Cryptography Overview

Dan Boneh
(Modified by Vijay Ganesh)

Cryptography: Today's Lecture

- An introduction to cryptography
  - Basic definitions
  - Uses of cryptography
- SSL/TLS
- Symmetric-key encryption
  - One-time pad, stream and block ciphers
- Public-key cryptography
- Digital signatures
- Limitations of cryptography

Cryptography

- Is
  - A tremendous tool
  - The basis for many security mechanisms
- Is not
  - The solution to all security problems
  - Reliable if implemented properly
  - Reliable if used properly

Auguste Kerckhoffs

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

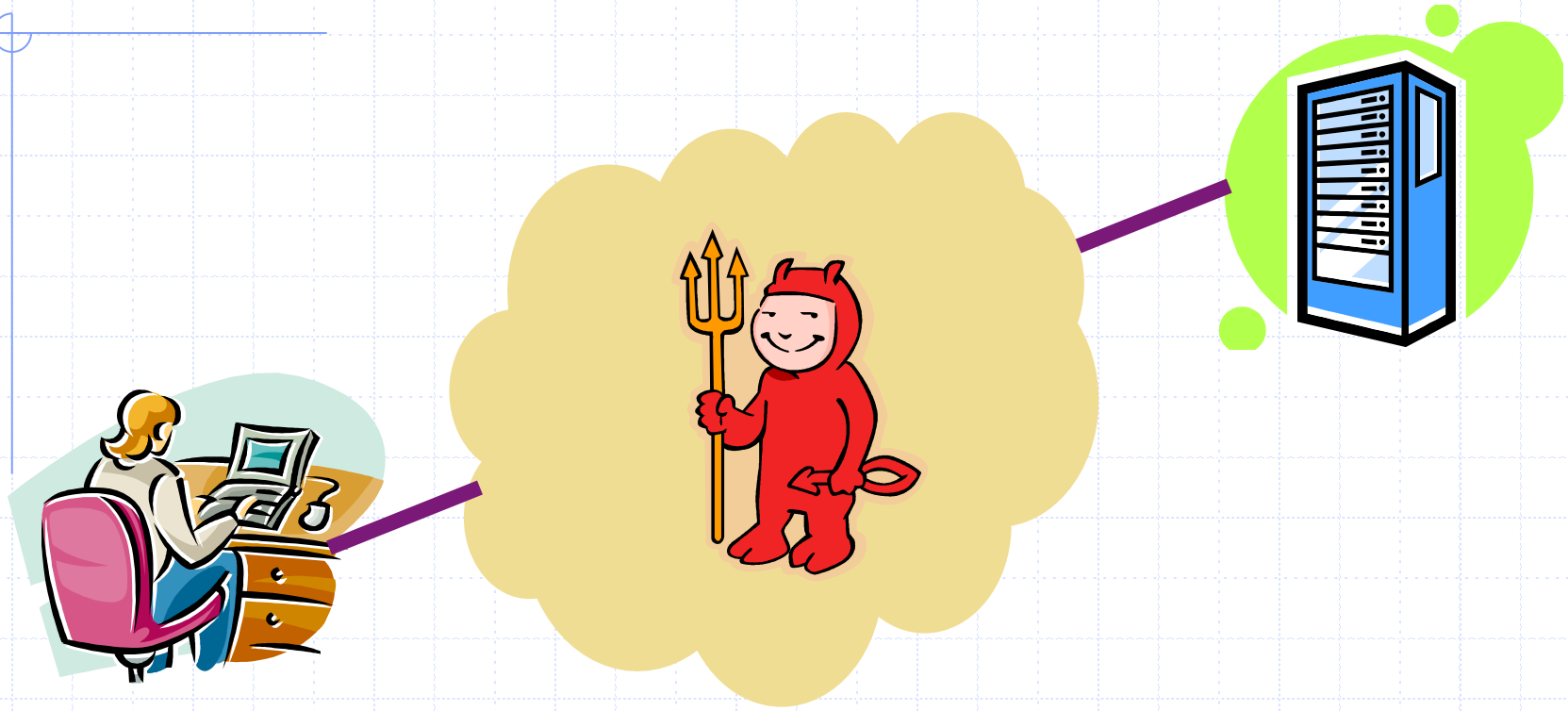
baptised as **Jean-Guillaume-Hubert-Victor-François-Alexandre-Auguste Kerckhoffs von Nieuwenhof**

Crypto threat model

- Assume attacker knows the cryptosystem
- Attacker does not know random numbers
  - Generated as systems run, not in advance
- Carefully define attacker model

- Easy lessons
  - Use good random number generators
  - No harm in public review of cryptography
    - This prevents silly and not-so-silly mistakes
    - Benefit from community of experts

Example: network transactions



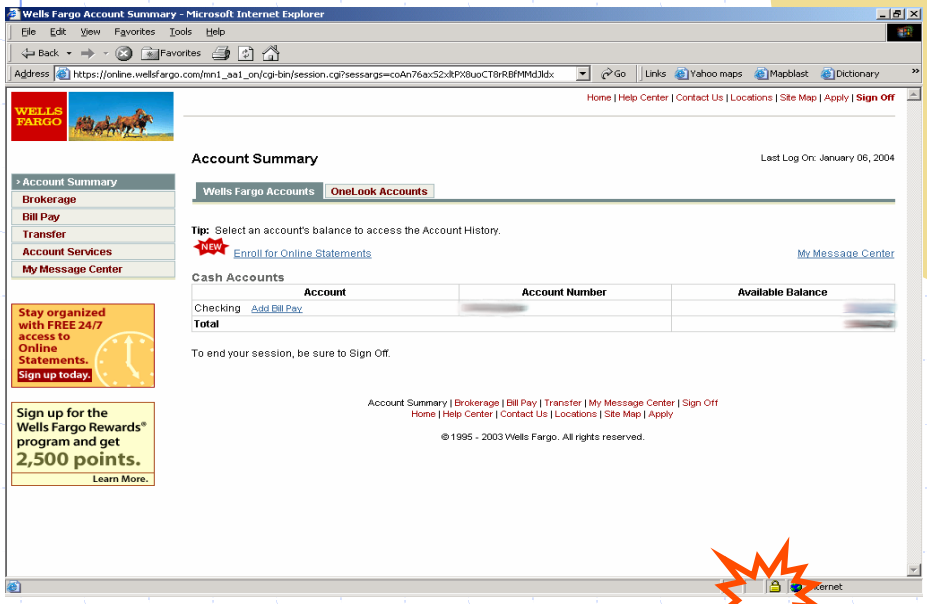
Assume attackers can control the network

- We will talk about how they do this in a few weeks
- Attackers can intercept packets, tamper with or suppress them, and inject arbitrary packets

Goal 1:secure communication

Step 1: Session setup to exchange key

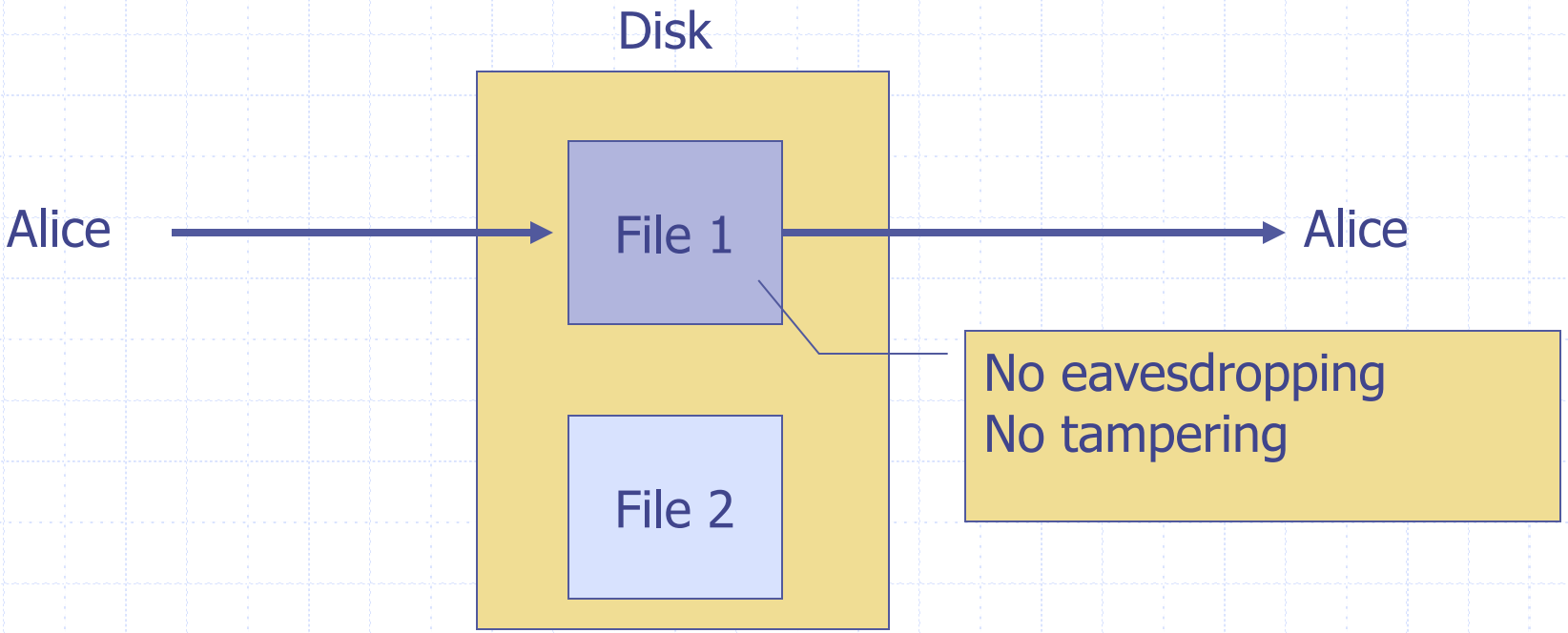
Step 2: encrypt data



Secure Sockets Layer / TLS

- Standard for Internet security
  - Originally designed by Netscape
  - Goal: "... provide privacy and reliability between two communicating applications"
- Two main parts
  - Handshake Protocol
    - Establish shared secret key using public-key cryptography
    - Signed certificates for authentication
  - Record Layer
    - Transmit data using negotiated key, encryption function

Goal 2: Protected files

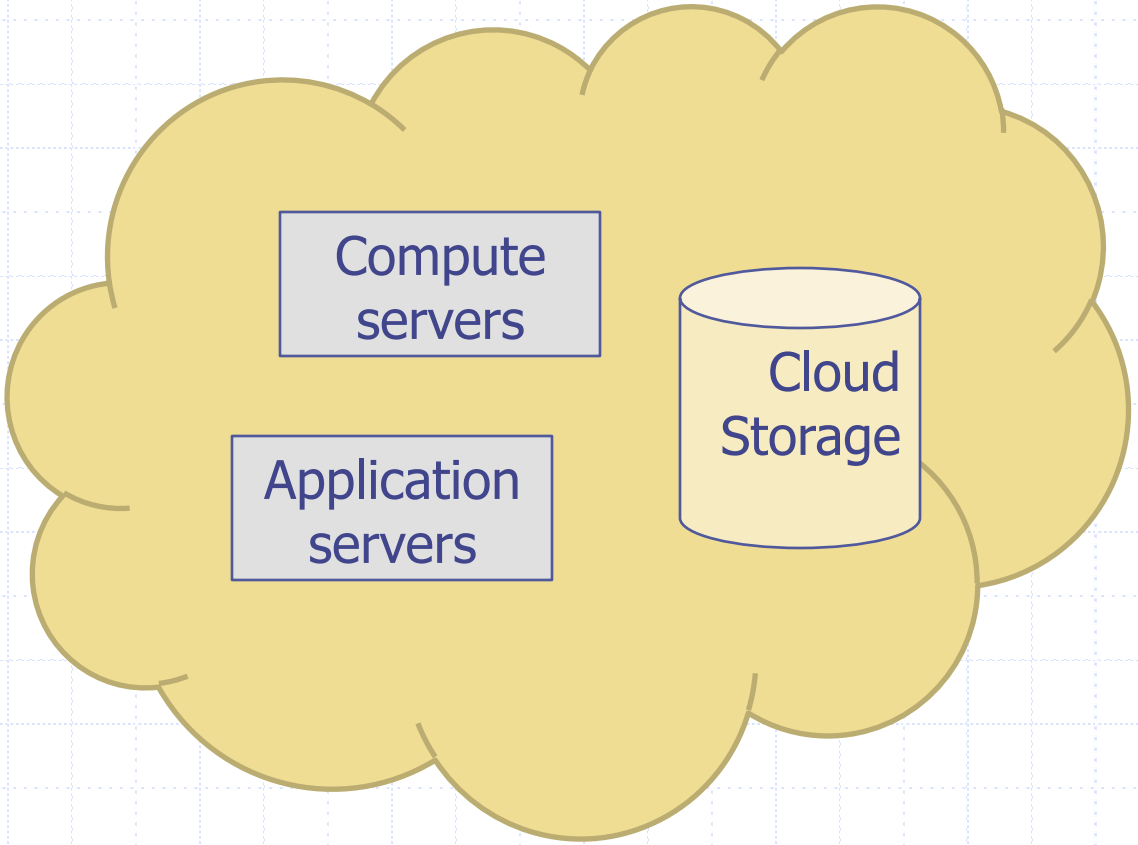


Analogous to secure communication:

Alice today sends a message to Alice tomorrow

Goal 3: Secure cloud computing?

