Security

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

CS 240: Computing Systems and Concurrency
Lecture 19

Marco Canini

Selected content adapted from D. Boneh.

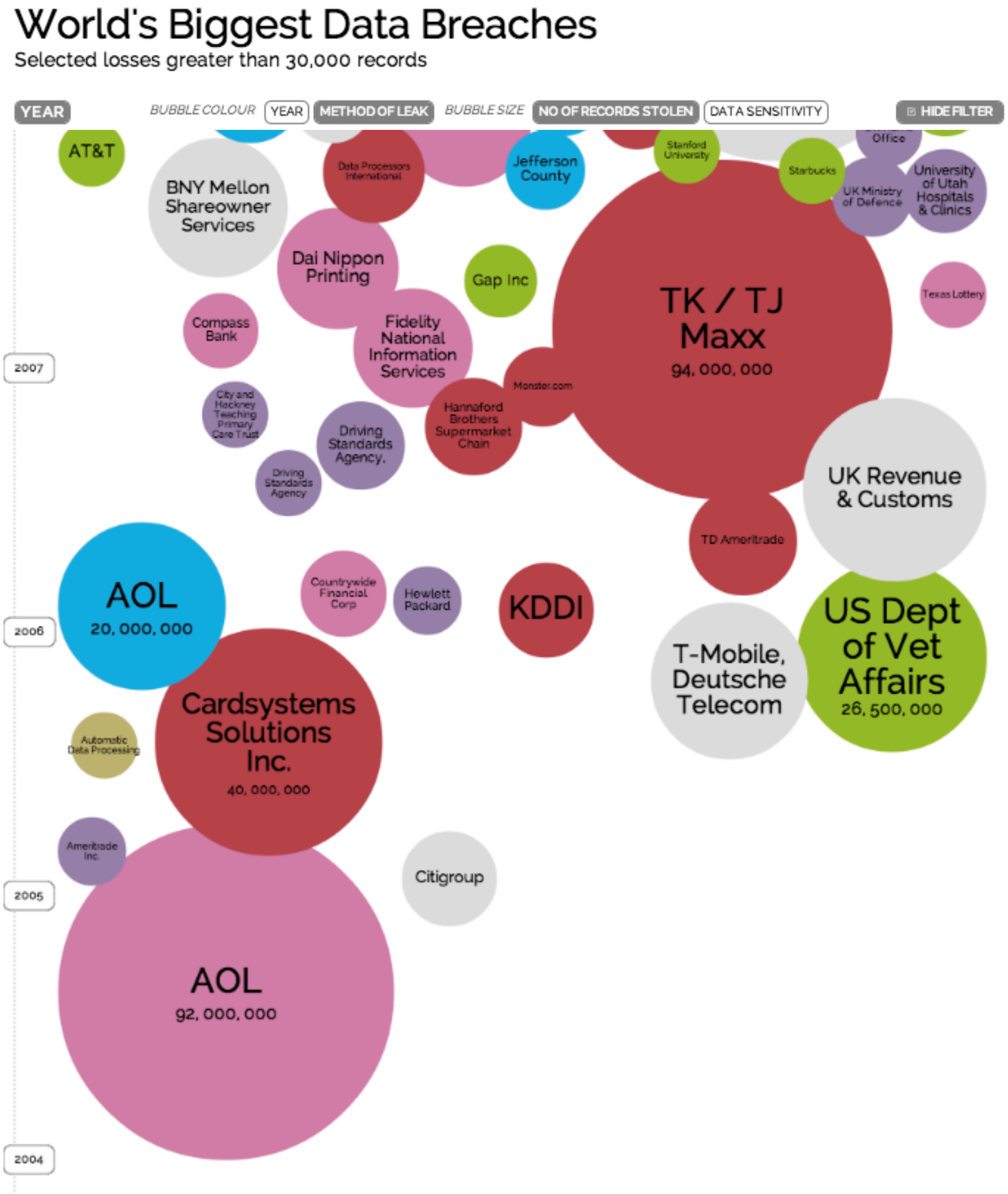
Today

1. **Introdution to Computer Security**
   – What do we mean by security?
2. Introduction to Cryptography
   – Symmetric-key crypto
   – Public-key crypto
   – Crypto hash functions

