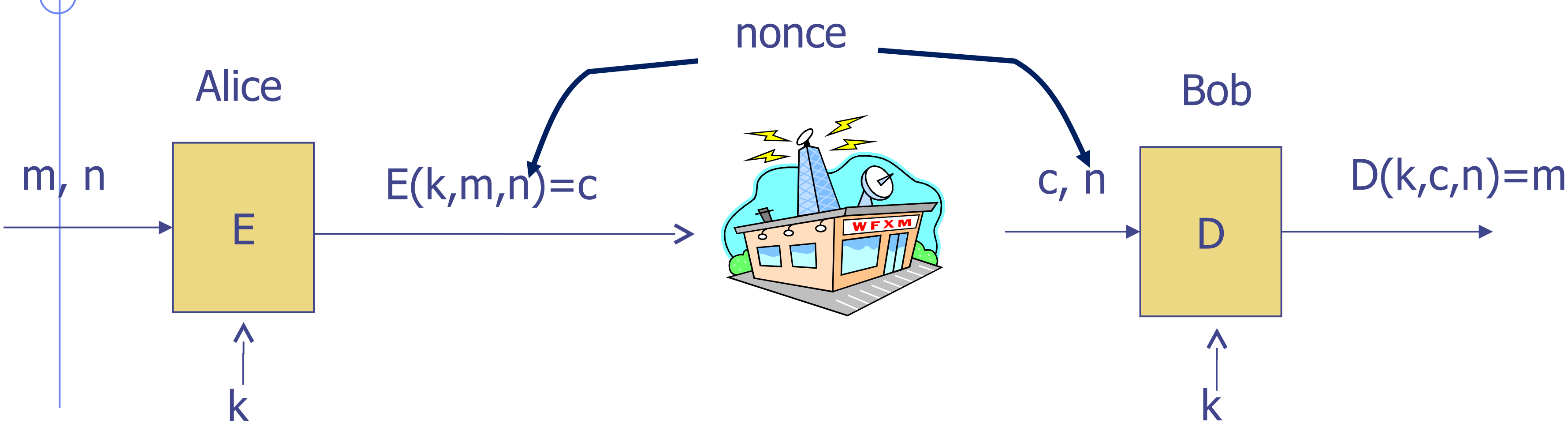
Taxonomy of Cryptographic attacks

a) Brute force
b) Differential cryptanalysis
c) Linear cryptanalysis
d) Meet-in-the-middle
e) Chosen-ciphertext
f) Chosen-plaintext
g) Ciphertext-only
h) Replay attack
i) Known-plaintext
j Power analysis
k) Timing
l) Man-in-the-middle

Symmetric Cryptography

Assumes parties already share a secret key

Building block: sym. encryption



E, D: cipher k: secret key (e.g. 128 bits)

m, c: plaintext, ciphertext n: nonce (aka IV)

Encryption algorithm is publicly known

- Never use a proprietary cipher

Use Cases

Single use key: (one time key)

- Key is only used to encrypt one message

  encrypted email: new key generated for every email
- No need for nonce (set to 0)

Multi use key: (many time key)

- Key used to encrypt multiple messages

  SSL: same key used to encrypt many packets
- Need either *unique* nonce or *random* nonce

First example: One Time Pad
(single use key)

- **Vernam** (1917)

Key:	0	1	0	1	1	1	0	0	1	0
Plaintext:	1	1	0	0	0	1	1	0	0	0

$\oplus$

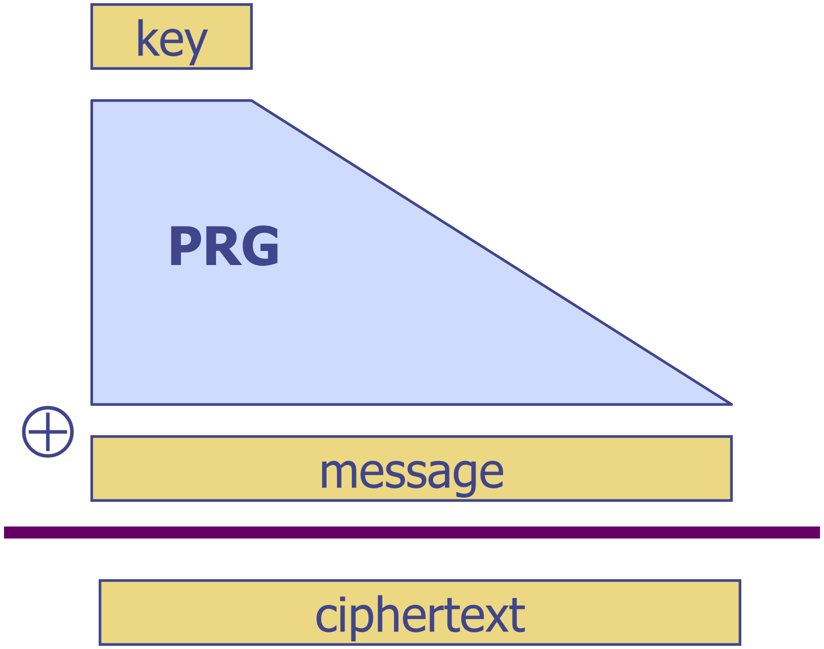
Ciphertext:	1	0	0	1	1	0	1	0	1	0

- **Shannon '49:**
  - OTP is "secure" against ciphertext-only attacks = attacker can only access the chiper text

Stream ciphers (single use key)

Problem: OTP key is as long the message

Solution: Pseudo random key -- stream ciphers



$c \leftarrow \text{PRG}(k) \oplus m$

Stream ciphers: RC4 (113MB/sec) , SEAL (293MB/sec)

PRG = pseudo random generator

- Unfortunately, generating random numbers looks a lot easier than it really is.
- It is fundamentally impossible to produce truly random numbers on any deterministic device.
- “Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.” Von Neumann
- The best we can hope for are pseudo-random numbers, a stream of numbers that appear as if they were generated randomly.

Dangers in using stream ciphers

One time key !! "Two time pad" is insecure:

$$C1 \leftarrow m1 \oplus PRG(k)$$

$$C2 \leftarrow m2 \oplus PRG(k)$$

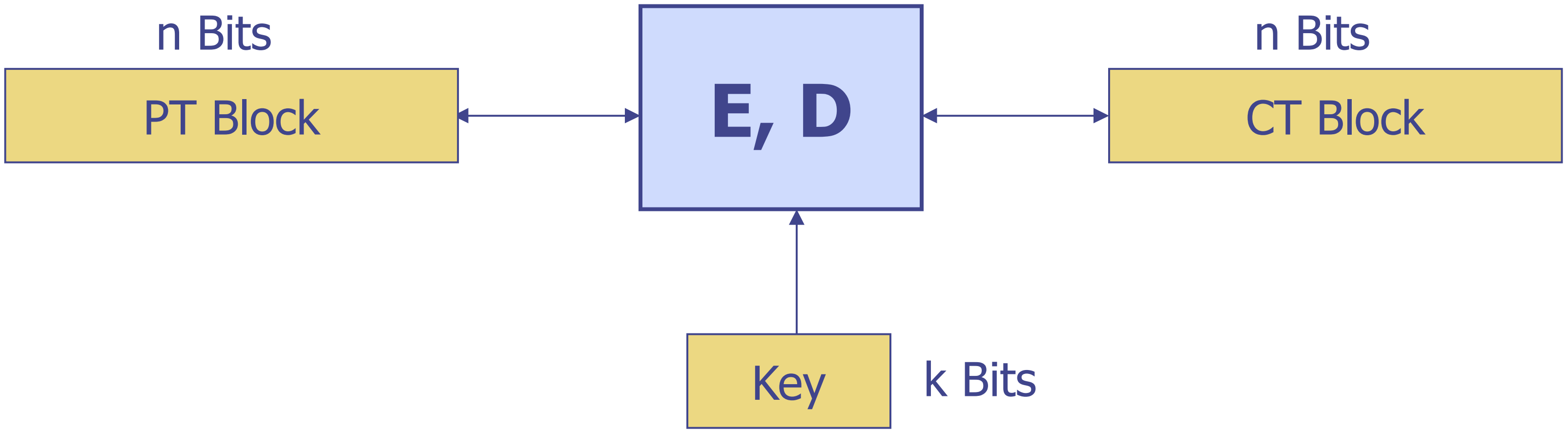
Eavesdropper does:

$$C1 \oplus C2 \quad \rightarrow \quad m1 \oplus m2$$

Enough redundant information in English that:

$$m1 \oplus m2 \rightarrow \quad m1 , \; m2$$

Block ciphers: crypto work horse



Canonical examples:

1. 3DES: n= 64 bits, k = 168 bits
2. AES: n=128 bits, k = 128, 192, 256 bits

IV handled as part of PT block

Building a block cipher

Input: (m, k)

Repeat simple "mixing" operation several times

- DES: Repeat 16 times:

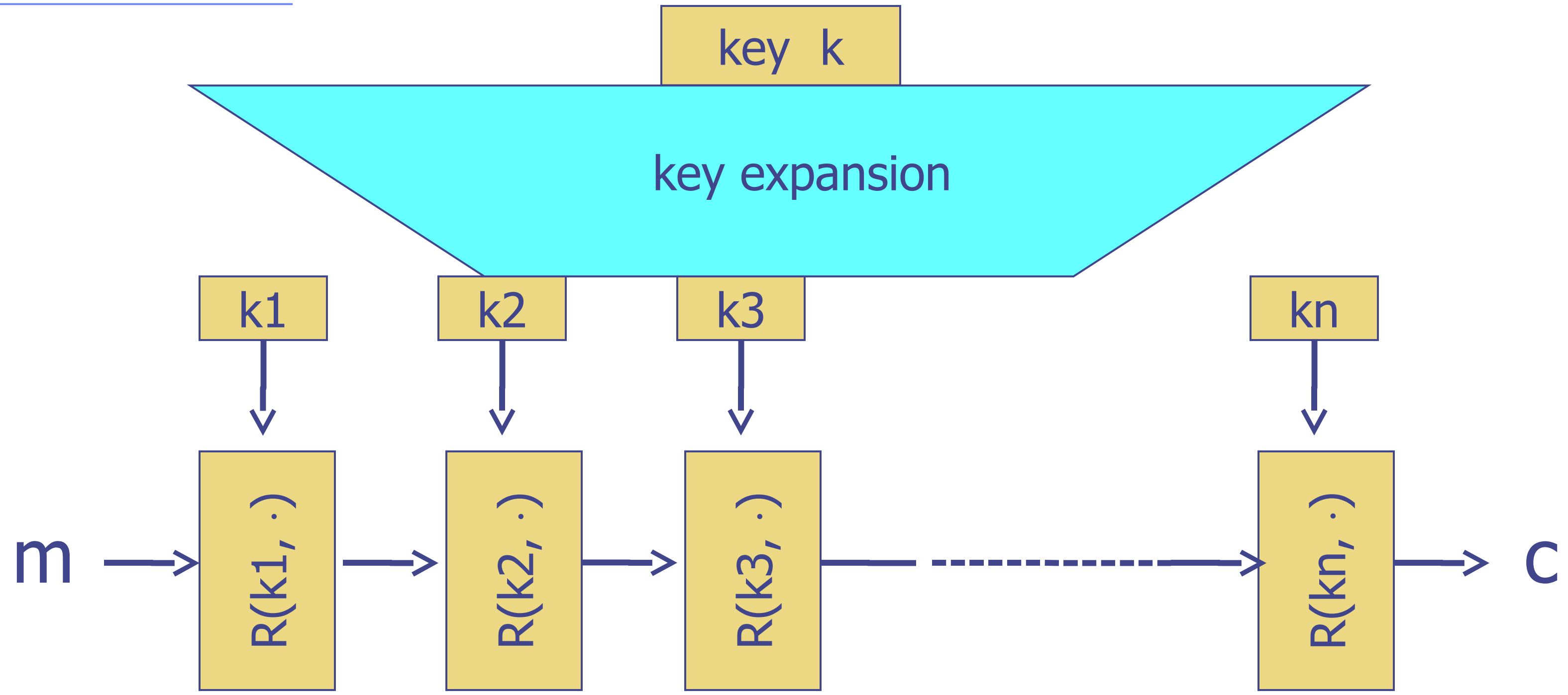
$$mL \leftarrow mR$$
$$mR \leftarrow mL \oplus F(k, mR)$$