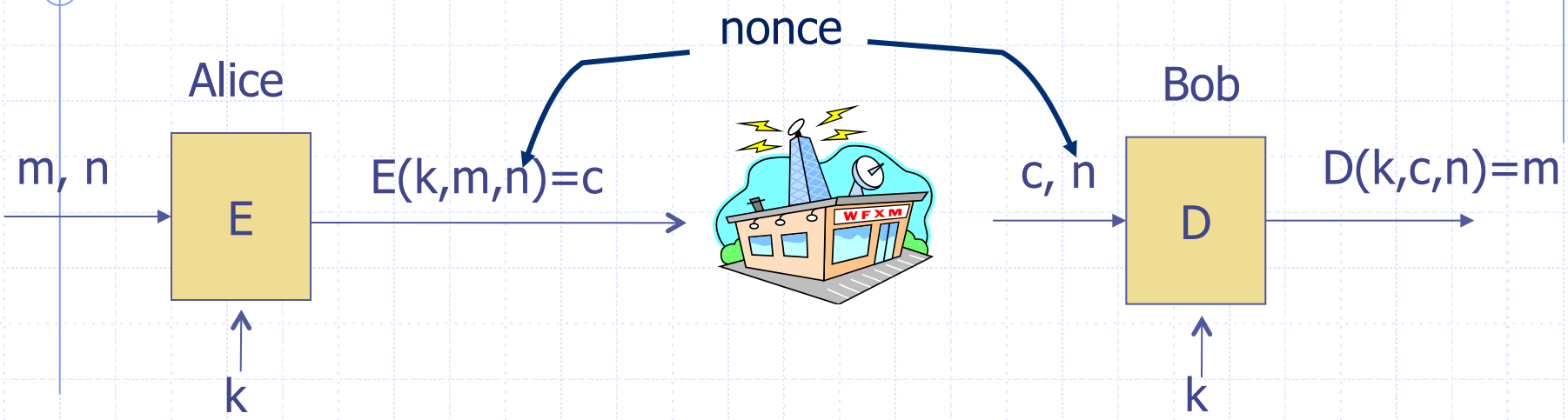
?



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 (Initial Vector)

Encryption algorithm known publicly

- 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)

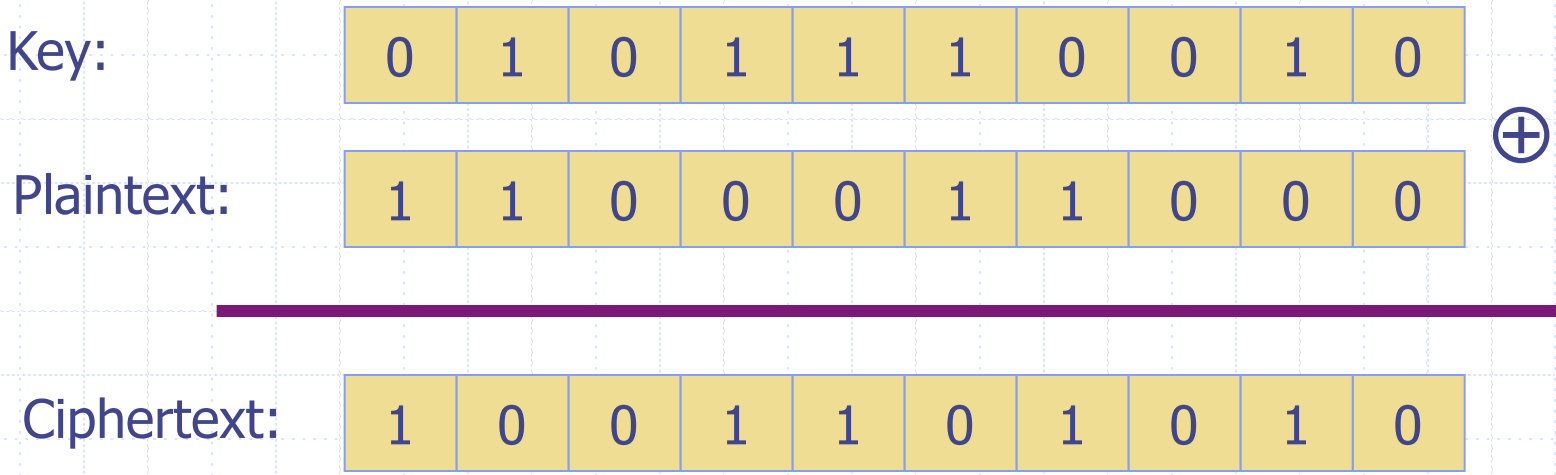
Multi use key: (many time key)

- 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)

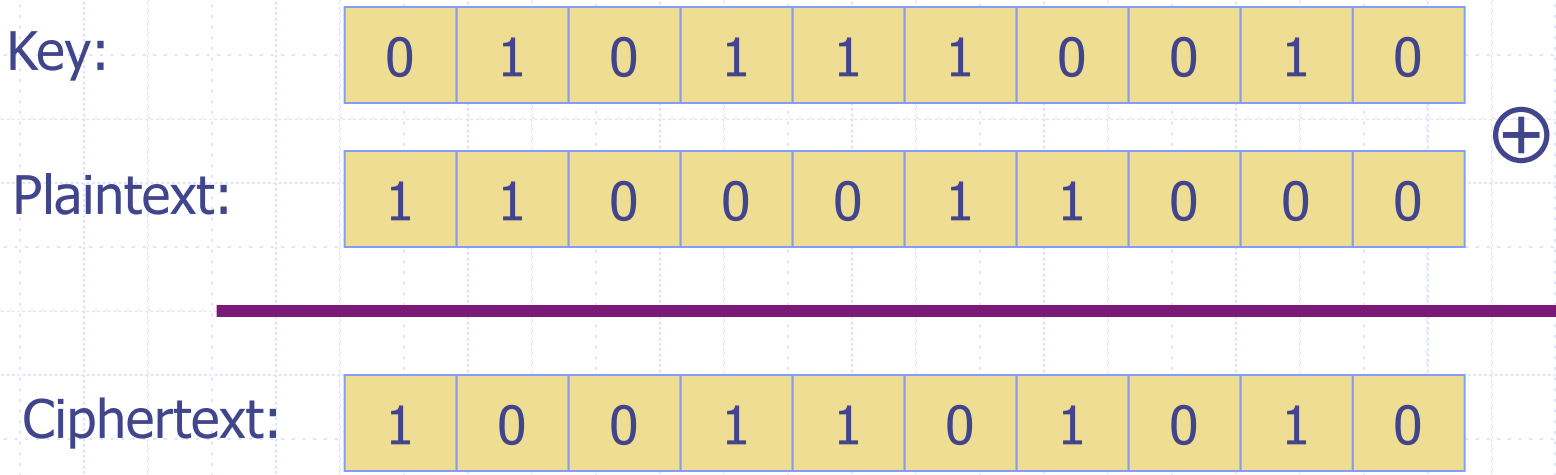


- Shannon '49:
  - OTP is "secure" against ciphertext-only attacks (COA)
  - Information-theoretically secure

One Time Pad: Perfect Security

(single use key)

- Vernam (1917)

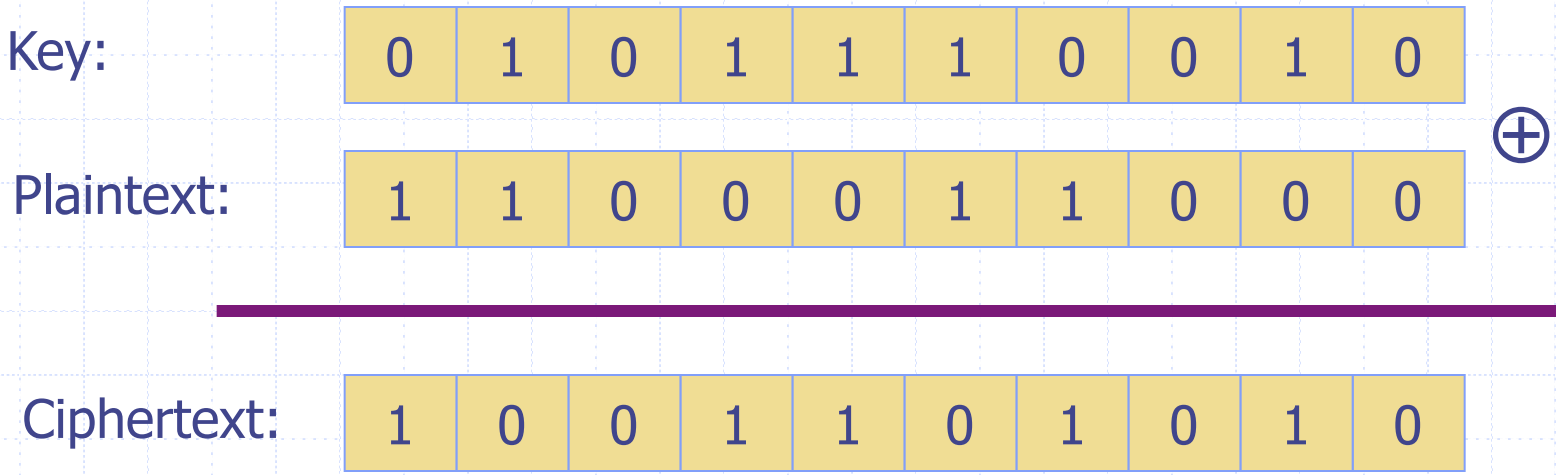


- Secure against adversary with unlimited computational power
- H(M) = H(M|C)
- Cipher-text doesn't give attacker any additional power

Problems with One-time Pad

(single use key)

- Vernam (1917)

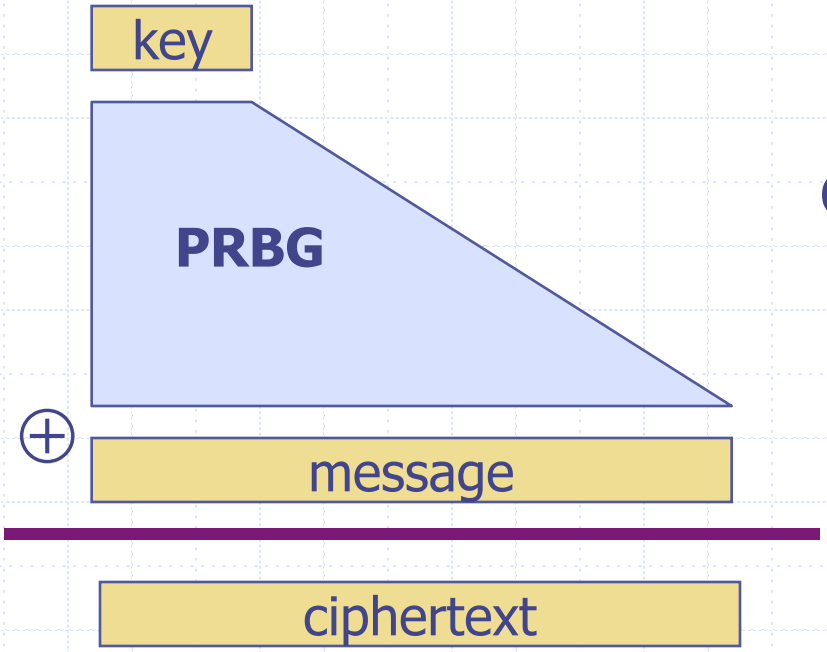


- Not secure if key is reused. Key must be truly random every use
  - Why? Entropy in plaintext
- Key as long as the message
- No Authentication

Stream ciphers (single use key)

Problem: OTP key is as long the message

Solution: Pseudo random key -- stream ciphers



$c \leftarrow PRBG(k) \oplus m$

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

Dangers in using stream ciphers

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

$$\begin{cases} C_1 \leftarrow m_1 \oplus \text{PRBG}(k) \\ C_2 \leftarrow m_2 \oplus \text{PRBG}(k) \end{cases}$$