Let’s start

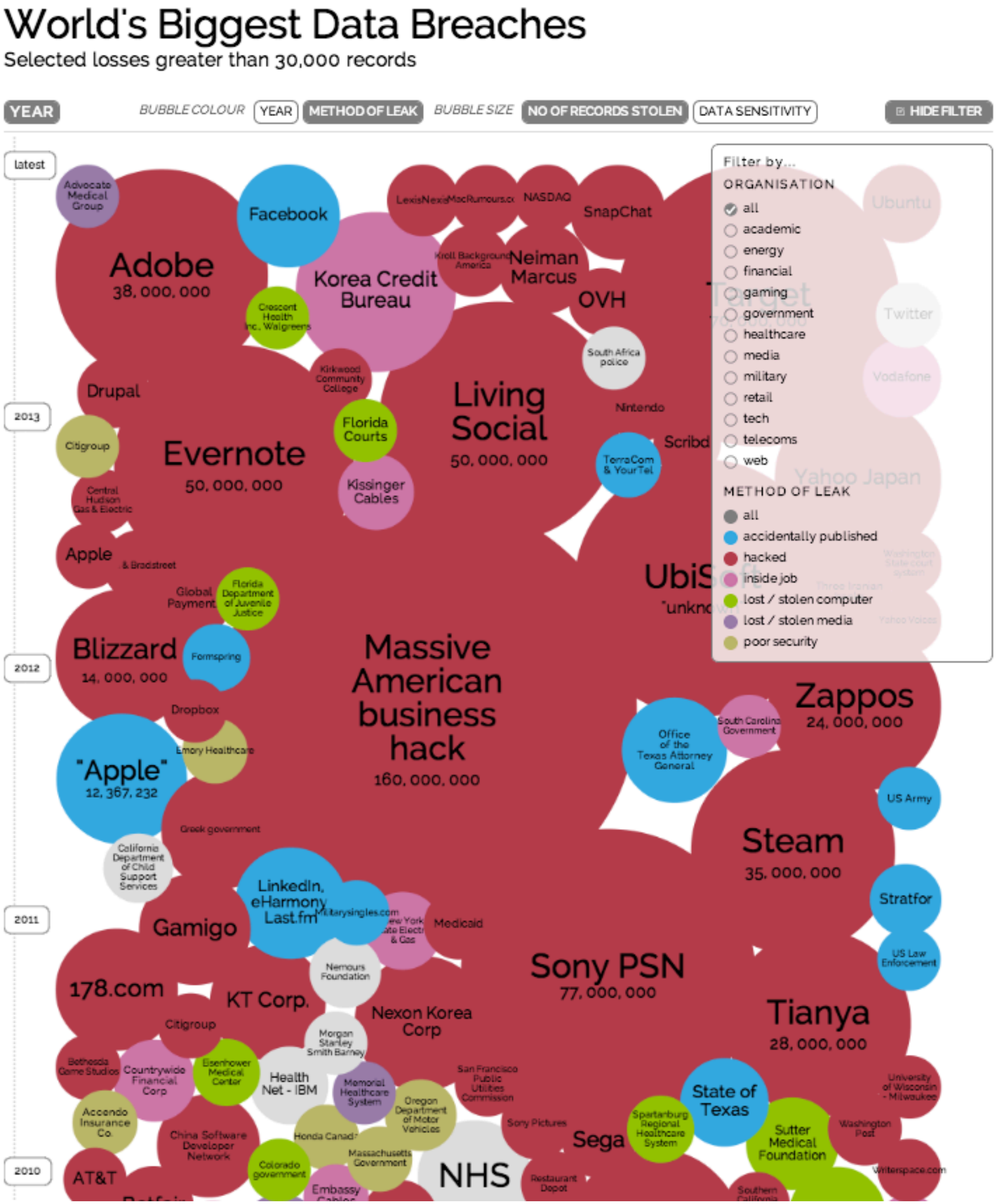
Every day news about new attacks and vulnerabilities