- AES-128: Mixing step repeated 10 times

Difficult to design: must resist subtle attacks

- differential attacks, linear attacks, brute-force, …

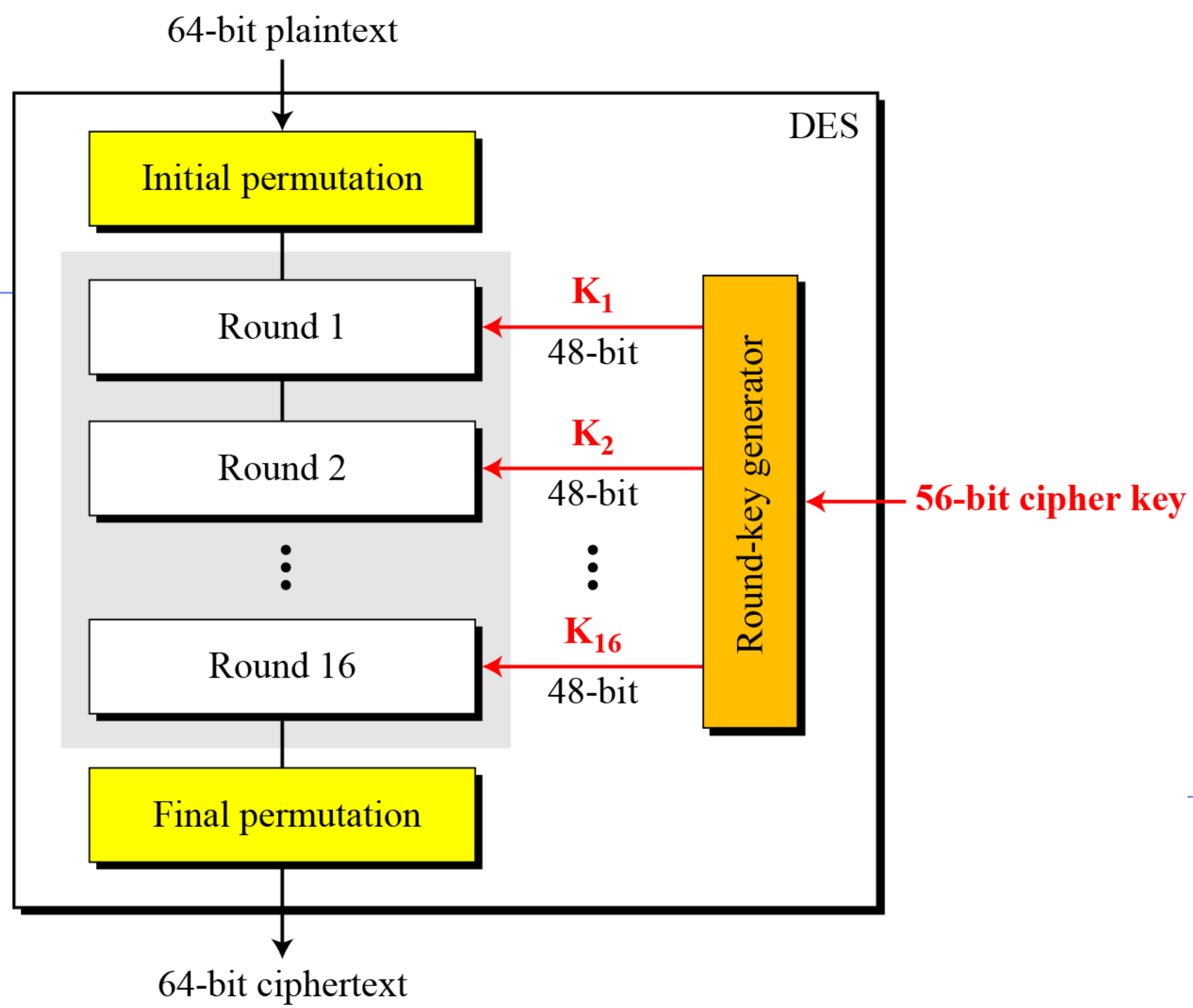
Block Ciphers Built by Iteration



R(k,m): round function

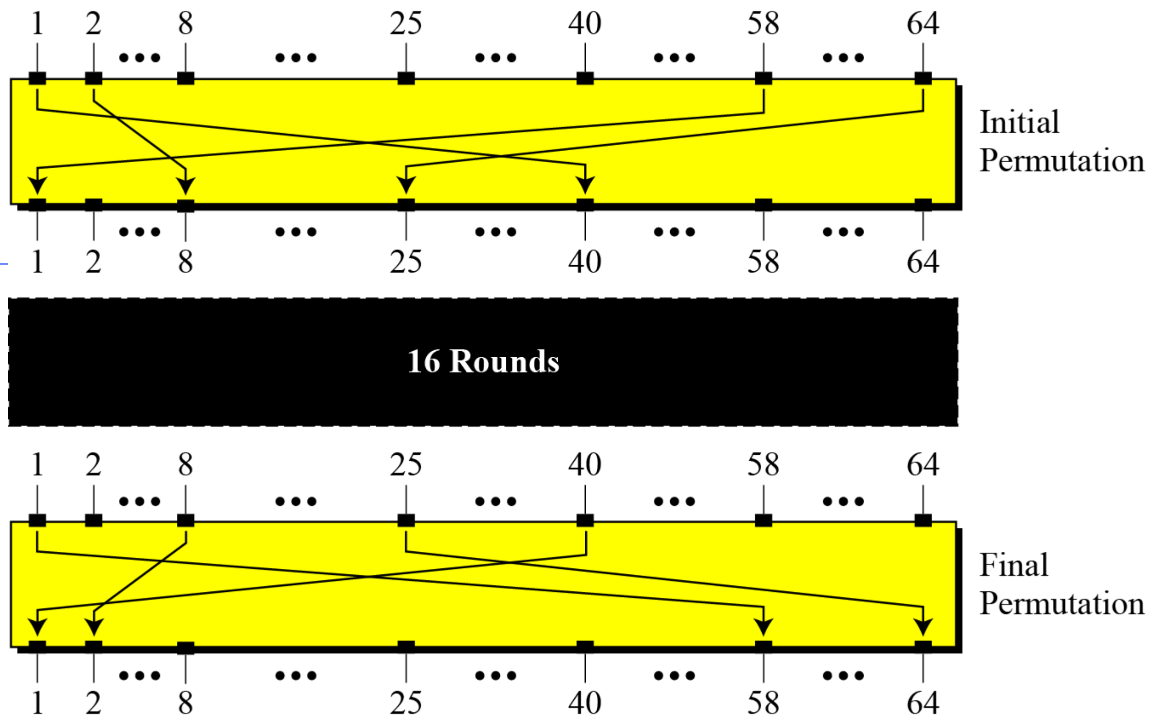
for DES (n=16), for AES (n=10)

General structure of DES



Initial and Final Permutations

Initial and final permutation steps in DES



Initial and final permutation tables

Initial Permutation								*Final Permutation*							
58	50	42	34	26	18	10	02	40	08	48	16	56	24	64	32
60	52	44	36	28	20	12	04	39	07	47	15	55	23	63	31
62	54	46	38	30	22	14	06	38	06	46	14	54	22	62	30
64	56	48	40	32	24	16	08	37	05	45	13	53	21	61	29
57	49	41	33	25	17	09	01	36	04	44	12	52	20	60	28
59	51	43	35	27	19	11	03	35	03	43	11	51	19	59	27
61	53	45	37	29	21	13	05	34	02	42	10	50	18	58	26
63	55	47	39	31	23	15	07	33	01	41	09	49	17	57	25

Rounds

DES uses 16 rounds. Each round of DES is a Feistel cipher.

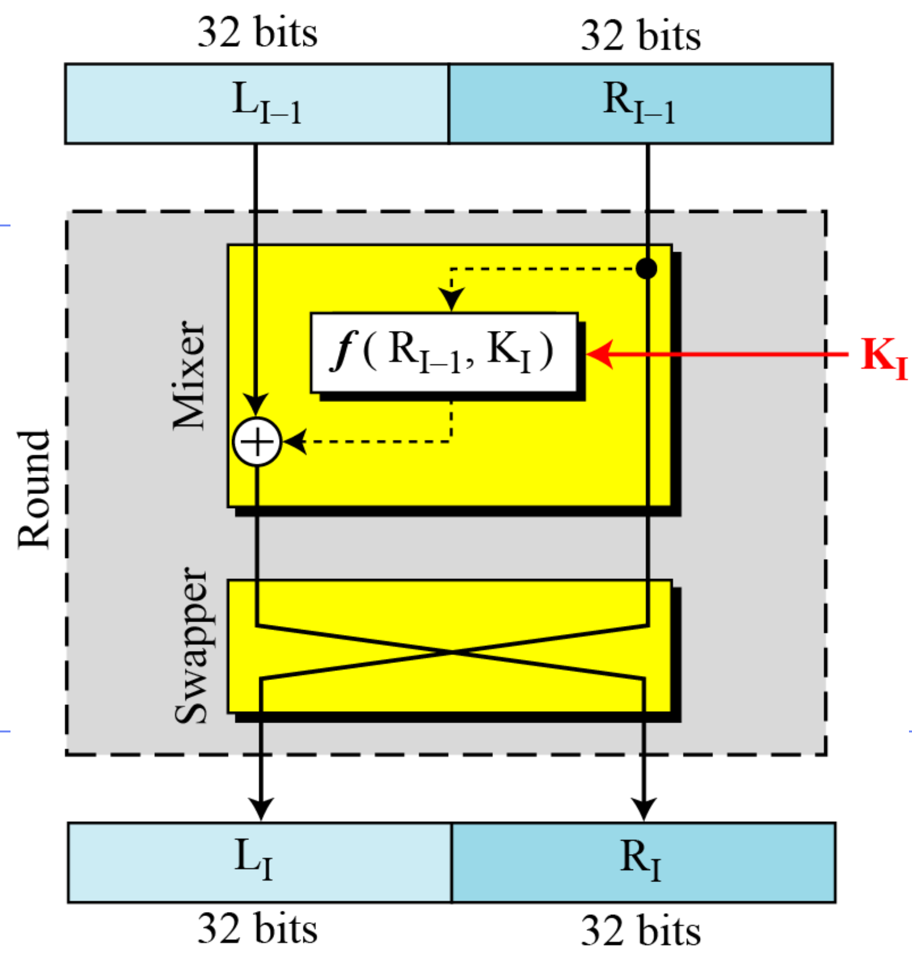
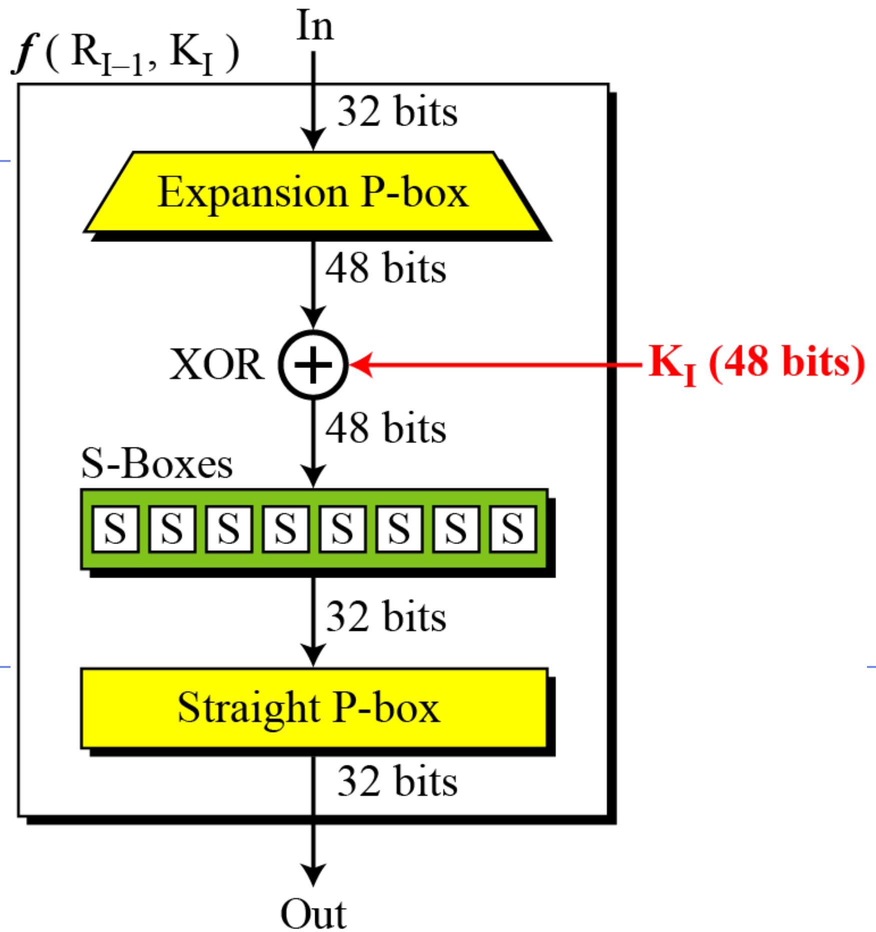