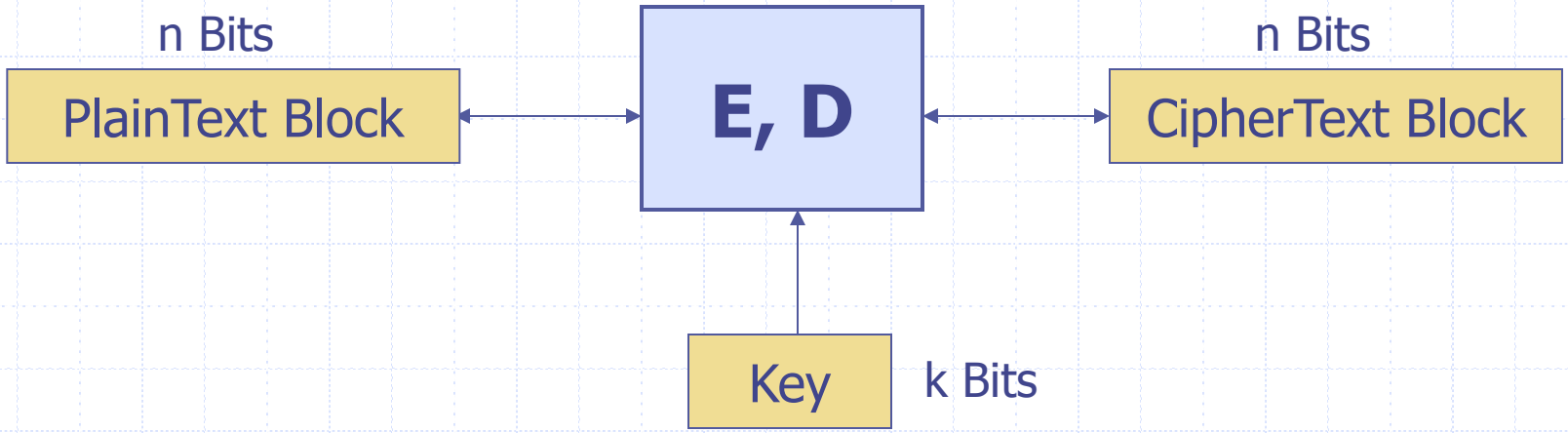
Eavesdropper does:

$$C_1 \oplus C_2 \rightarrow m_1 \oplus m_2$$

Enough redundant information in English that:

$$m_1 \oplus m_2 \rightarrow m_1 , m_2$$

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

Initialization Vector (IV) handled as part of PlainText (PT) block

Building a block cipher

Input: (m, k)

Repeat simple "mixing" operation several times

- DES: Repeat 16 times:

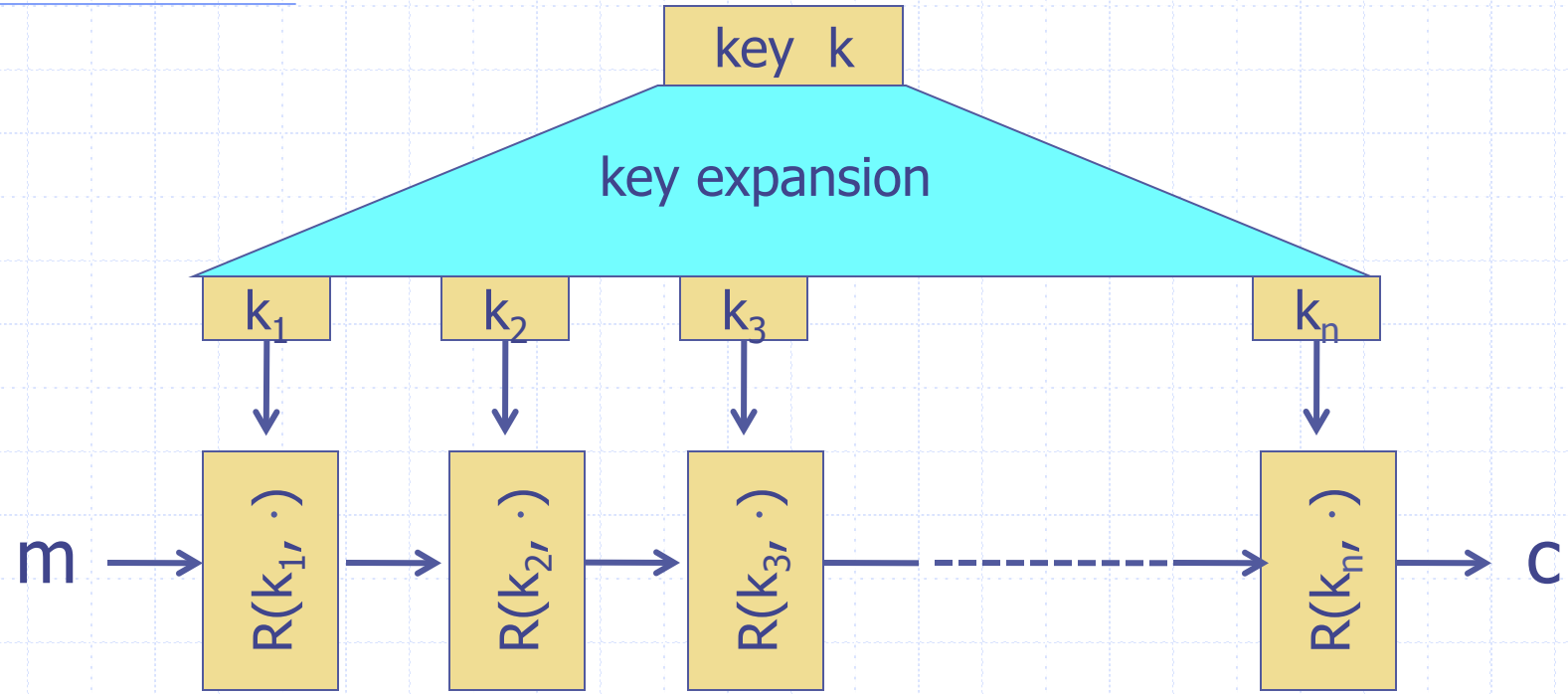
$$\begin{cases} m_L \leftarrow m_R \\ m_R \leftarrow m_L \oplus F(k, m_R) \end{cases}$$

- AES-128: Mixing step repeated 10 times

Difficult to design: must resist subtle attacks

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

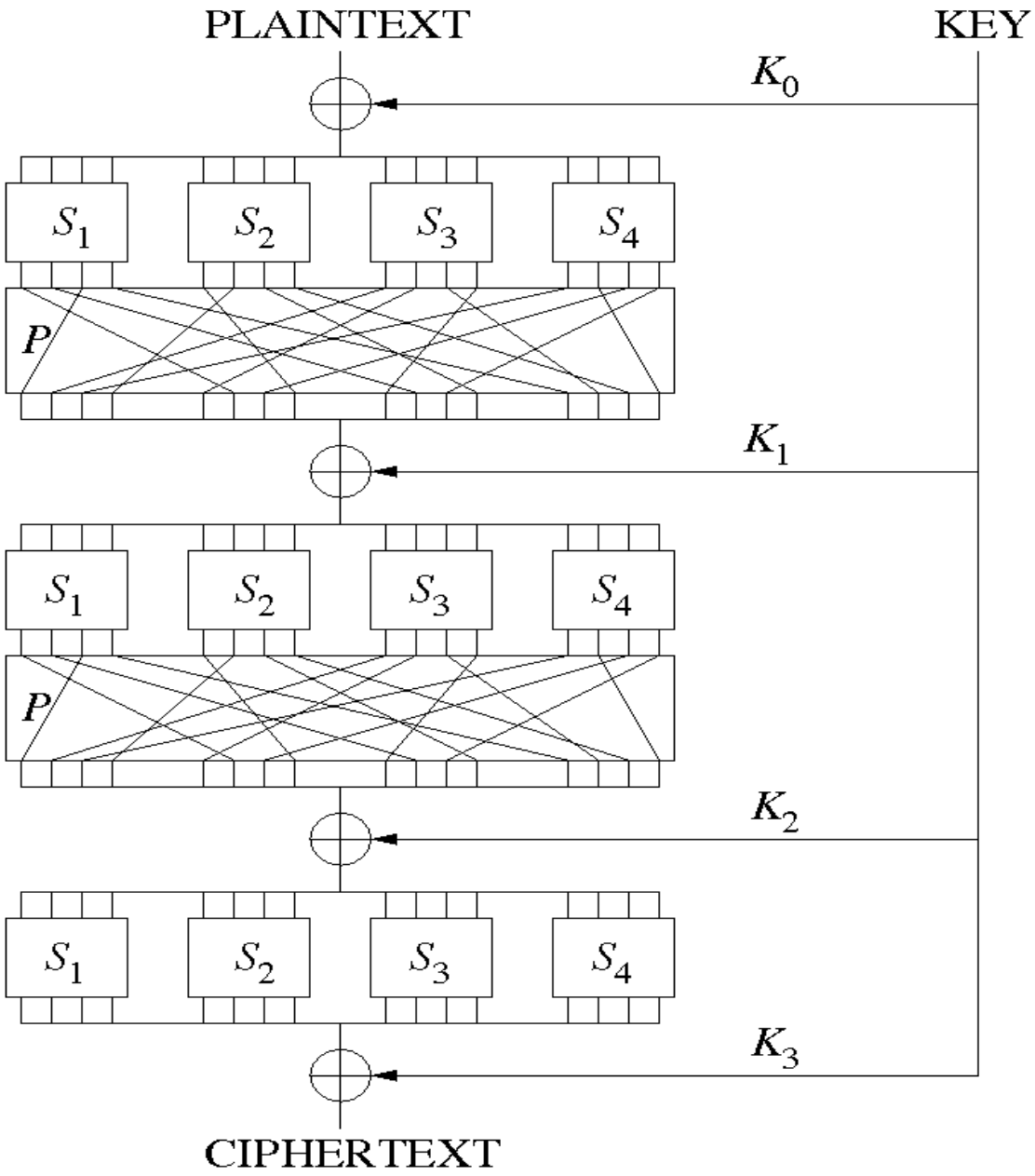
Block Ciphers Built by Iteration



R(k,m): round function

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

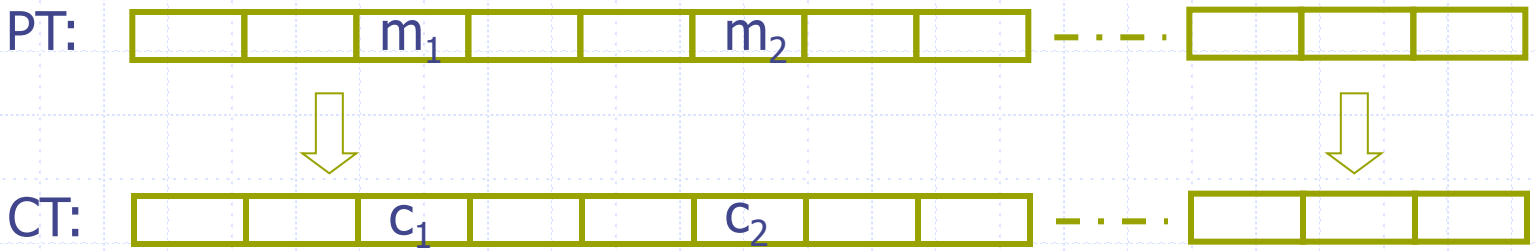
Substitution Permutation Network



Source: Wikipedia

Incorrect use of block ciphers

- Electronic Code Book (ECB):

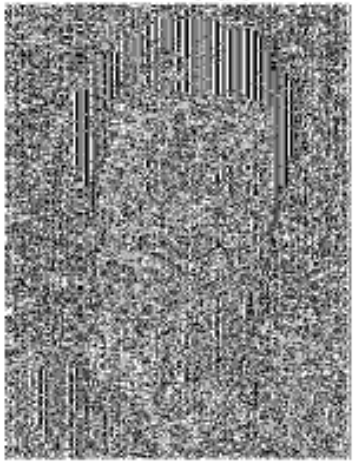


- Problem:
  - if $m_1 = m_2$ then $c_1 = c_2$

In pictures

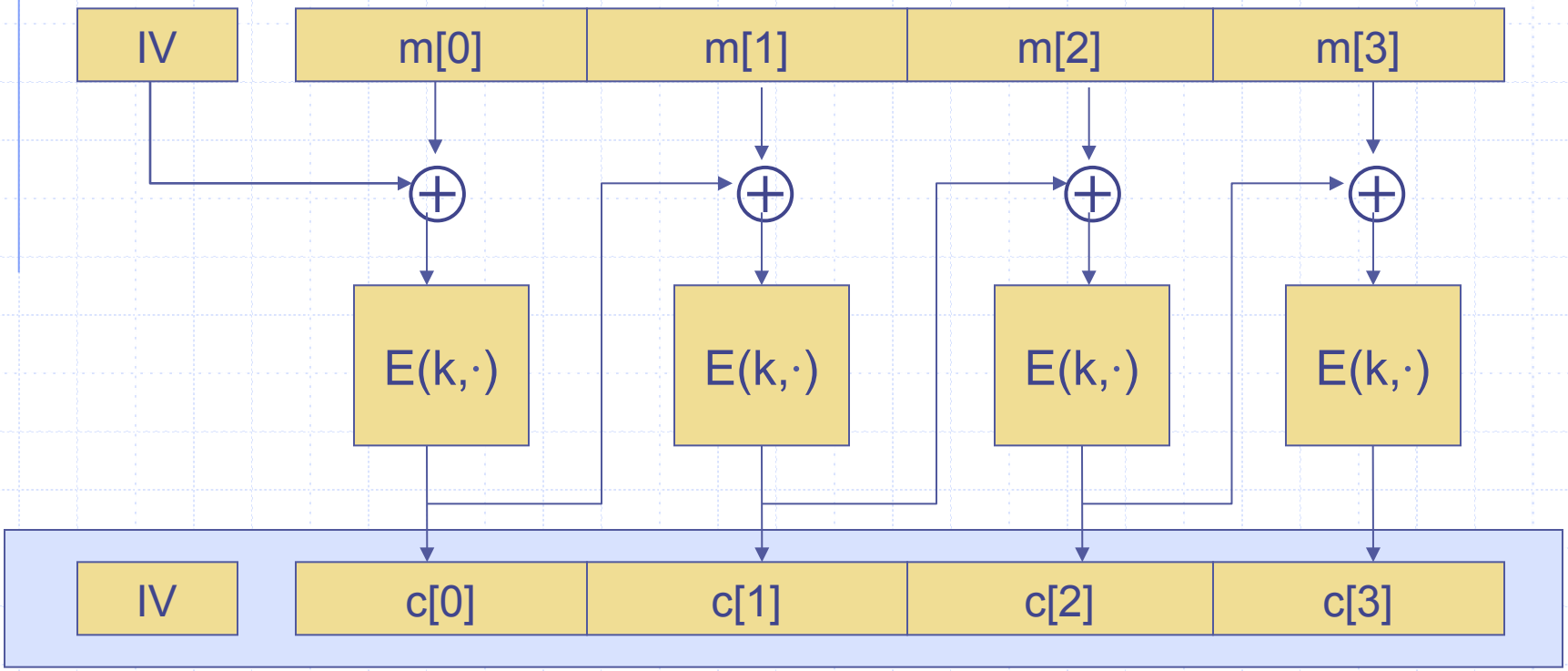
An example plaintext

Encrypted with AES in ECB mode



Correct use of block ciphers I: CBC mode

E a secure PRP. Cipher Block Chaining with Initiation Vector (IV):



Q: how to do decryption?