Check out www.cert.org for plenty of examples



http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/

Exponential growth of security incidents over the past 25+ years



Exponential growth of security incidents over the past 25+ years

Networked systems are more and more complex

Difficult to protect them from ever more sophisticated attacks

http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/

The computer security problem

Two factors:

- **Lots of buggy software** (and gullible users)
- **Money can be made from finding and exploiting vulnerabilities**

1. Marketplace for vulnerabilities
2. Marketplace for owned machines (PPI)
3. Many methods to profit from owned client machines

current state of computer security

Why own machines:
1. IP address and bandwidth stealing

Attacker's goal: look like a random Internet user

Use the IP address of infected machine or phone for:

- **Spam** (e.g. the storm botnet)

  Spamalytics: 1:12M pharma spams leads to purchase
  1:260K greeting card spams leads to infection

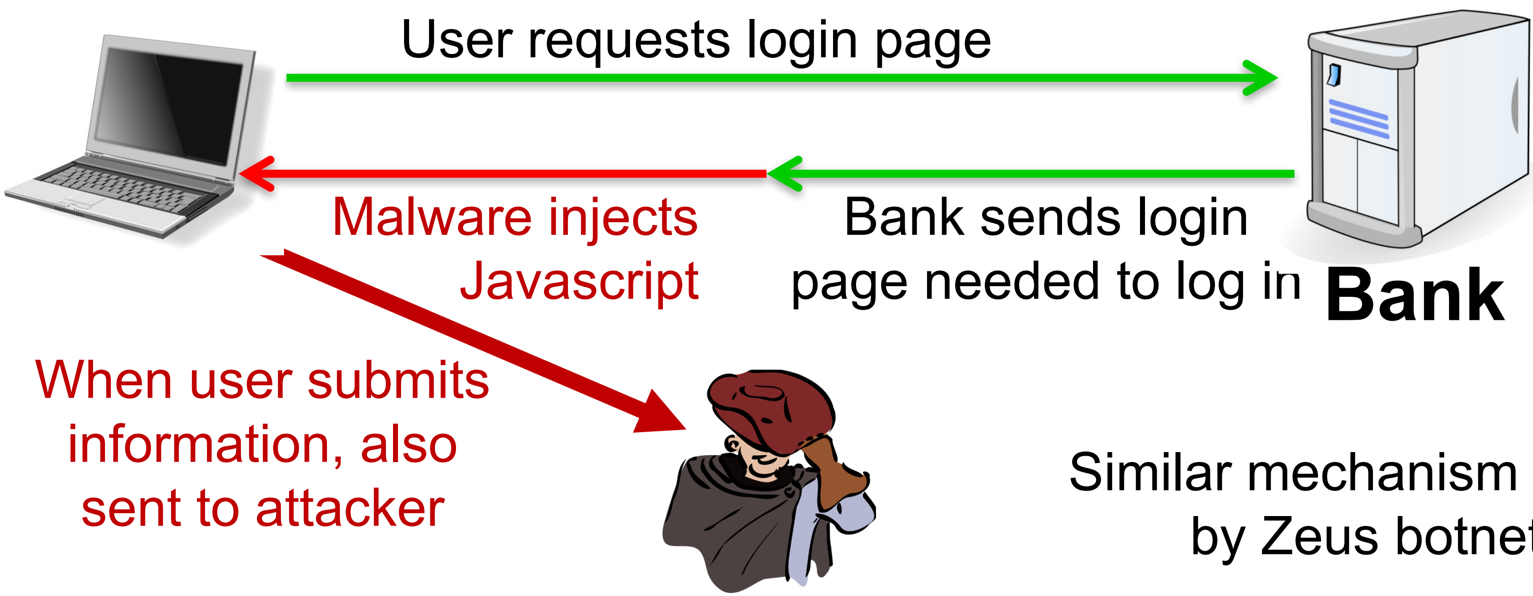
- **Denial of Service:** Services: 1h (20$), 24h (100$)

- **Click fraud** (e.g. Clickbot.a)

Why own machines:
2. Steal user credentials

keylog for banking passwords, web pwds., gaming pwds.