Figure 6.4
A round in DES (encryption site)

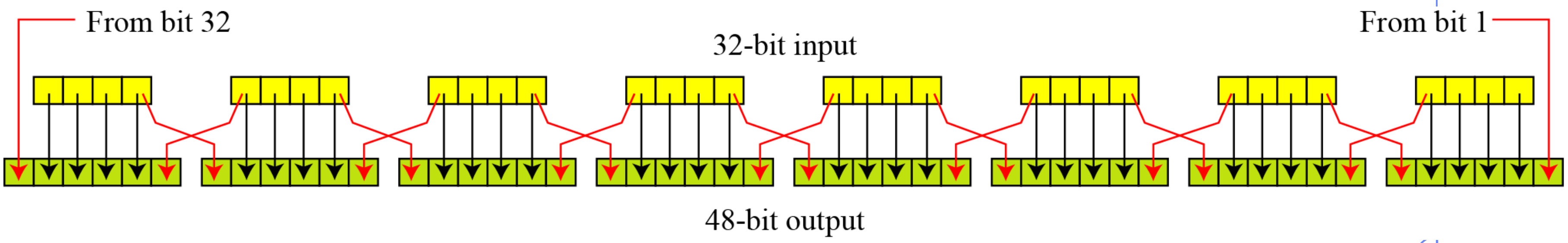
DES Function

The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.



Expansion P-box

R_{I-1} is a 32-bit input and K_I is a 48-bit key, we first to expand R_{I-1} to 48 bits.

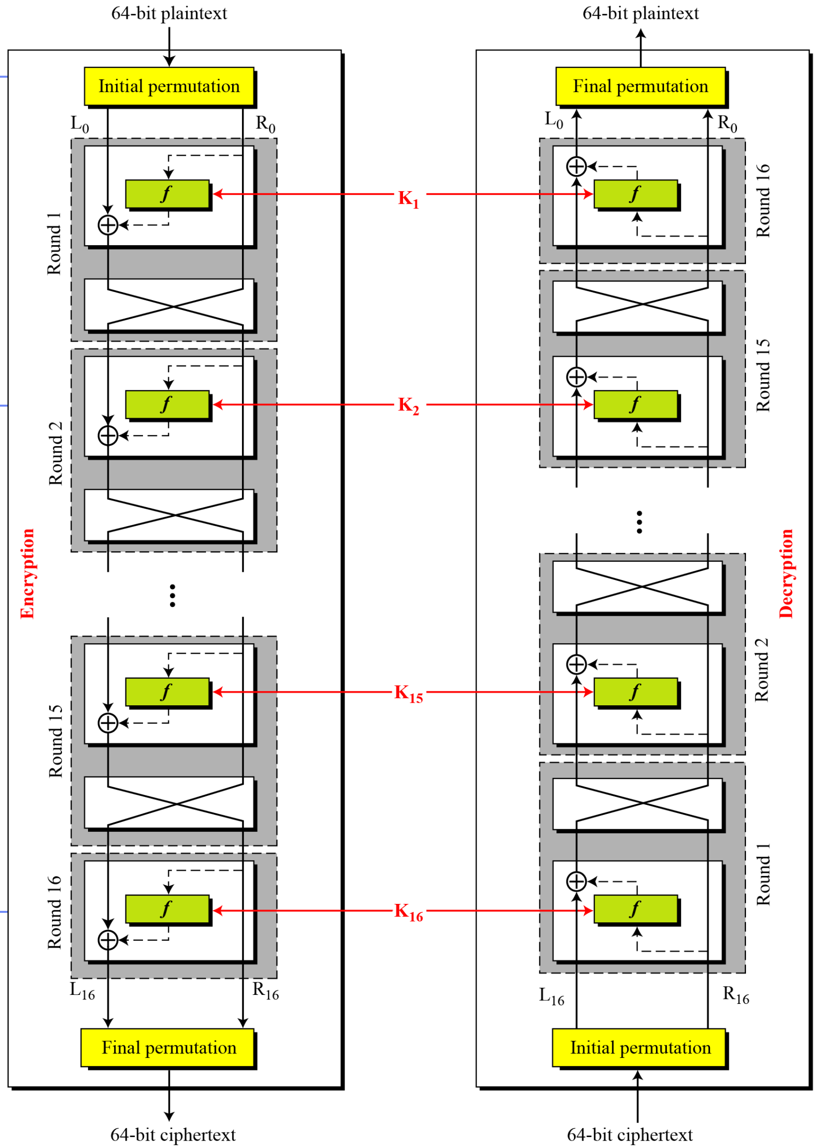


Expansion P-box table

The relationship between the input and output can be defined mathematically, but DES uses a table

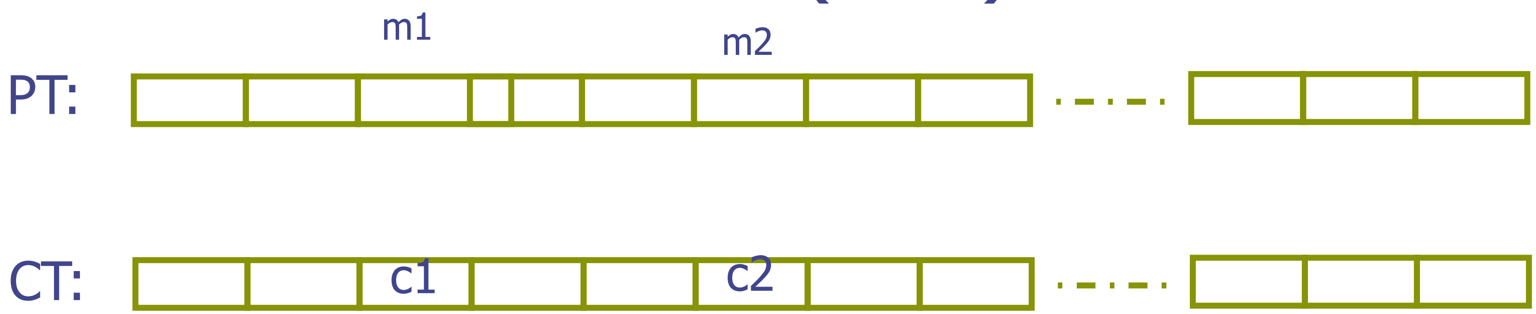
32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

DES cipher and reverse cipher = same hw component for encryption and decryption



Incorrect use of block ciphers

Electronic Code Book (ECB):

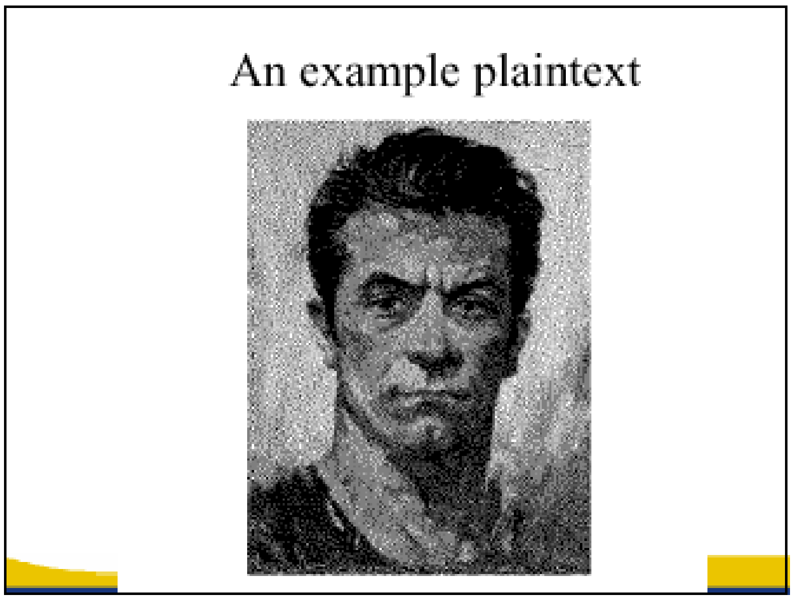


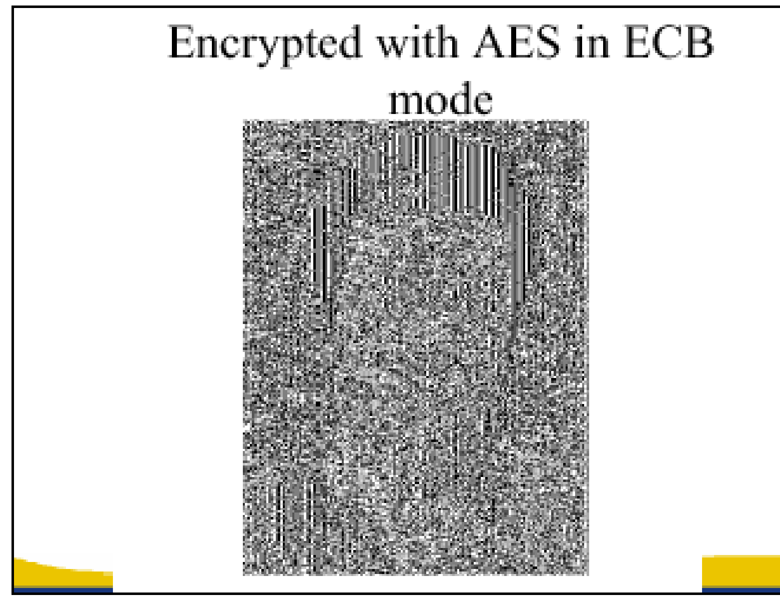
Parallel encryption of the various blocks through the same key

<u>Problem</u>:

- if m1=m2 then c1=c2

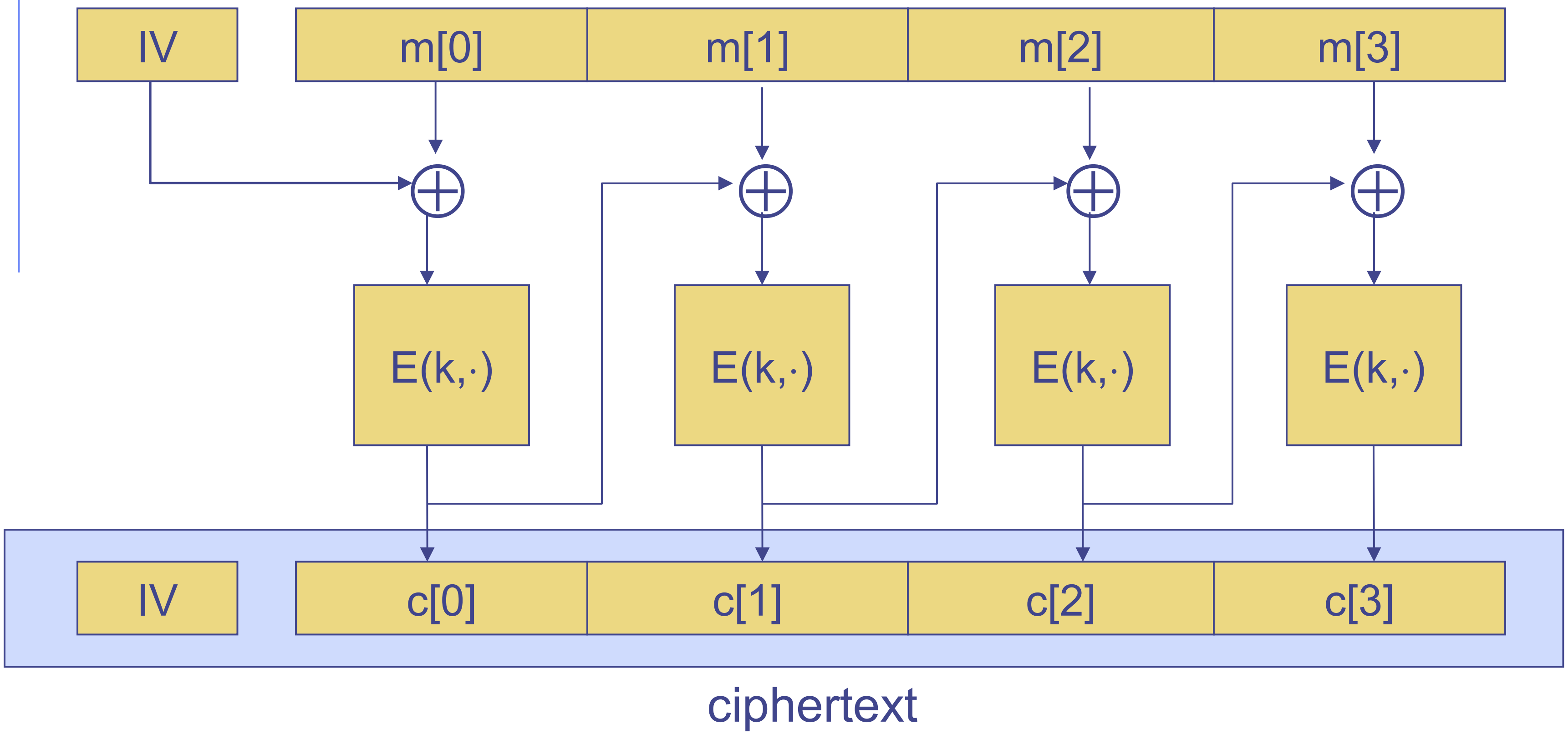
In pictures





Correct use of block ciphers I: CBC mode

E a secure PRP. Cipher Block Chaining with random IV:



Q: how to do decryption?

Use cases: choosing an IV

Single use key: no IV needed (IV=0)

Multi use key: (CPA Security)

Best: use a fresh <u>*random*</u> IV for every message

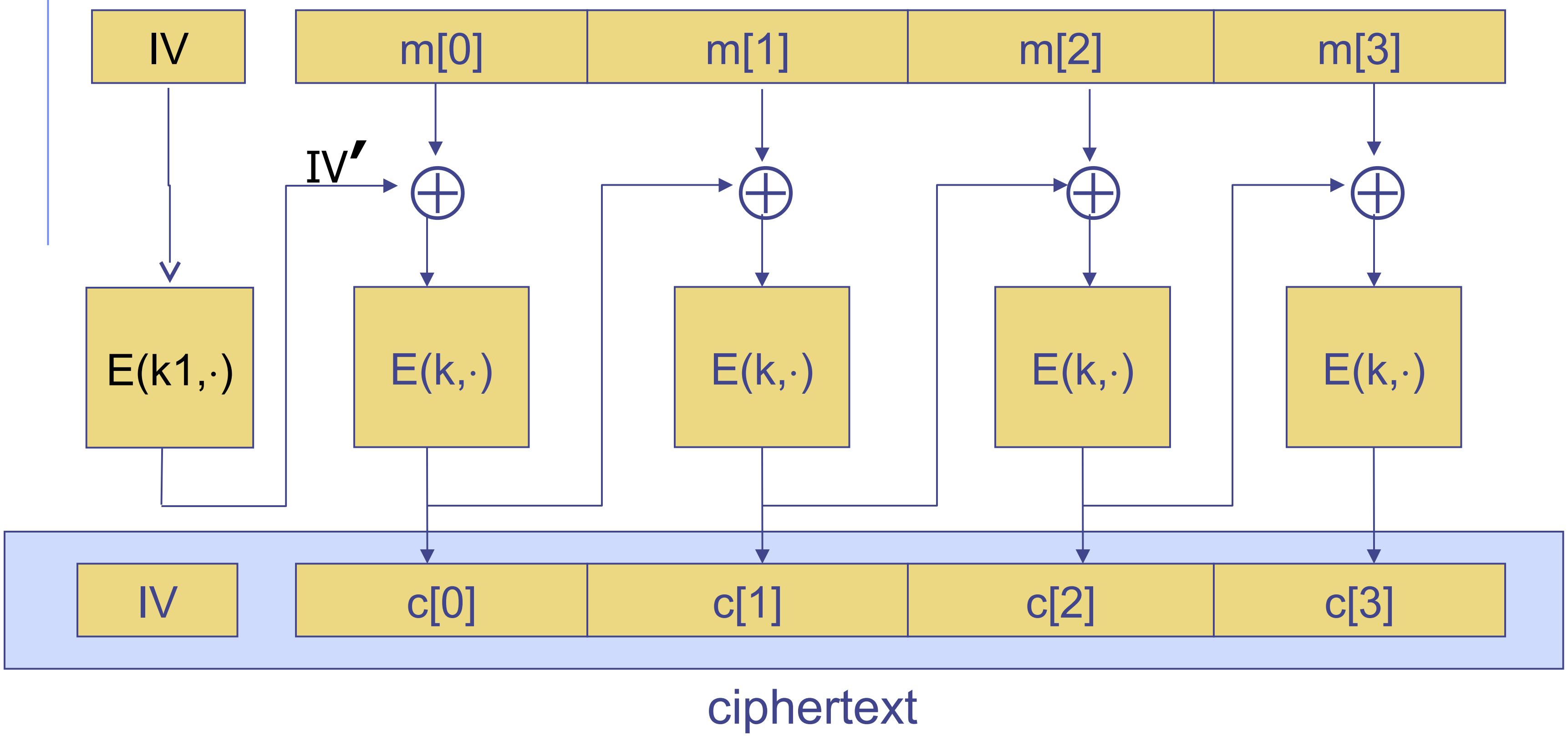
Can use <u>*unique*</u> IV (e.g counter)

but then first step in CBC <u>must be</u> IV′ ← E(k1,IV)

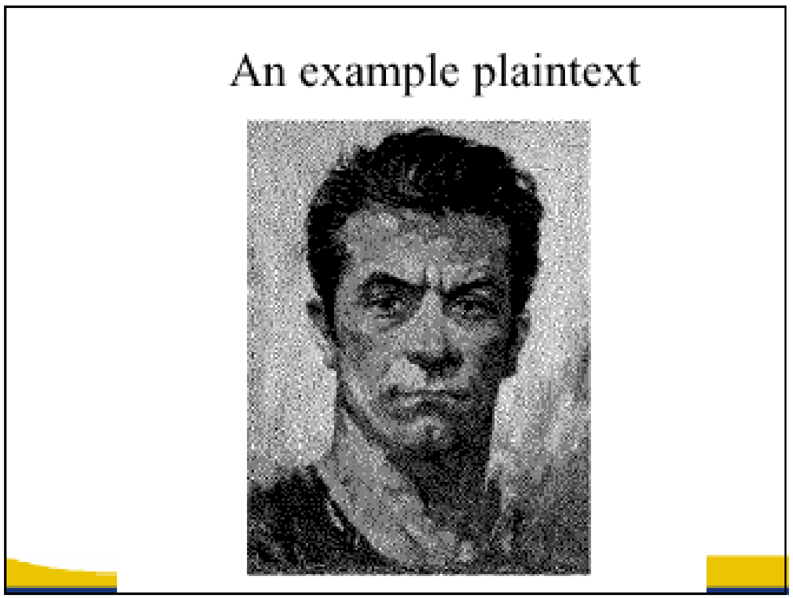
benefit: may save transmitting IV with ciphertext

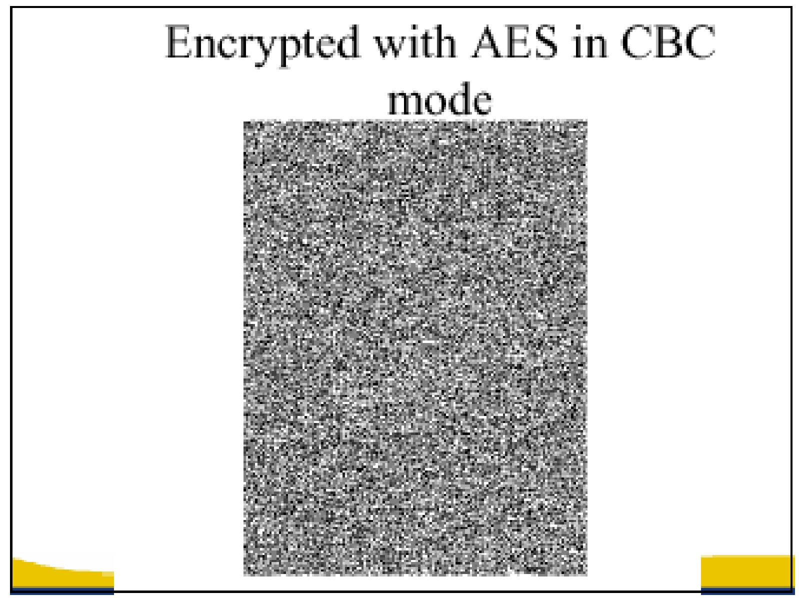
CBC with Unique IVs

unique IV means: (k,IV) pair is used for only one message
may be predictable so use E(k1,·) as PRF



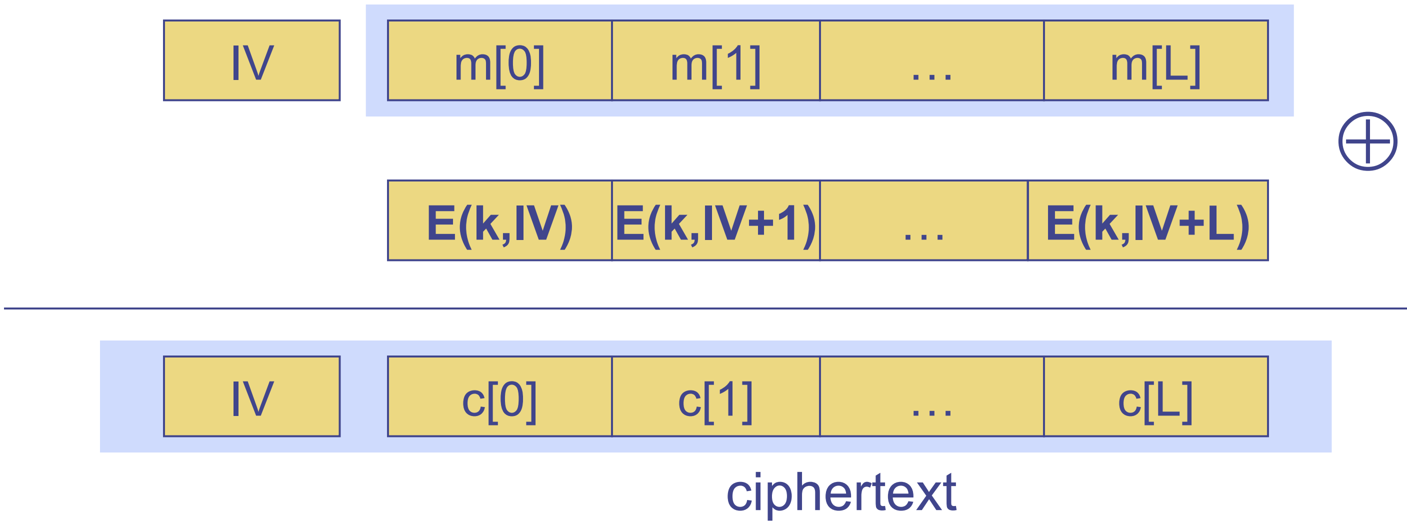
In pictures





Correct use of block ciphers II: CTR mode

Counter mode with a random IV: (parallel encryption)



Performance: Crypto++ 5.2.1 [Wei Dai]

Pentium 4, 2.1 GHz (on Windows XP SP1, Visual C++ 2003)

Cipher	Block/key size	Speed (MB/sec)
RC4		113
SEAL		293
3DES	64/168	9
AES	128/128	61

Data integrity

Message Integrity: MACs

- Goal: message integrity. No confidentiality.
  - ex: Protecting public binaries on disk.