Use cases: how to choose an IV

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

Multi use key: (CPA Security)

Best: use a fresh *random* IV for every message (IV ← X)

Can use *unique* IV (e.g counter) [Bitlocker]

but then first step in CBC must be $IV' \leftarrow E(k_1, IV)$

benefit: may save transmitting IV with ciphertext

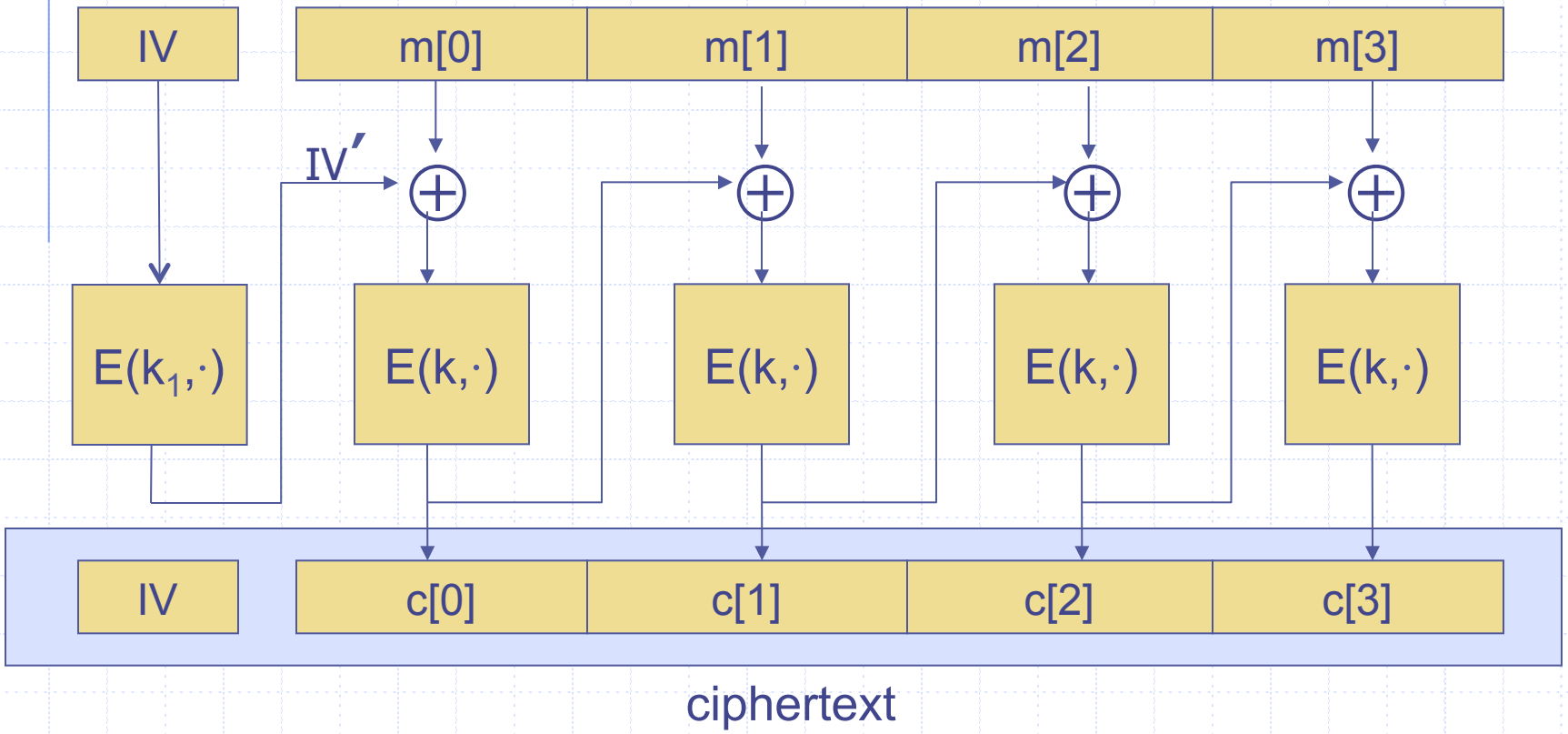
Multi-use key, but unique messages

SIV: eliminate IV by setting IV ← F(k', PT)

F: secure PRF with key k'

CBC with Unique IVs

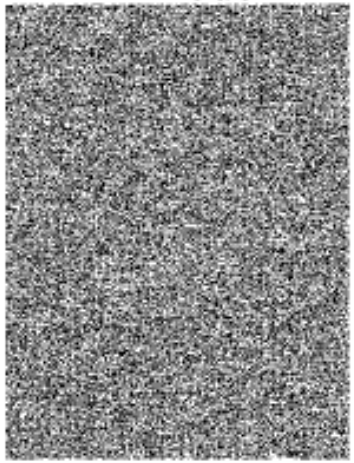
unique IV means: (k,IV) pair is used for only one message
may be predictable so use $E(k_1,\cdot)$ as PRF



In pictures

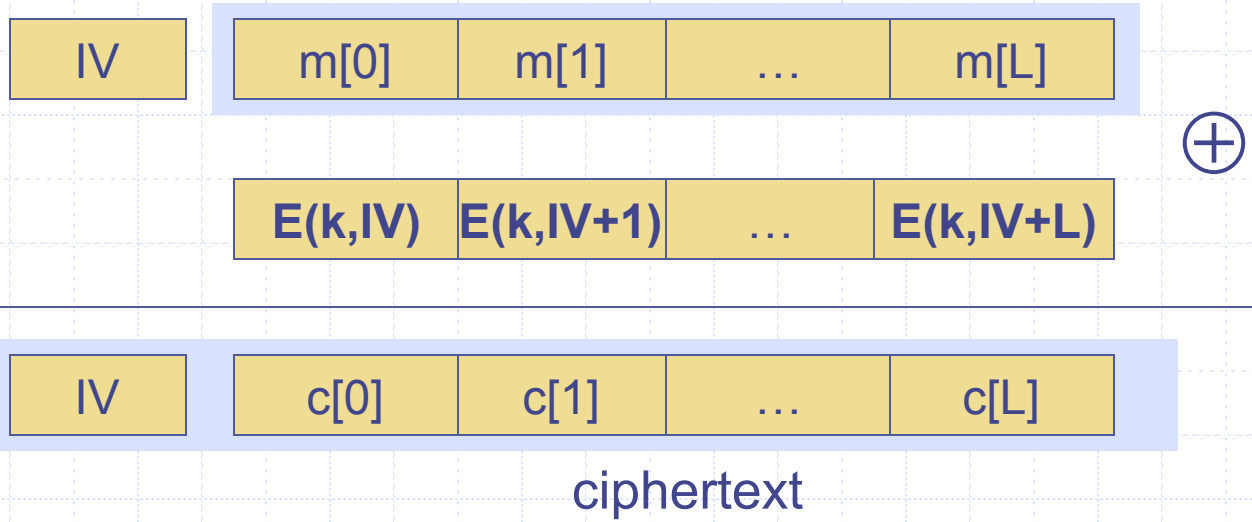
An example plaintext

Encrypted with AES in CBC mode



Correct use of block ciphers II: CTR mode

Counter mode with a random IV: (parallel encryption)



- Why are these modes secure? not today.

Performance: Crypto++ 5.2.1 [Wei Dai]

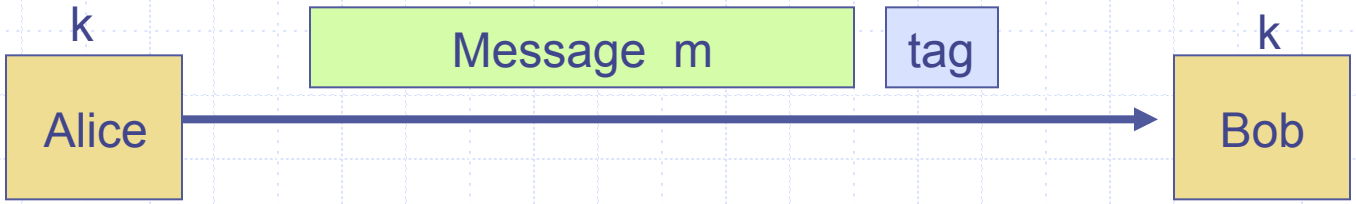
Pentium 4, 2.1 GHz (on Windows XP SP1, Visual C++ 2003)

Cipher	Block/key size	Speed (MB/sec)
RC4		113
SEAL		293
3DES	64/168	9
AES	128/128	61
IDEA	64/128	19
SHACAL-2	512/128	20

Data integrity

Message Integrity: MACs

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



Generate tag:
tag ← S(k, m)

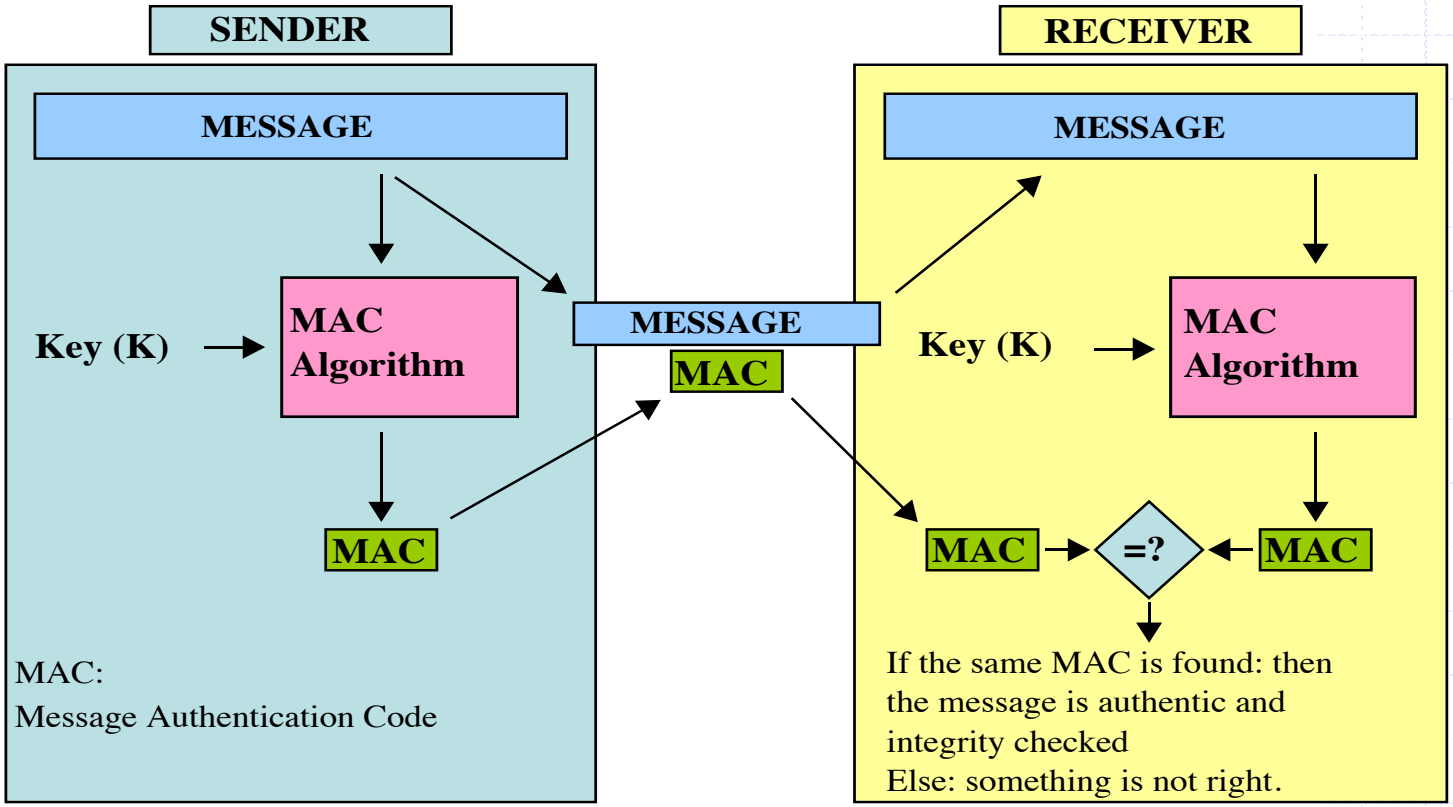
Verify tag: ?
V(k, m, tag) = `yes'