Example: SilentBanker (and many like it)



Similar mechanism used by Zeus botnet

Why own machines:
3. Spread to isolated systems

Example: **Stuxtnet**

Windows infection $\Rightarrow$

Siemens PCS 7 SCADA control software on Windows $\Rightarrow$

Siemens device controller on isolated network

Stuxnet

Stuxnet: Anatomy of a Computer Virus (watch at https://vimeo.com/25118844)
Direction and Motion Graphics: Patrick Clair http://patrickclair.com
Written by: Scott Mitchell
Production Company: Zapruder's Other Films.

What do we mean by security?

What do we mean by security?

- Information security is larger than computer security
  - Defending information from unauthorized access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction

- What does it mean for a computer system to be secure?

What do we mean by security?

- What does it mean for a computer system to be secure?
  - Achieving some goal in the presence of an adversary

  - The system only does what it is expected to
  - Should prevent unauthorized use
  - What is “unauthorized”?
  - What about spam?

When is a computer system secure?

- When it does exactly what it should
  - Not more
  - Not less
- But how to know what a system is supposed to do?
  - Somebody tells us?
    - But do we trust them?
  - We write the specification ourselves?
    - How do we verify that the software meets the specification?
  - We write the code ourselves?
    - But what fraction of the software you use have you written?
    - Can you trust the hardware it runs on?

When is a computer system secure?

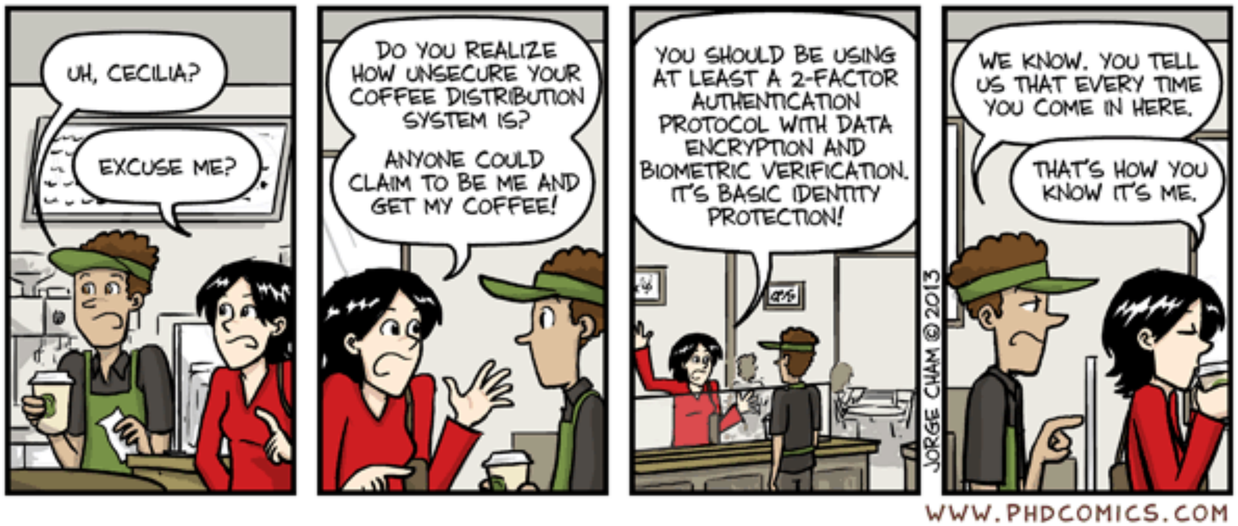
- A program is secure when it doesn't do something it shouldn't
- Easier to specify a list of "bad" things:
  - Delete or corrupt important files
  - Crash my system
  - Send my password or credit card details over the Internet
- But... what if most of the time the program doesn't do bad things, but occasionally it does? Is it secure?
- Difficult to verify that a system does what it is expected to, impossible to verify that it does not what it is not expected to

"Security is mostly a superstition" – Helen Keller (1880-1968), American writer and activist

- Security is all about trade-offs
  - Performance
  - Cost
  - Usability
  - Functionality
- The right question is: how do you know when something is secure enough?
  - Manage security risks vs benefits
  - Requires understanding of the trade-offs involved

How to think about trade-offs?