Generate tag:
tag $\leftarrow$ S(k, m)

Verify tag:
V(k, m, tag) $\stackrel{?}{=}$ `yes'

note: non-keyed checksum (CRC) is an insecure MAC !!

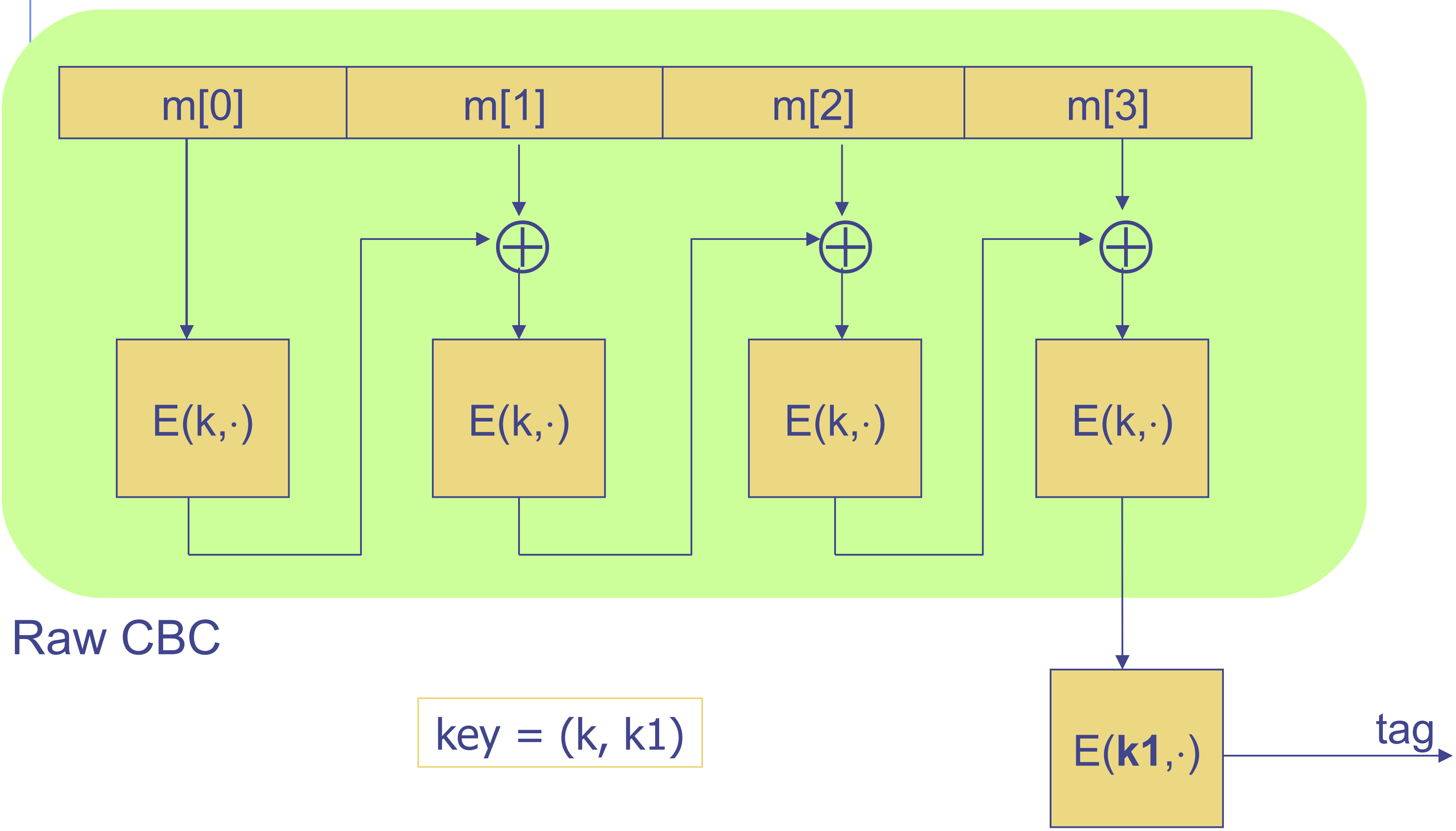
Secure MACs

- Attacker information: chosen message attack
  - for m1,m2,…,mq attacker is given ti ← S(k,mi)
- Attacker's goal: existential forgery.
  - produce some **new** valid message/tag pair (m,t).

    (m,t) ∉ { (m1,t1) , … , (mq,tq) }

- A secure PseudoRandomFunction gives a secure MAC:
  - S(k,m) = F(k,m)
  - V(k,m,t): `yes' if t = F(k,m) and `no' otherwise.

Construction 1:   ECBC



Construction 2: HMAC (Hash-MAC)

Most widely used MAC on the Internet.

H: hash function.

example: SHA-256 ; output is 256 bits

Building a MAC out of a hash function:

Standardized method: HMAC

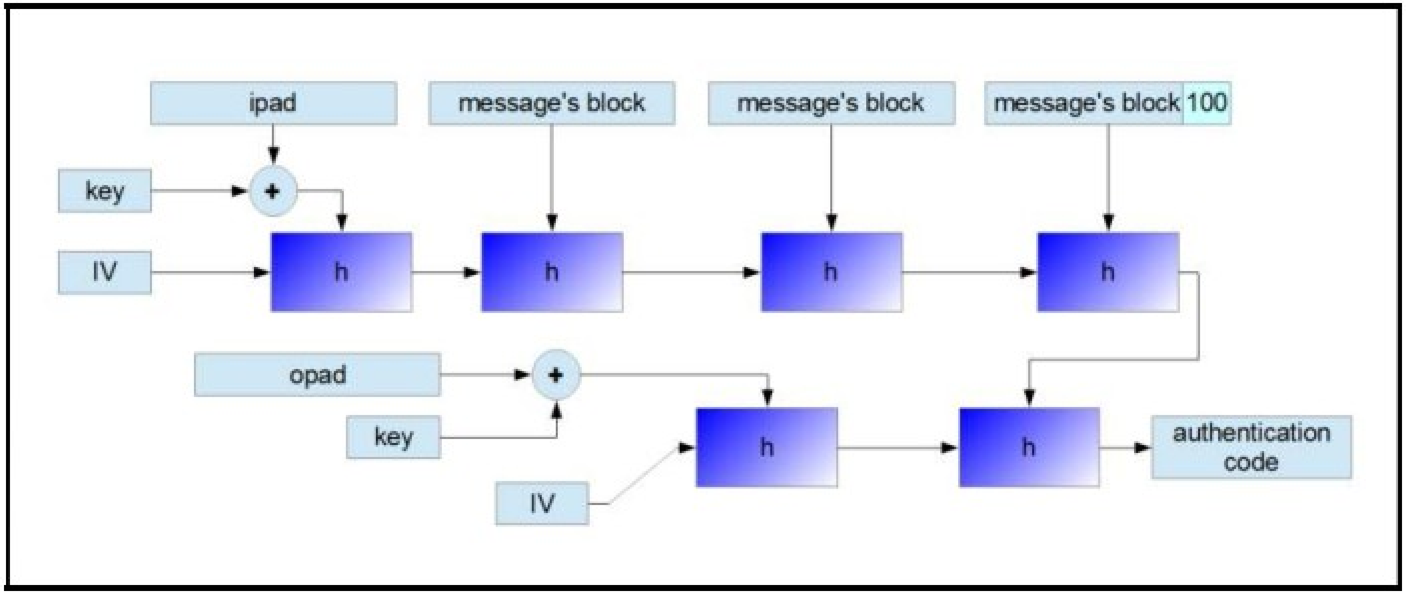
$$S(k, m) = H(k \oplus opad \,\|\, \mathbf{H(k \oplus ipad \,\|\, m)})$$

HMAC

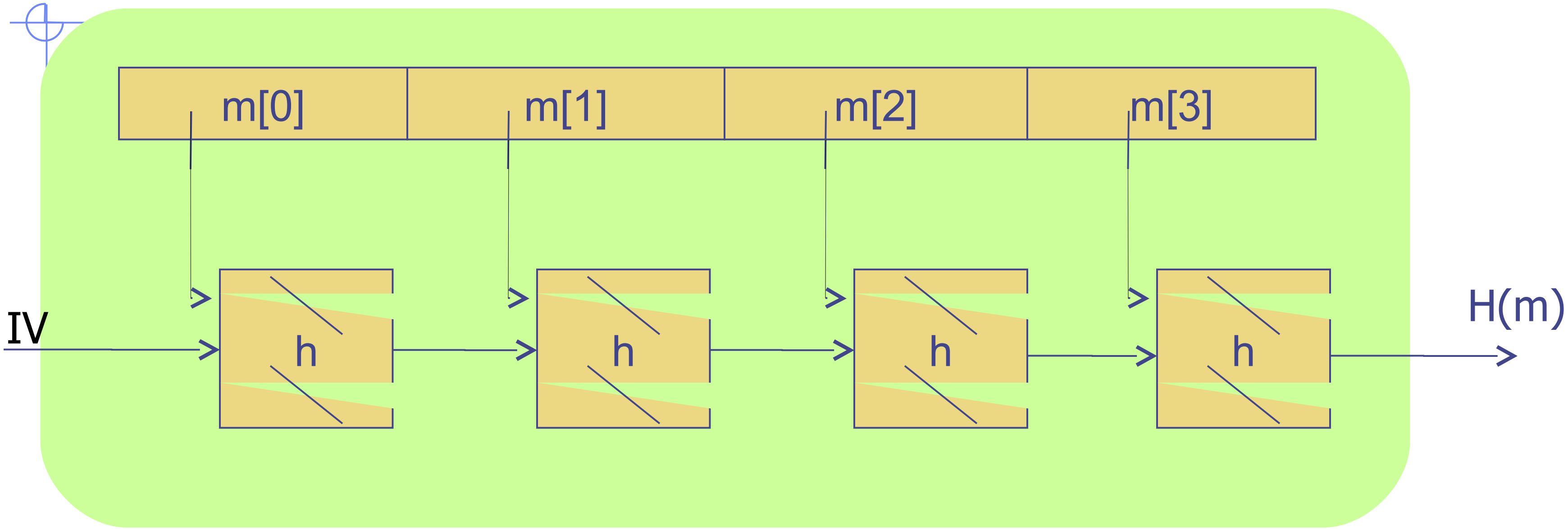
i) A popular system of checking message integrity, uses one-way hash f unctions to produce unique mac values.

ii) ipad and opad are used to modify the secret key. It is recommended to choose the values that would make both inputs to the hash functions look as dissimilar as possible

Iii) Using a secure hash function that doesn't produce the same outputs for different input data) guarantees the security of HMAC .