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

MACs

MAC diagram

http://upload.wikimedia.org/wikipedia/commons/0/08/MAC.svg



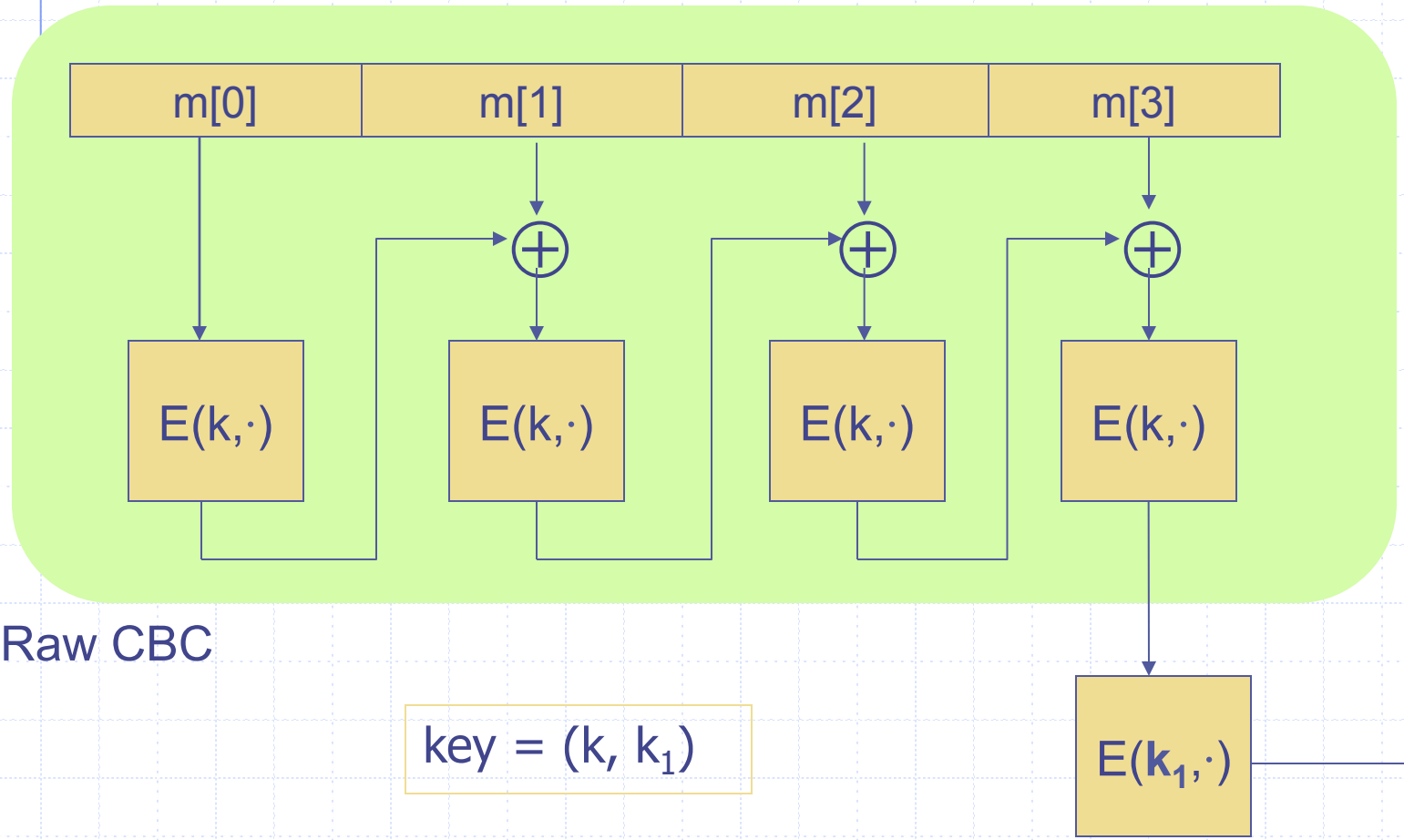
Secure MACs

- Attacker information: chosen message attack
  - for $m_1, m_2, \ldots, m_q$ attacker is given $t_i \leftarrow S(k, m_i)$
- Attacker's goal: existential forgery.
  - produce some **new** valid message/tag pair (m,t).

$$(m,t) \notin \{ (m_1,t_1) , \ldots , (m_q,t_q) \}$$

- A secure PRF 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



Raw CBC

key = (k, k_1)

MAC Construction: HMAC (Hash-MAC)

Most widely used MAC on the Internet.

H: hash function.

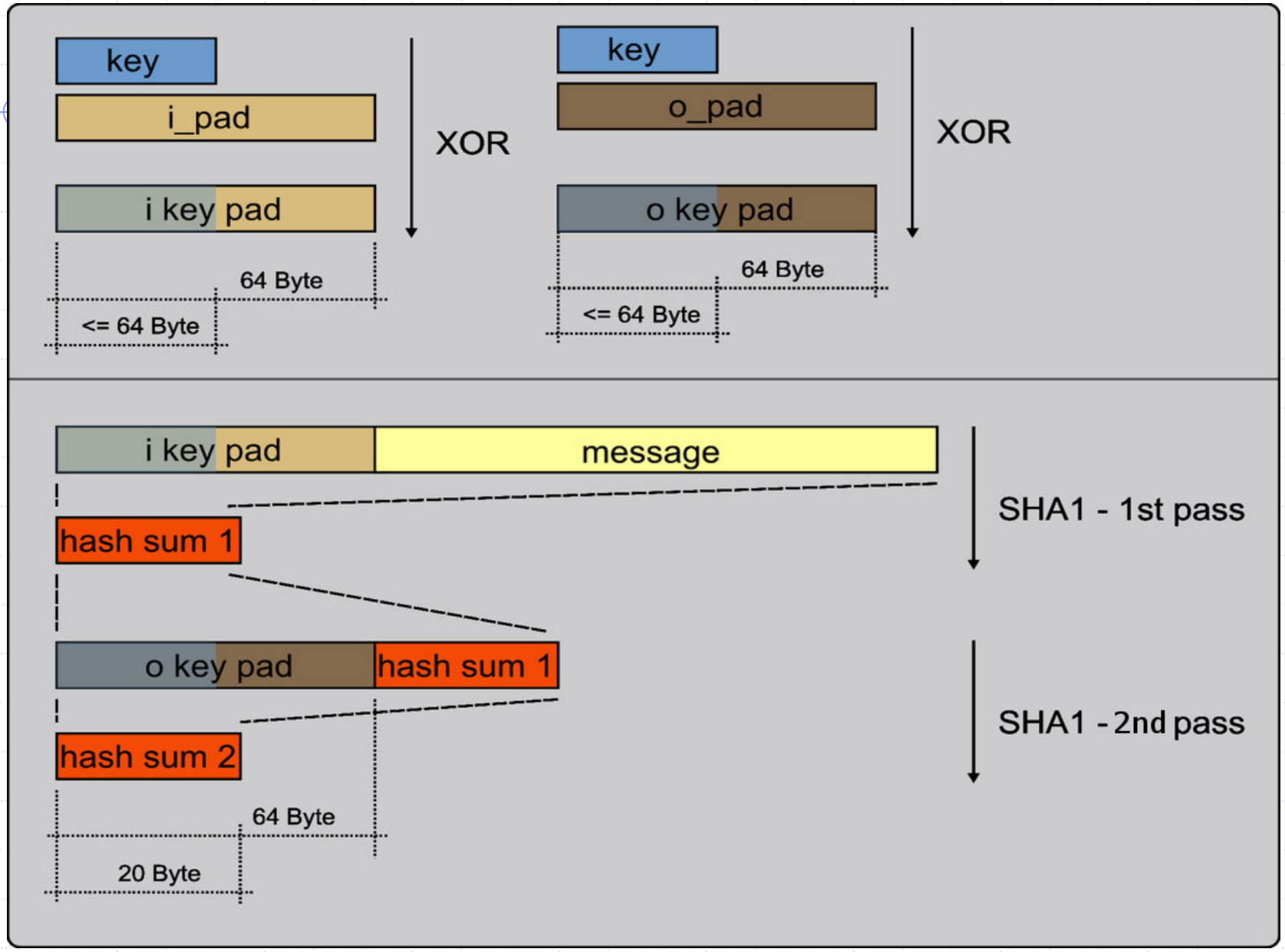
example: SHA-256 ; output is 256 bits

Building a MAC out of a hash function:

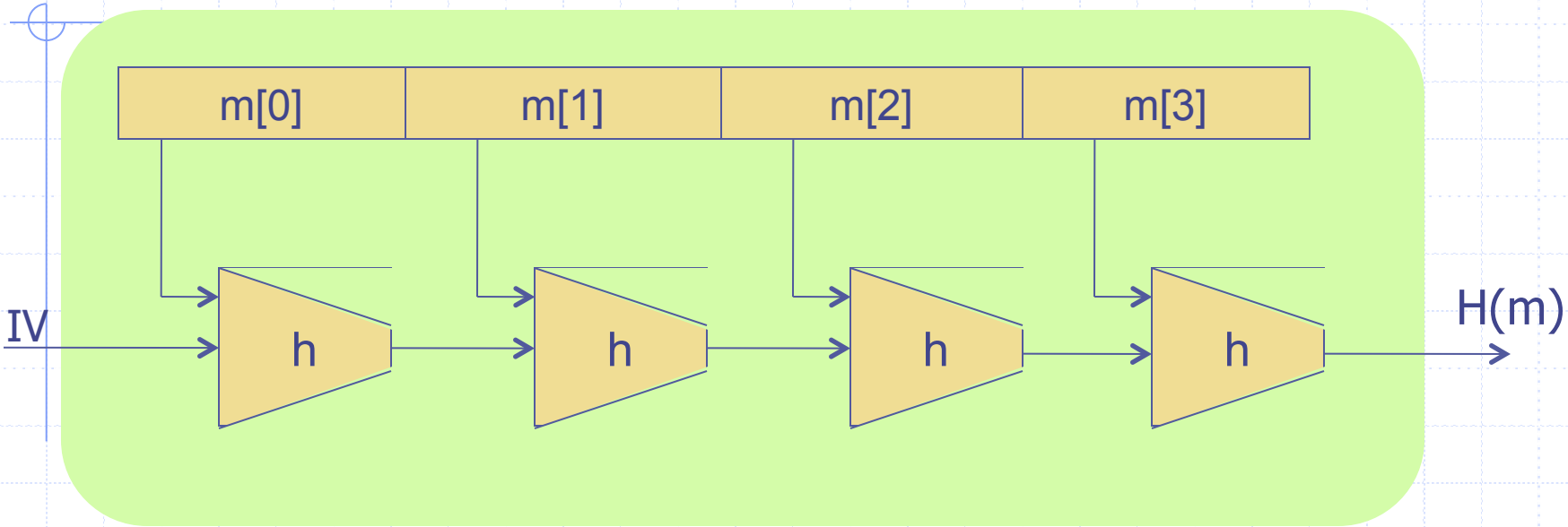
Standardized method: HMAC

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

Hash MAC (HMAC-SHA256)