- What are you trying to protect? How valuable is it?
  - Nuclear missile launch station vs. … coffee machine



- In what way is it valuable?
  - May be important only to one person (e.g. private e-mail or passwords)
  - May be important because accurate and reliable (e.g. bank's accounting logs)
  - May be important because of a service it provides (e.g. Google's web servers)

High level plan

- Policy: the goal you want to achieve
  - e.g. only Alice should read file F
- Threat model: assumptions about what the attacker could do
  - e.g. can guess passwords, cannot physically grab file server
  - Better to err on the side of assuming attacker can do something
- Mechanism: knobs that your system provides to help uphold policy
  - e.g. user accounts, passwords, file permissions, encryption
- Resulting goal: no way for adversary within threat model to violate policy
  - Note that goal has nothing to say about mechanism

Security goals

- Prevent common vulnerabilities from occurring (e.g. buffer overflows)
  - Recover from attacks
- Traceability, accountability and auditing of security-relevant actions
  - Monitoring
- Detect attacks
  - Privacy, confidentiality, anonymity
  - Protect secrets
- Authenticity
  - Needed for access control, authorization, etc.
- Integrity
  - Prevent unwanted modification or tampering
- Availability and reliability
  - Reduce risk of DoS

Classic CIA triad

- **Confidentiality**
  - NO unauthorized disclosure of information
    - E.g. a credit card transaction system attempts to enforce confidentiality by encrypting credit card details over the Internet and in the transaction processing network
- **Integrity**
  - NO unauthorized information modification
    - E.g. traditional Unix file permissions can be an important factor in single system measures for protecting data integrity
- **Availability**
  - Information or system remains available despite attacks
    - High availability systems aim to remain available at all times, preventing disruptions due to power outages, upgrades, hardware failures, Denial of Service (DoS) attacks, …

Example security mechanisms

- Verifying the identity of a prospective user by demanding a password
  - Authentication
- Shielding the computer to prevent interception and subsequent interpretation of electromagnetic radiation
  - Covert channels
- Enciphering information sent communication channels
  - Cryptography
- Locking the room containing the computer
  - Physical aspects of security
- Controlling who is allowed to make changes to a computer system
  - Social aspects of security

Introduction to cryptography

κρμπτο γραφη (Cryptography)

- Greek for "secret writing"
- Confidentiality
  - Obscure a message from eaves-droppers
- Integrity
  - Assure recipient that the message was not altered
- Authentication
  - Verify the identity of the source of a message
- Non-repudiation
  - Convince a 3rd party that what was said is accurate

Things To Remember

- Cryptography is:
  - A tremendous tool
  - The basis for many security mechanisms
- Cryptography is **NOT**:
  - The solution to all security problems
  - Reliable unless implemented and used properly
  - Something you should try to invent yourself
    - many many examples of broken ad-hoc designs
  - Privacy | Steganography | (Encoding|Decoding)

Auguste Kerckhoffs

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

Cryptography is everywhere

Secure communication:

- web traffic: HTTPS
- wireless traffic: 802.11i WPA2 (and WEP), GSM, Bluetooth

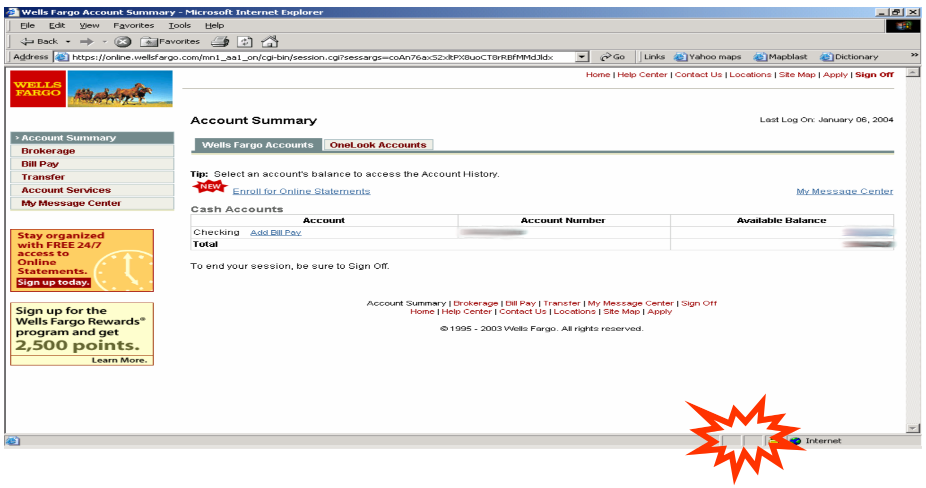
Encrypting files on disk: EFS, TrueCrypt