SHA-256: Merkle-Damgard



h(t, m[i]): compression function

Thm 1: if h is collision resistant then so is H

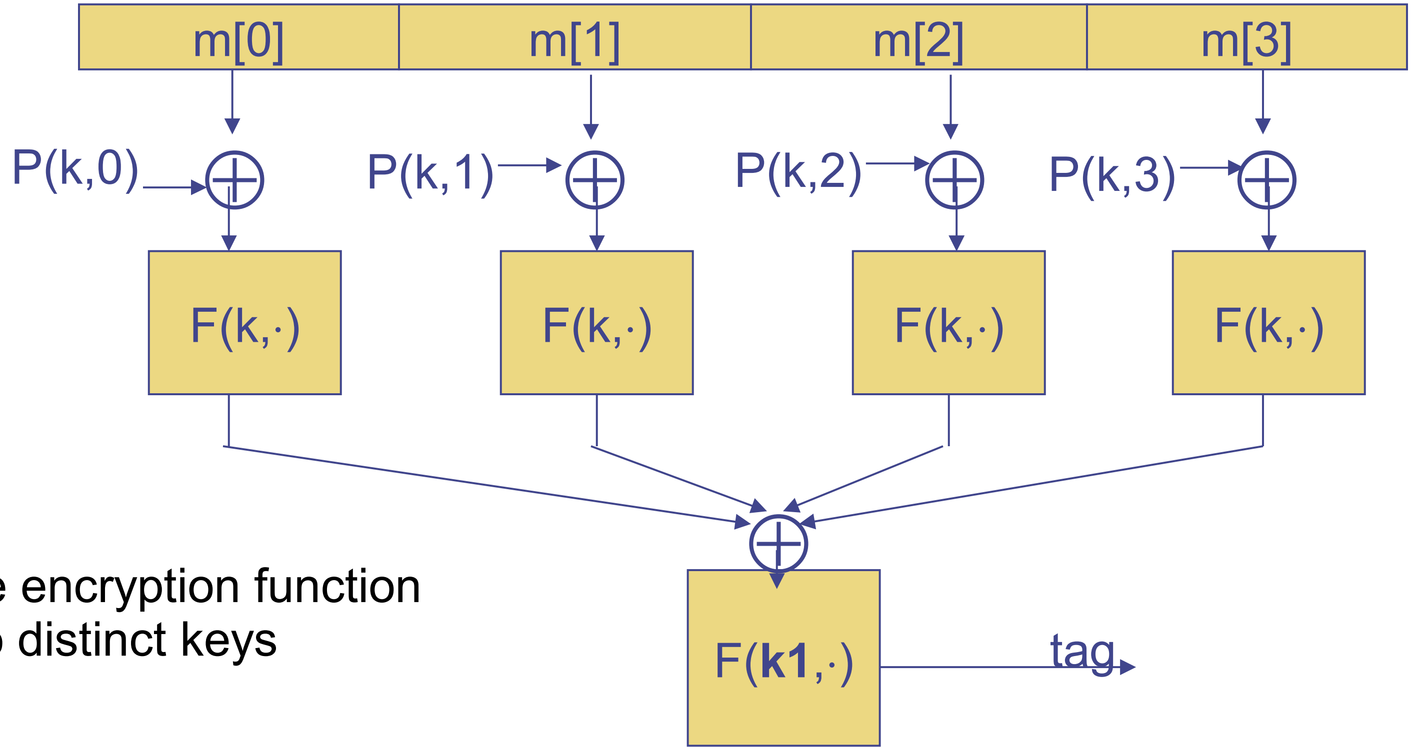
"Thm 2": if h is a PRF then HMAC is a PRF

PRF=pseudo random function

Construction 3: PMAC – parallel MAC

ECBC and HMAC are sequential.

PMAC:



One encryption function
Two distinct keys

- These MAC constructions are secure
  - No time to prove it

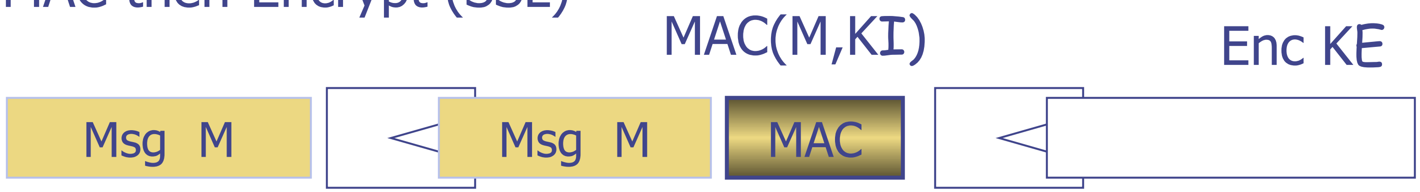
- Why the last encryption step in ECBC?
  - CBC (aka Raw-CBC) is not a secure MAC:
    - Given tag on a message m, attacker can deduce tag for some other message m’
    - How: good crypto exercise …

Authenticated Encryption: Encryption + MAC

Combining MAC and ENC (CCA)

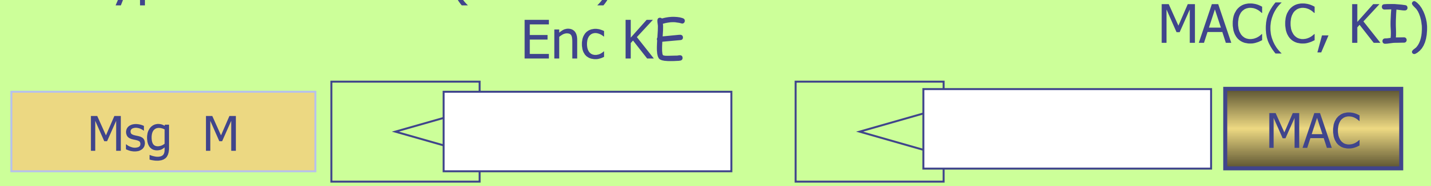
Encryption key KE   MAC key = KI

Option 1: MAC-then-Encrypt (SSL)

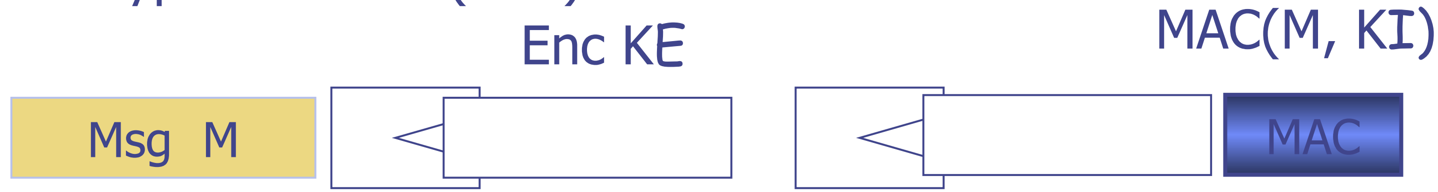


Option 2: Encrypt-then-MAC (IPsec)

Secure on general ground s



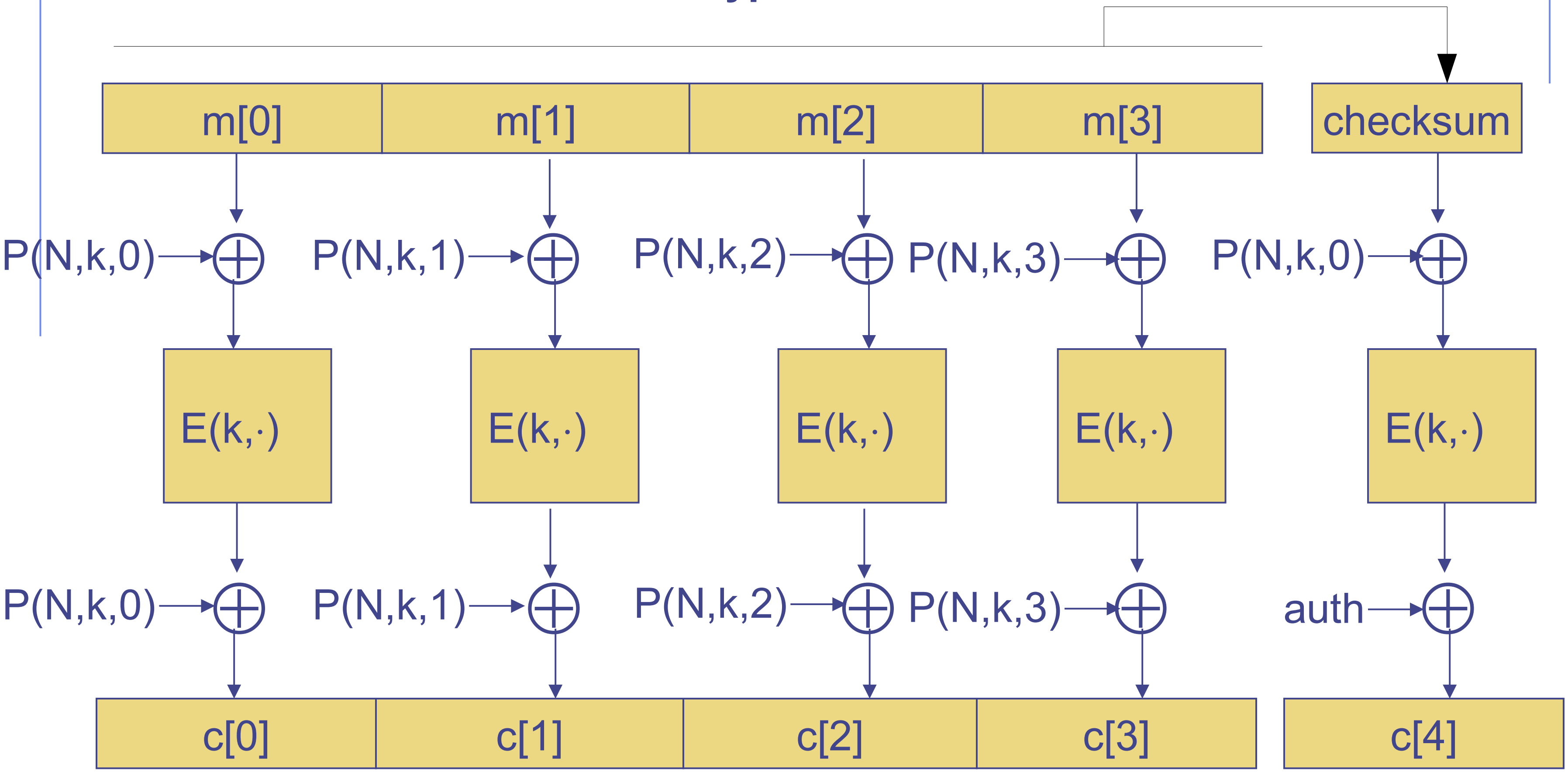
Option 3: Encrypt-and-MAC (SSH)