SHA-256: Merkle-Damgard



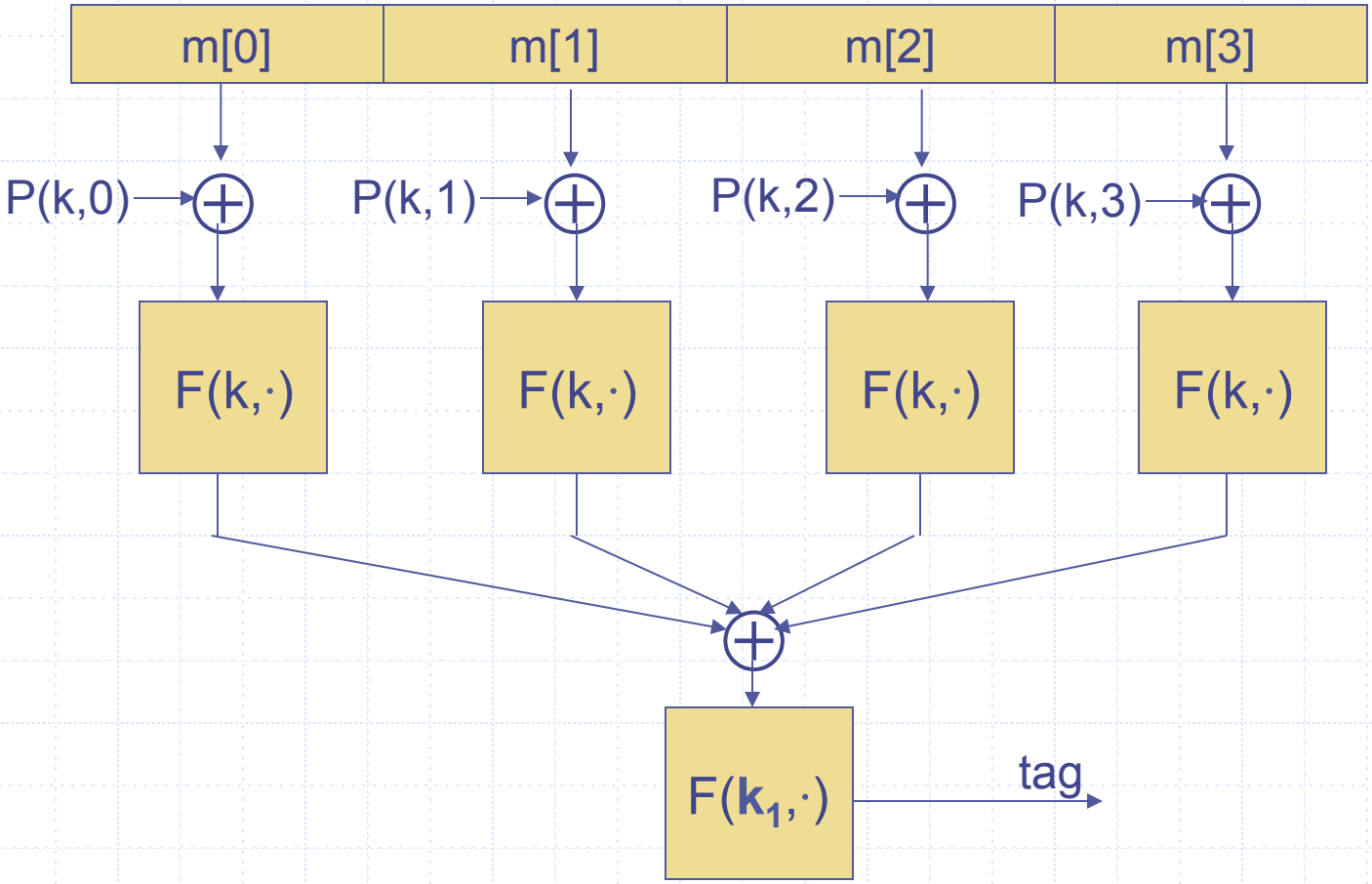
h(t, m[i]): compression function

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

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

Construction 3: PMAC – parallel MAC

ECBC and HMAC are sequential. PMAC:



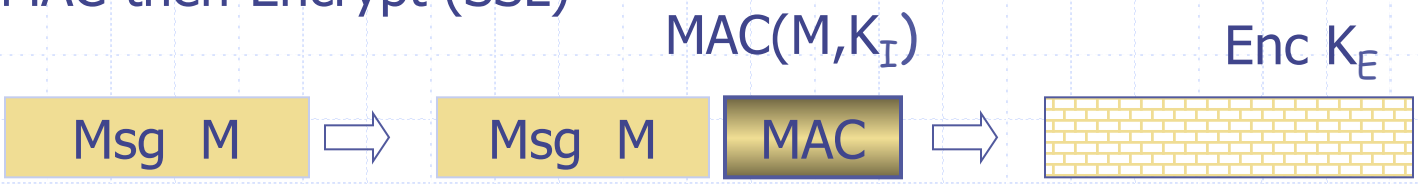
- Why are these MAC constructions secure?
  - … not today – take CS255
- Why the last encryption step in ECBC?
  - CBC (aka Raw-CBC) is not a secure MAC:
    - Given tag on a message m, attacker can deduce tag for some other message m'
    - How: good crypto exercise …

Authenticated Encryption: Encryption + MAC

Combining MAC and ENC (CCA)

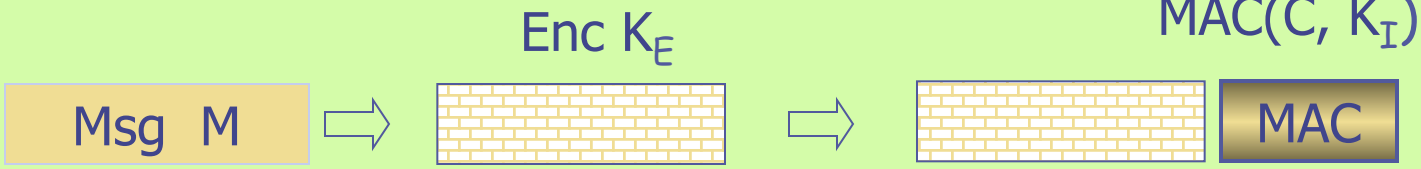
Encryption key K_E MAC key = K_I

Option 1: MAC-then-Encrypt (SSL)

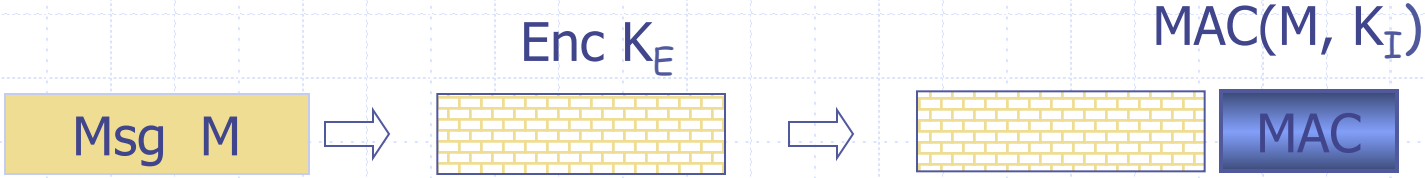


Option 2: Encrypt-then-MAC (IPsec)

Secure on general grounds



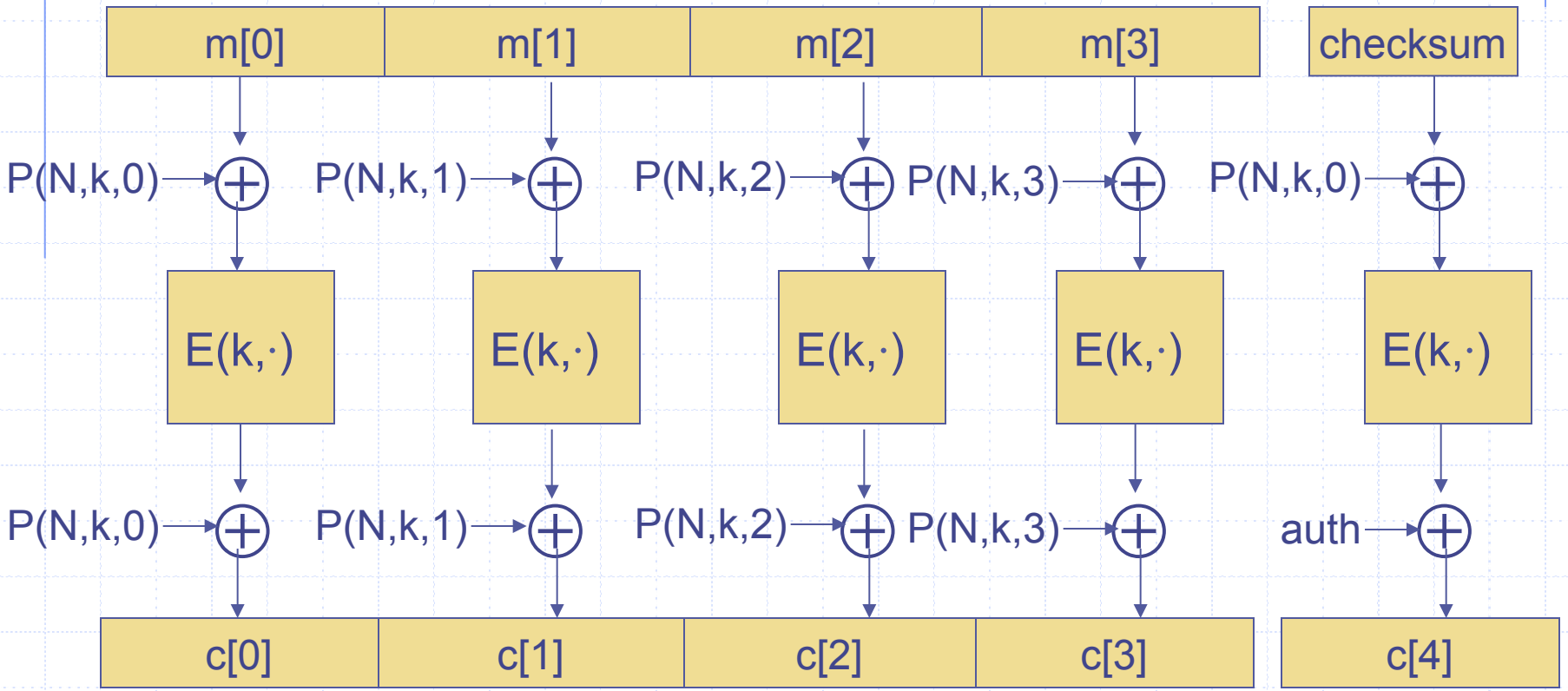
Option 3: Encrypt-and-MAC (SSH)