Content protection (e.g. DVD, Blu-ray): CSS, AACS

User authentication

… and much much more

Secure communication



Alice

HTTPS

SSL / TLS

Bob

no eavesdropping
no tampering

Secure Sockets Layer / TLS

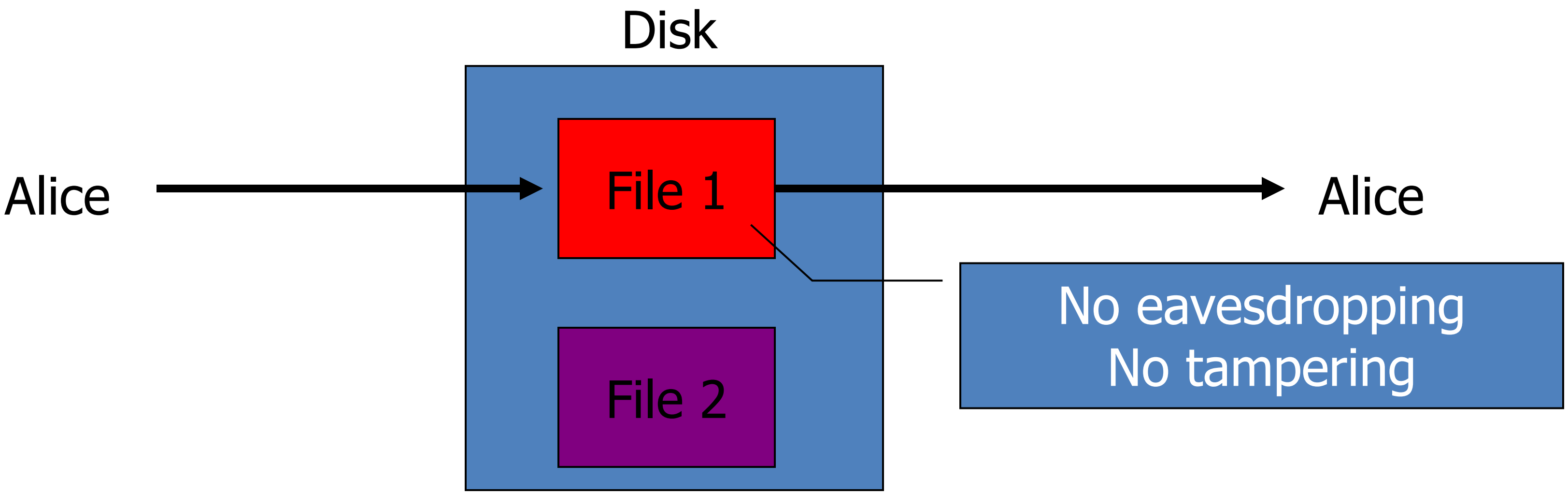
Two main parts

1. Handshake Protocol: **Establish shared secret key using public-key cryptography**

2. Record Layer: **Transmit data using shared secret key**

   Ensure confidentiality and integrity

Protected files on disk



Analogous to secure communication:
Alice today sends a message to Alice tomorrow

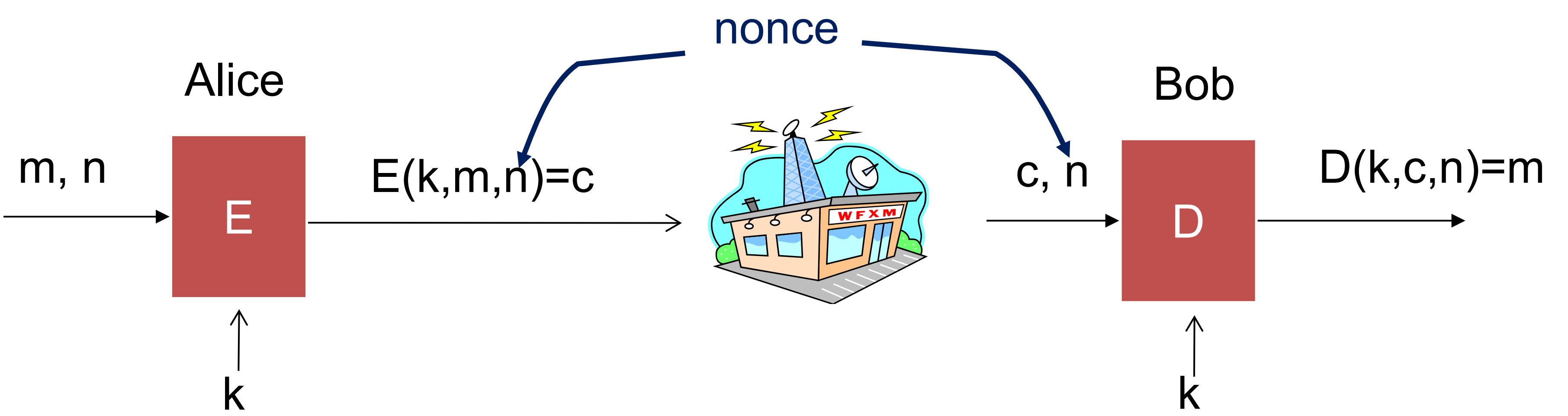
Terminology

- Encryption algorithm
  - Transforms a plaintext into a ciphertext that is unintelligible for non-authorized parties
  - Usually parametrized with a cryptographic key