OCB

offset codebook mode

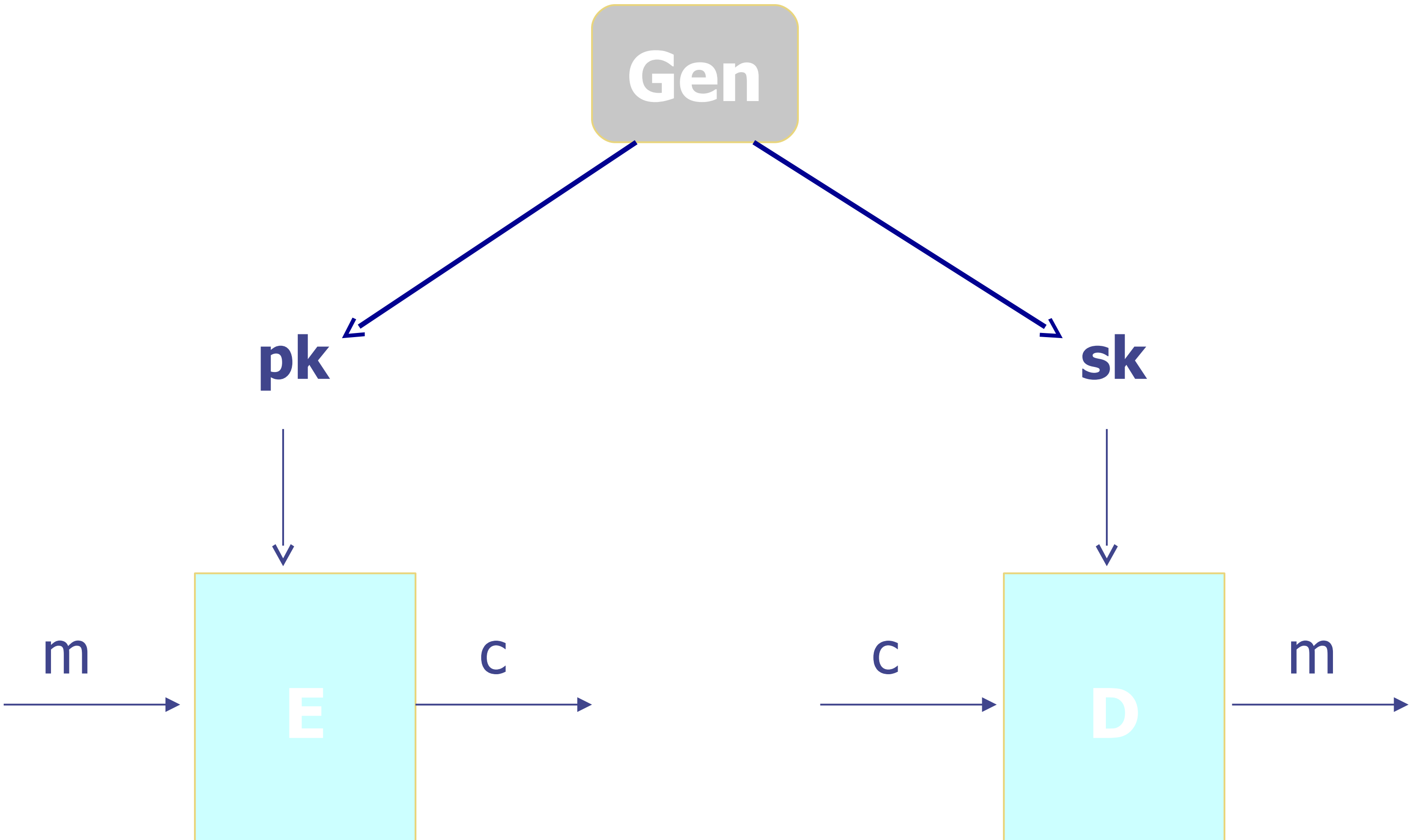
More efficient authenticated encryption



Rogaway, ...

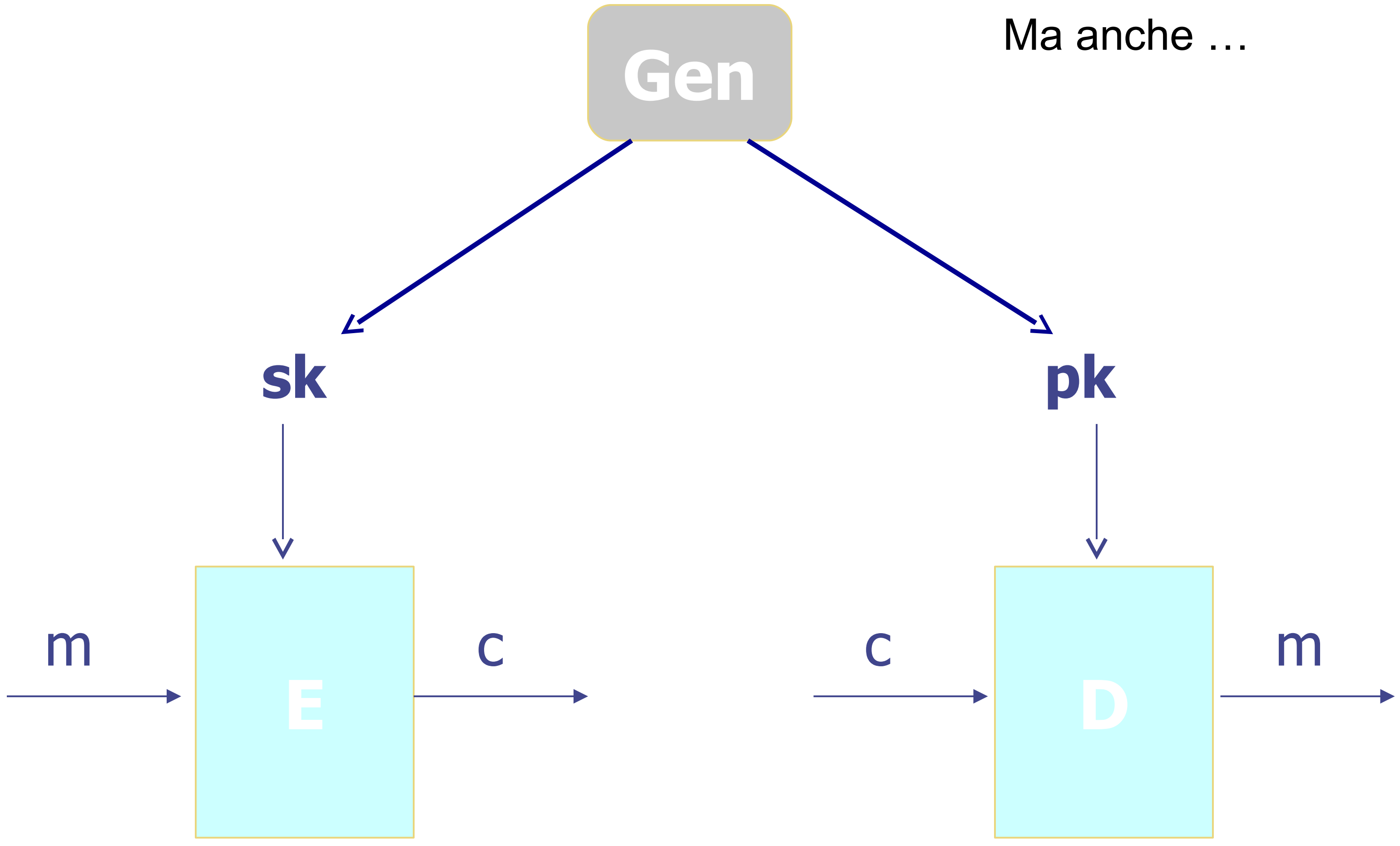
Public-key Cryptography

Public key encryption: (Gen, E, D)



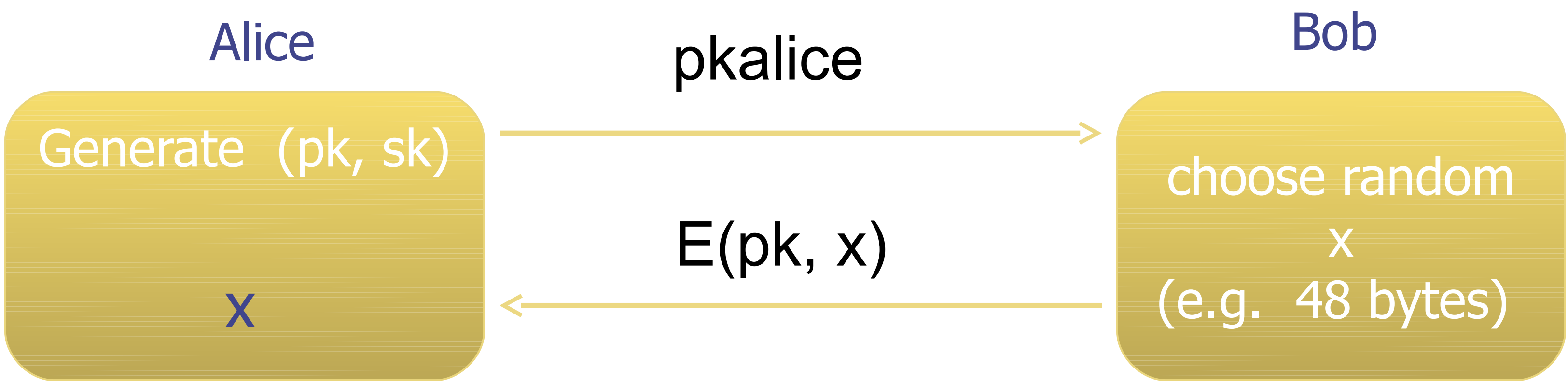
Public key encryption: (Gen, E, D)

Ma anche …



Applications

Session setup (for now, only eavesdropping security)



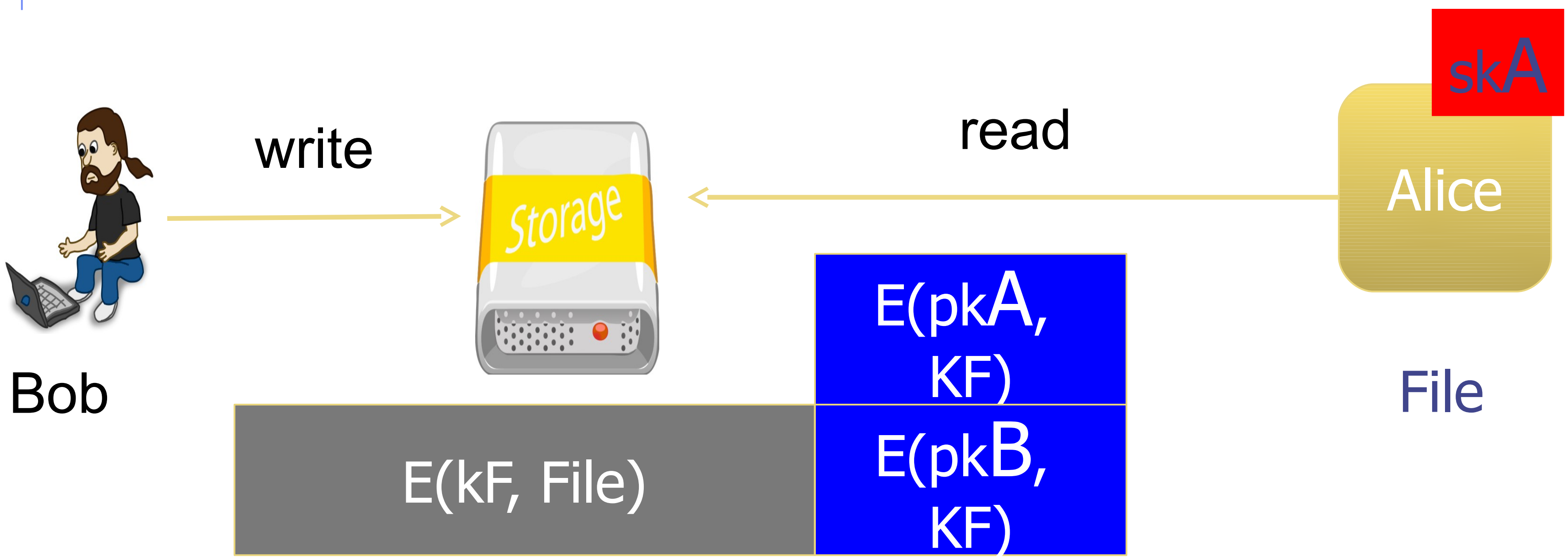
Non-interactive applications: (e.g. Email)

- Bob sends email to Alice encrypted using pkalice
- Note: Bob needs pkalice (public key management)

Applications

Encryption in non-interactive settings:

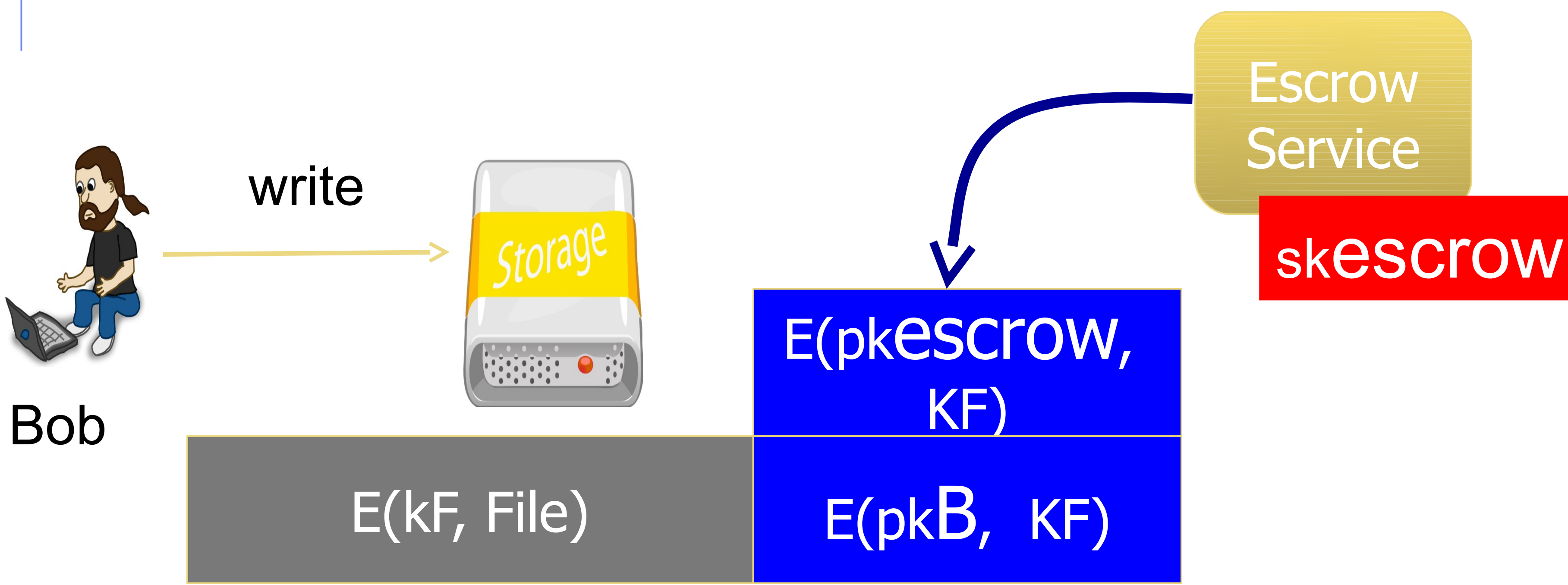
- Encrypted File Systems



Applications

Encryption in non-interactive settings:

- Key escrow: data recovery without Bob's key



Trapdoor functions (TDF)

A trapdoor func. X⟶Y is a triple of efficient algs. (G, F, F-1)

- G(): randomized alg. outputs key pair (pk, sk)
- F(pk,⋅): det. alg. that defines a func. X ⟶Y
- F-1(sk,⋅): Y ⟶ X that inverts F(pk,⋅)

Security: F(pk, ⋅) is one-way without sk

Public-key encryption from TDFs

- (G, F, F^{-1}): secure TDF $X \longrightarrow Y$
- (E_s, D_s) : symm. auth. encryption with keys in K
- H: $X \longrightarrow K$ a hash function

We construct a pub-key enc. system (G, E, D):

Key generation G: same as G for TDF

Public-key encryption from TDFs

- (G, F, F-1): secure TDF $X \longrightarrow Y$
- (Es, Ds) : symm. auth. encryption with keys in K
- H: $X \longrightarrow K$ a hash function

E(pk, m) :

$x \leftarrow X$, $y \leftarrow F(pk, x)$

$k \leftarrow H(x)$, $c \leftarrow Es(k, m)$

output (y, c)

D(sk, (y,c)) :

$x \leftarrow F\text{-}1(sk, y)$,

$k \leftarrow H(x)$, $m \leftarrow Ds(k, c)$

output m

Digital Signatures

- Public-key encryption
  - Alice publishes encryption key
  - Anyone can send encrypted message
  - Only Alice can decrypt messages with this key
- Digital signature scheme
  - Alice publishes key for verifying signatures
  - Anyone can check a message signed by Alice
  - Only Alice can send signed messages

Digital Signatures from TDPs

- (G, F, F-1): secure TDP $X \longrightarrow X$
- H: $M \longrightarrow X$ a hash function

Sign(sk, m∈X) :

output

sig = F-1(sk, H(m))

Verify(pk, m, sig) :

output

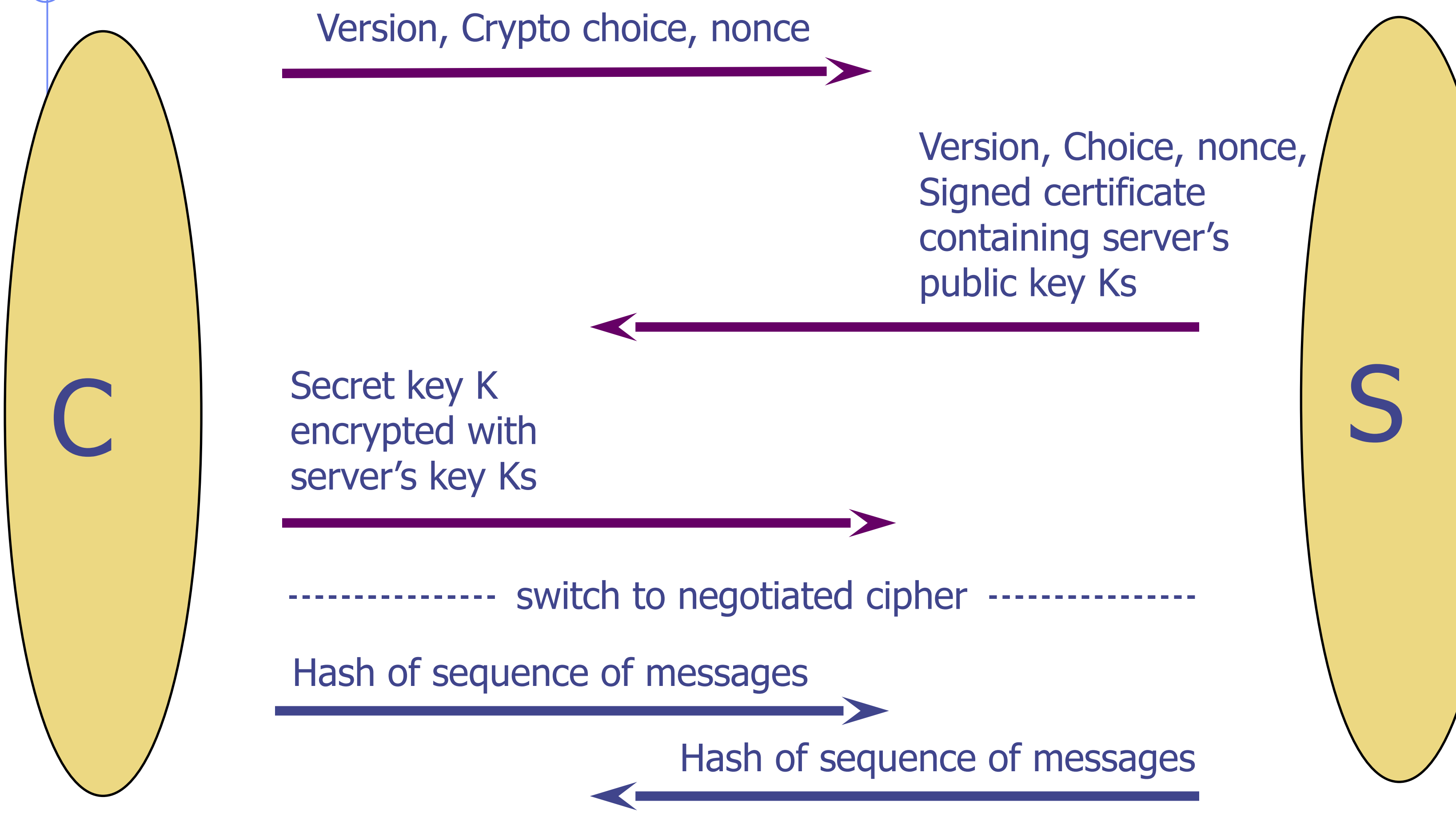
$$\begin{cases} 1 & \text{if} \quad H(m) = F(pk, sig) \\ 0 & \text{otherwise} \end{cases}$$

Security: existential unforgeability under a chosen message attack in the random oracle model

Public-Key Infrastructure (PKI)

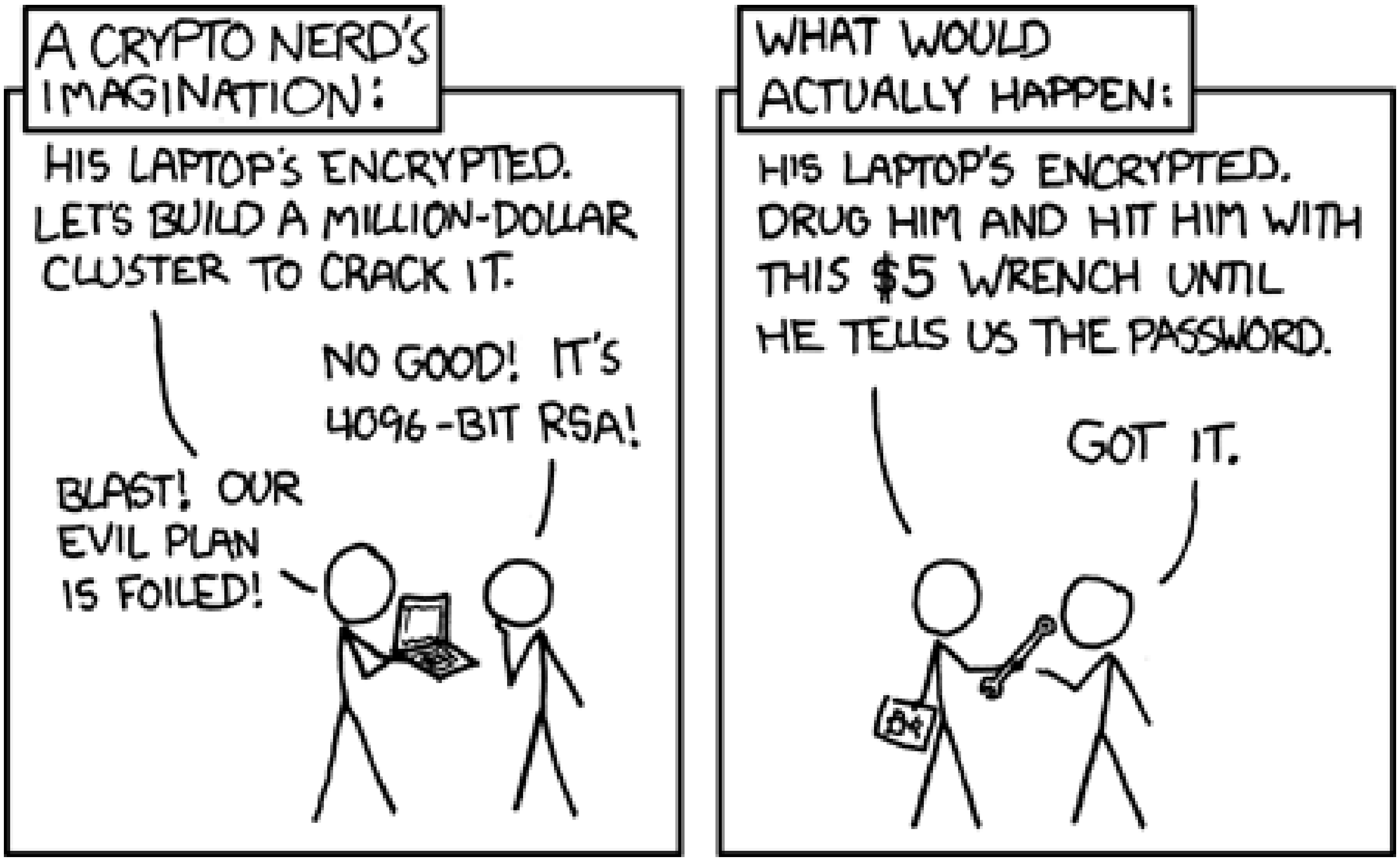
- Anyone can send Bob a secret message
  - Provided they know Bob's public key
- How do we know a key belongs to Bob?
  - If imposter substitutes another key, can read Bob's mail
- One solution: PKI
  - Trusted root Certificate Authority (e.g. Symantec)
    - Everyone must know the verification key of root CA
    - Check your browser; there are hundreds!!
  - Root authority signs intermediate CA
  - Results in a certificate chain

Back to SSL/TLS



Limitations of cryptography

- Most security problems are not crypto problems
  - This is good
    - Cryptography works!
  - This is bad
    - People make other mistakes; crypto doesn’t solve them
- Misuse of cryptography is fatal for security
  - WEP – ineffective, highly embarrassing for industry
  - Occasional unexpected attacks on systems subjected to serious review


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.