OCB

offset codebook mode

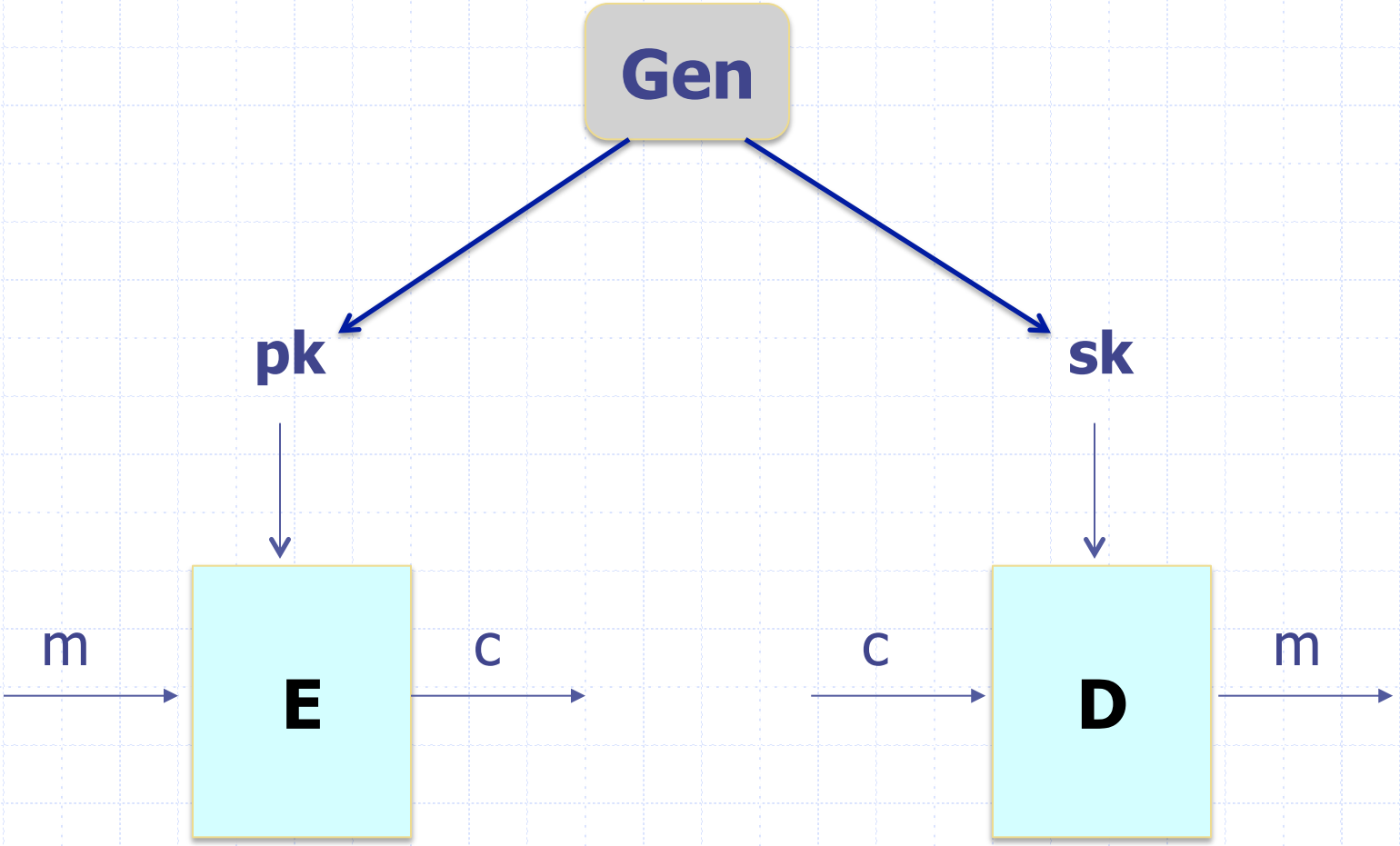
More efficient authenticated encryption



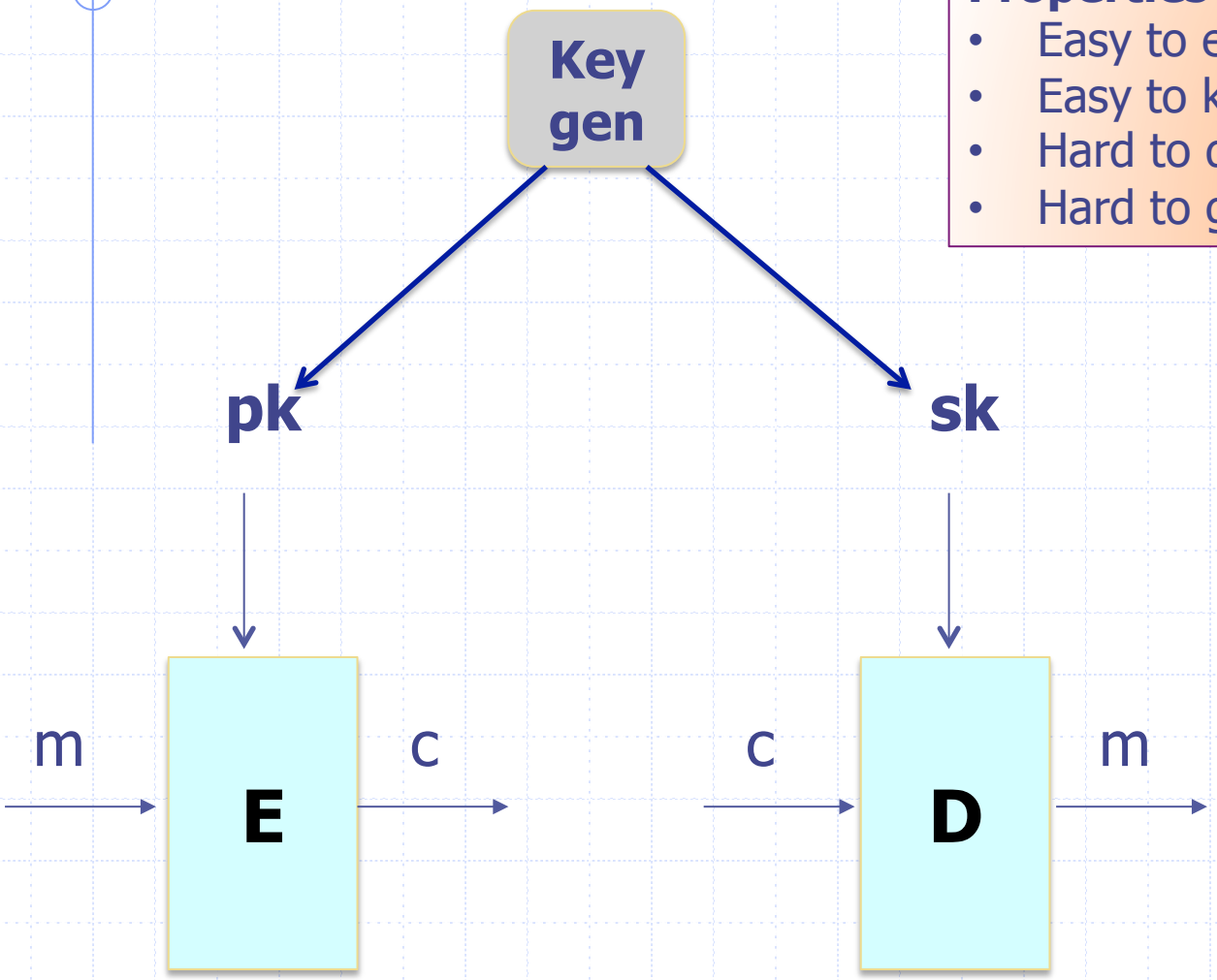
Rogaway, ...

Public-key Cryptography

Public key encryption: (Gen, E, D)



Public key encryption: (Gen, E, D)

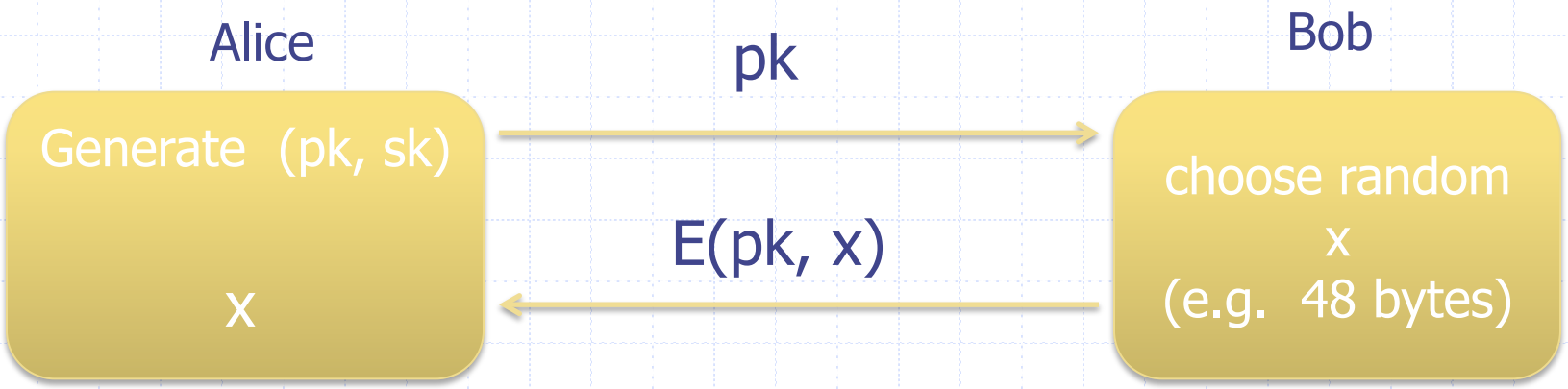


Properties of public key crypto:

- Easy to encrypt
- Easy to keygen
- Hard to decrypt without sk
- Hard to guess sk, knowing only pk

Applications

Session setup (for now, only eavesdropping security)

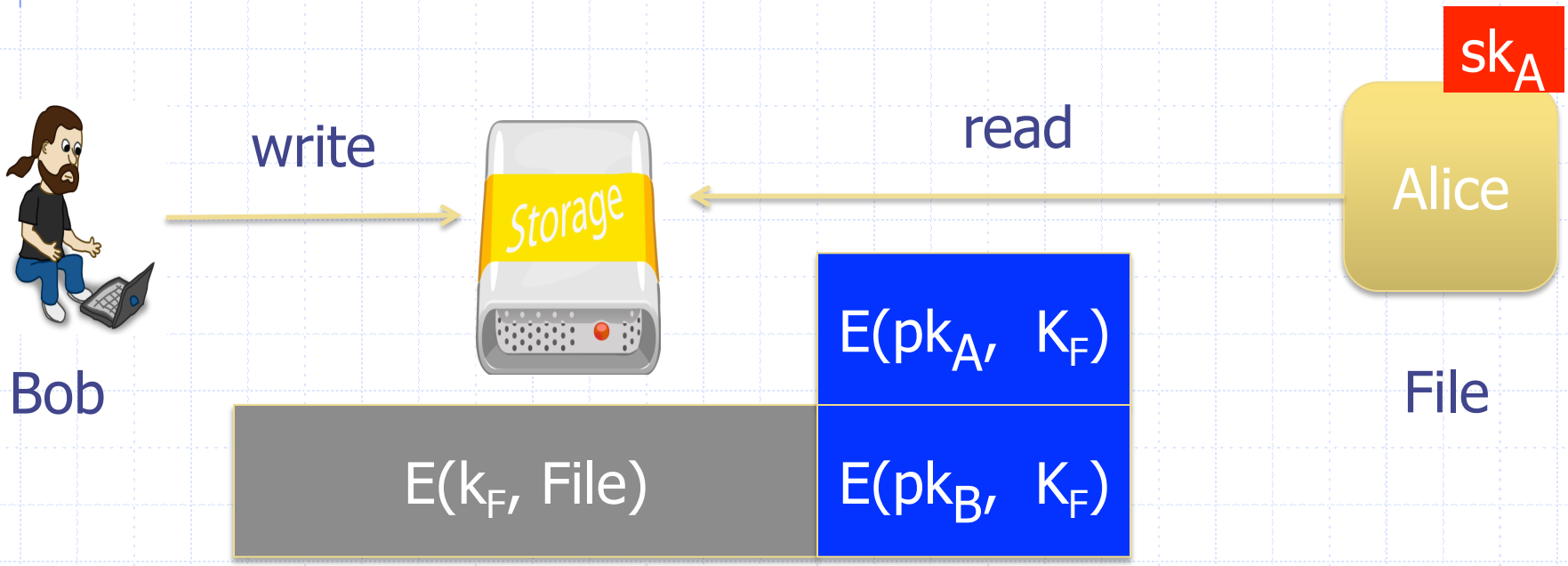


Non-interactive applications: (e.g. Email)

- Bob sends email to Alice encrypted using pk_{alice}
- Note: Bob needs pk_{alice} (public key management)

Applications

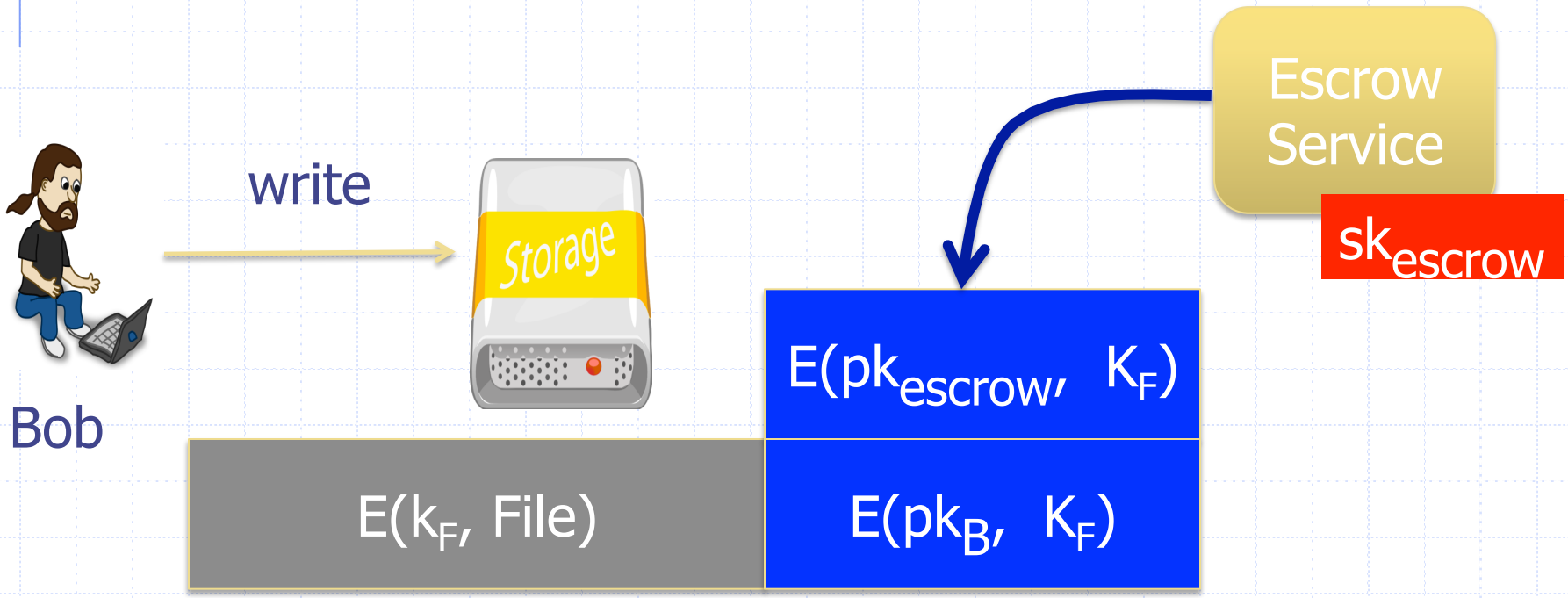
- Encryption in non-interactive settings:
  - Encrypted File Systems



Applications

Encryption in non-interactive settings:

- Key escrow: data recovery without Bob's key



Trapdoor functions (TDF)

Def: a trapdoor func. $X \longrightarrow Y$ is a triple of efficient algs. (G, F, F^{-1})

- G(): randomized alg. outputs key pair (pk, sk)
- F(pk, $\cdot$): det. alg. that defines a func. $X \longrightarrow Y$
- F^{-1}(sk, $\cdot$): defines a func. $Y \longrightarrow X$ that inverts F(pk, $\cdot$)

Security: F(pk, $\cdot$) 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 $X \longrightarrow Y$
- (E_s, D_s) : symm. auth. encryption with keys in K
- H: $X \longrightarrow K$ a hash function

E(pk, m) :

$x \xleftarrow{R} X$, $y \leftarrow F(pk, x)$

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

output (y, c)

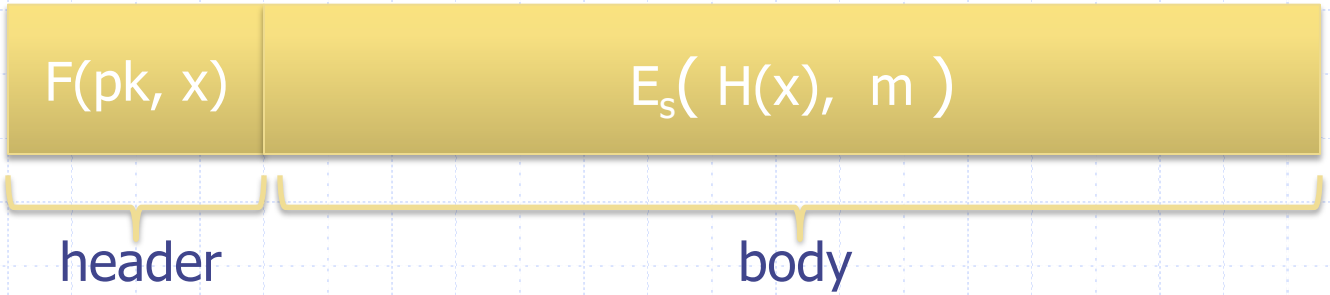
D(sk, (y,c)) :

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

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

output m

In pictures:



Security Theorem:

If **(G, F, F^{-1})** is a secure TDF,

(E_s, D_s) provides auth. enc.

and **H:** $X \longrightarrow K$ is a "random oracle"

then **(G,E,D)** is CCA^{ro} secure.

Digital Signatures

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 TDFs

- (G, F, F^{-1}): secure TDF $X \longrightarrow Y$
- $H: M \longrightarrow Y$ a hash function

Sign(sk, m∈Y) :

output

$$sig = F^{-1}(sk, H(m))$$

Verify(pk, m, sig) :

output

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

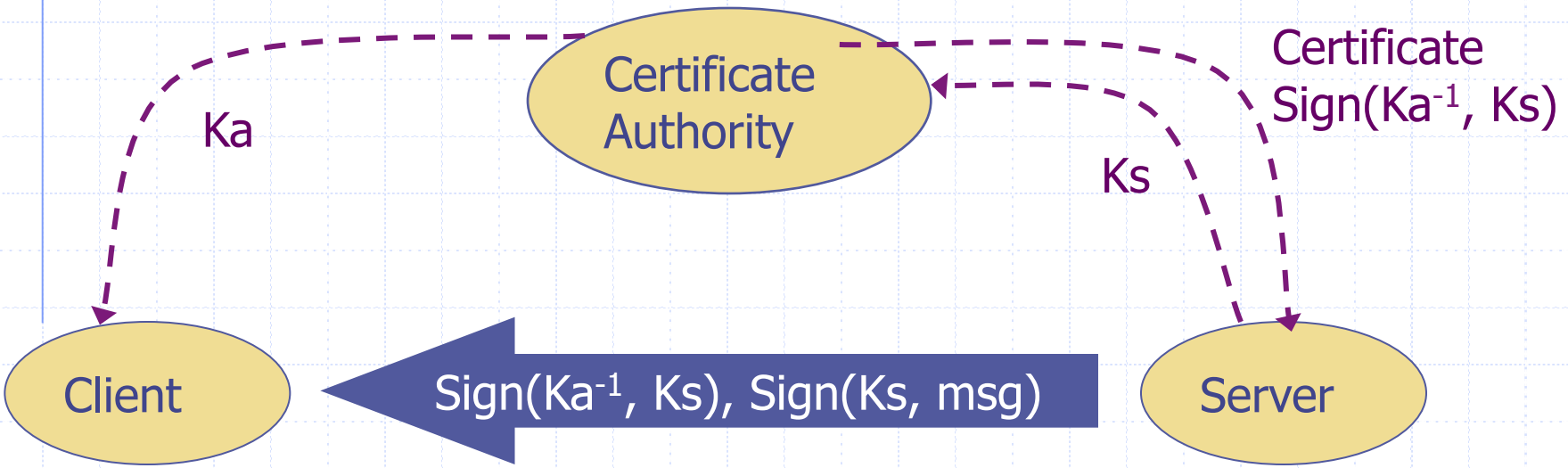
Security: existential unforgeability under a chosen message attack in the random oracle model

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 authority (VeriSign, IBM, United Nations)
    - Everyone must know the verification key of root authority
    - Check your browser; there are hundreds!!
  - Root authority can sign certificates
  - Certificates identify others, including other authorities
  - Leads to certificate chains

Public-Key Infrastructure

Known public signature verification key Ka



Server certificate can be verified by any client that has CA key Ka

Certificate authority is "off line"


Certificate Manager
Your Certificates
Other People's
Web Sites
Authorities
You have certificates on file that identify these certificate authorities:
Certificate Name
Security Device
Comodo CA Limited
Digital Signature Trust Co.
Entrust.net
Equifax
Equifax Secure
Equifax Secure Inc.
GTE Corporation
GeoTrust Inc.
GlobalSign nv-sa
Government Root Certification A...
IPS Internet publishing Services s.l.
IPS Seguridad CA
NetLock Halozatbiztonsagi Kft.
QuoVadis Limited
RSA Data Security, Inc.
RSA Security Inc
SECOM Trust.net
Sonera
View
Edit
Import
Delete
OK

The New York Times

An Attack Sheds Light on Internet Security Holes

By RIVA RICHMOND

Published: April 6, 2011

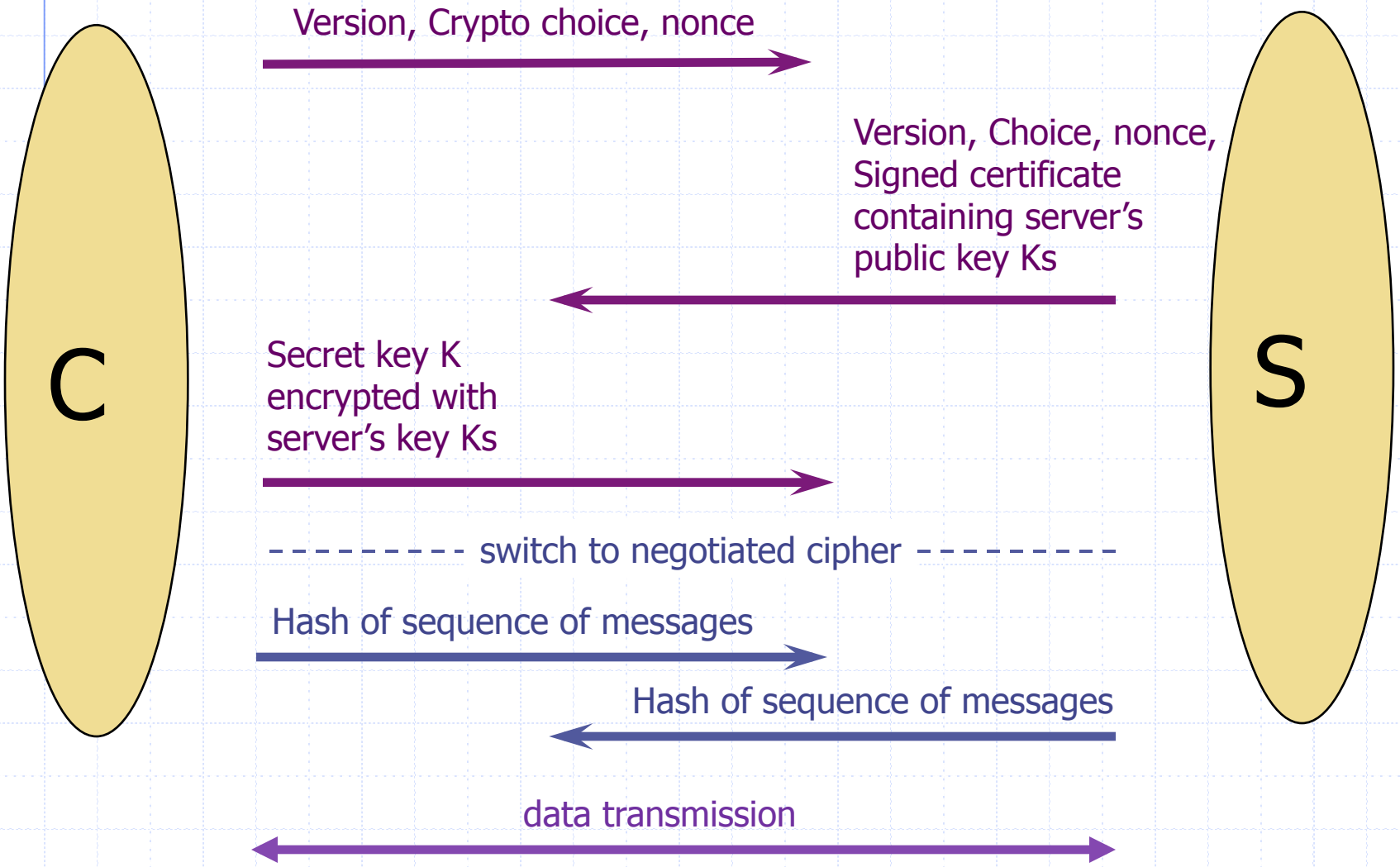
The Comodo Group, an Internet security company, has been attacked in the last month by a talkative and professed patriotic Iranian hacker who infiltrated several of the company's partners and used them to threaten the security of myriad big-name Web sites.
But the case is a problem for not only Comodo, ... It has also cast a spotlight on the global system that supposedly secures communications and commerce on the Web.

The encryption used by many Web sites to prevent eavesdropping on their interactions with visitors is not very secure. This technology is in use when Web addresses start with "https" (in which "s" stands for secure) and a closed lock icon appears on Web browsers. These sites rely on third-party organizations, like Comodo, to provide "certificates" that guarantee sites' authenticity to Web browsers.

But many security experts say the problems start with the proliferation of organizations permitted to issue certificates.

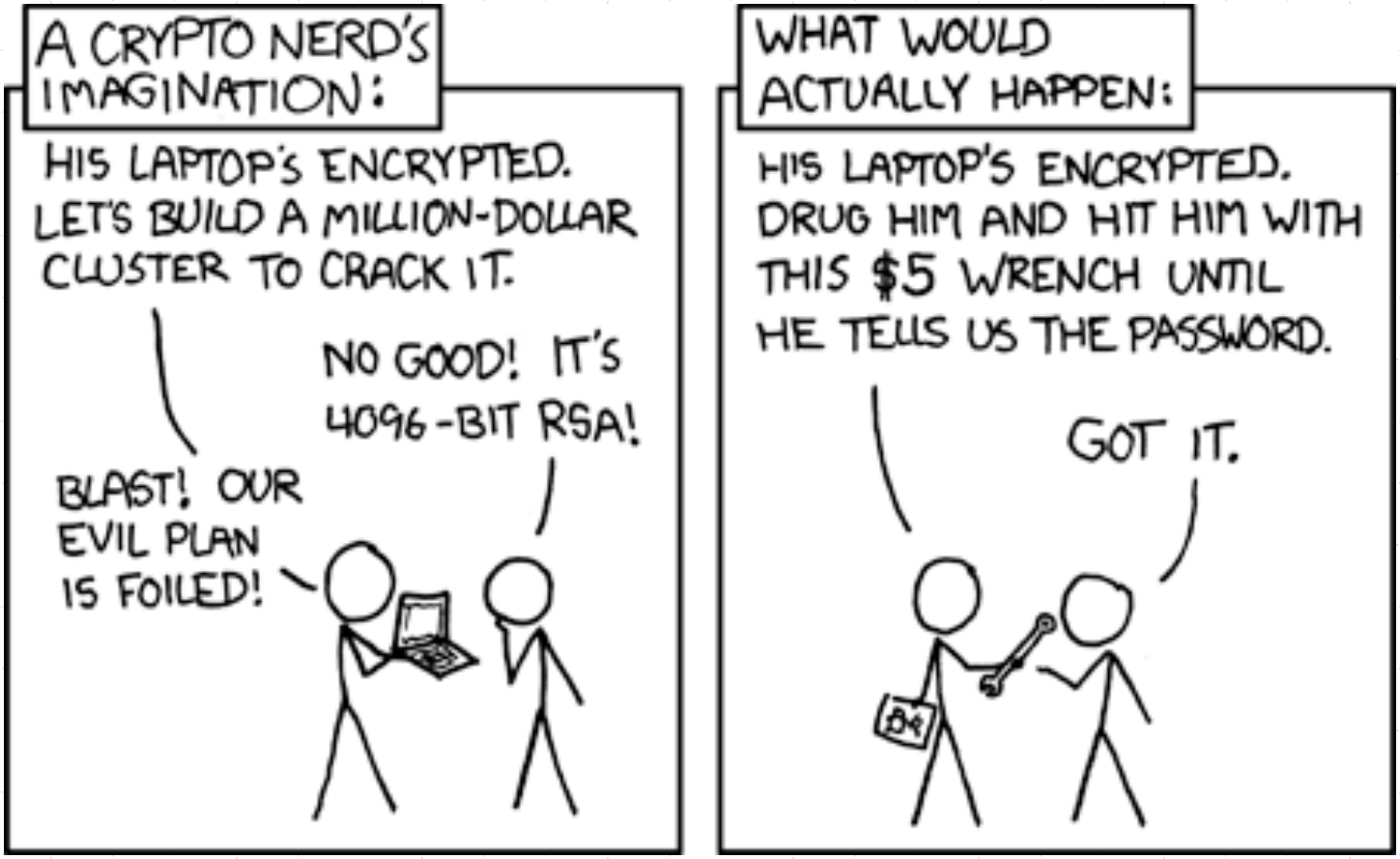
Browser makers like Microsoft, Mozilla, Google and Apple have authorized a large and growing number of entities around the world — both private companies and government bodies — to create them. Many private "certificate authorities" have, in turn, worked with resellers and deputized other unknown companies to issue certificates in a "chain of trust" that now involves many hundreds of players, any of which may in fact be a weak link.

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.