- Asymmetric (Public) key cryptography
  - Crypto system: encryption + decryption algorithms + key generation
- Symmetric (Shared) key cryptography
  - Cipher/decipher: symmetric-key encryption/decryption algorithms

Symmetric cryptography

Assumes parties already share a secret key
Same secret key for both encryption and decryption

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

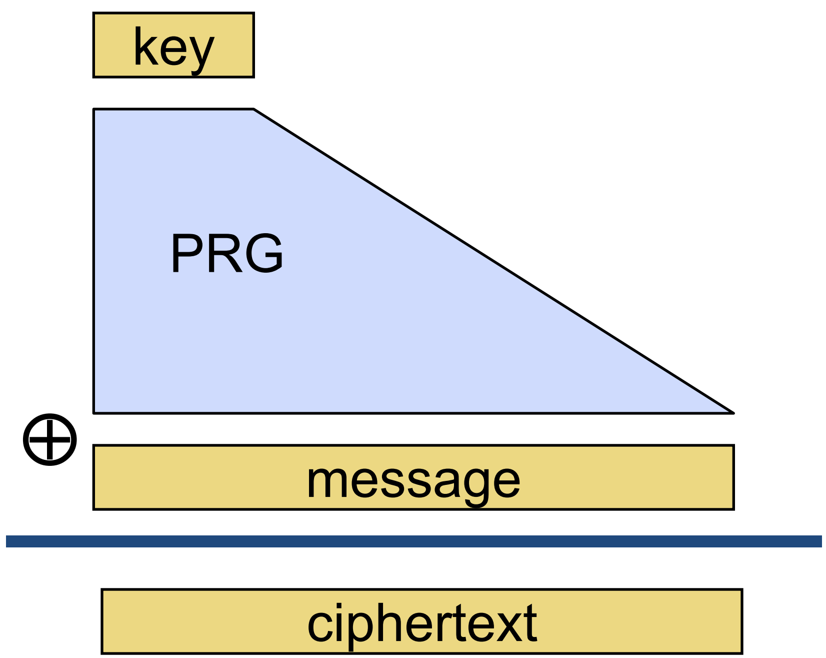
$\oplus$

- Shannon '49:
  - OTP is "secure" against ciphertext-only attacks

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 (126 MB/sec) , Salsa20/12 (643 MB/sec)

Dangers in using stream ciphers

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

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

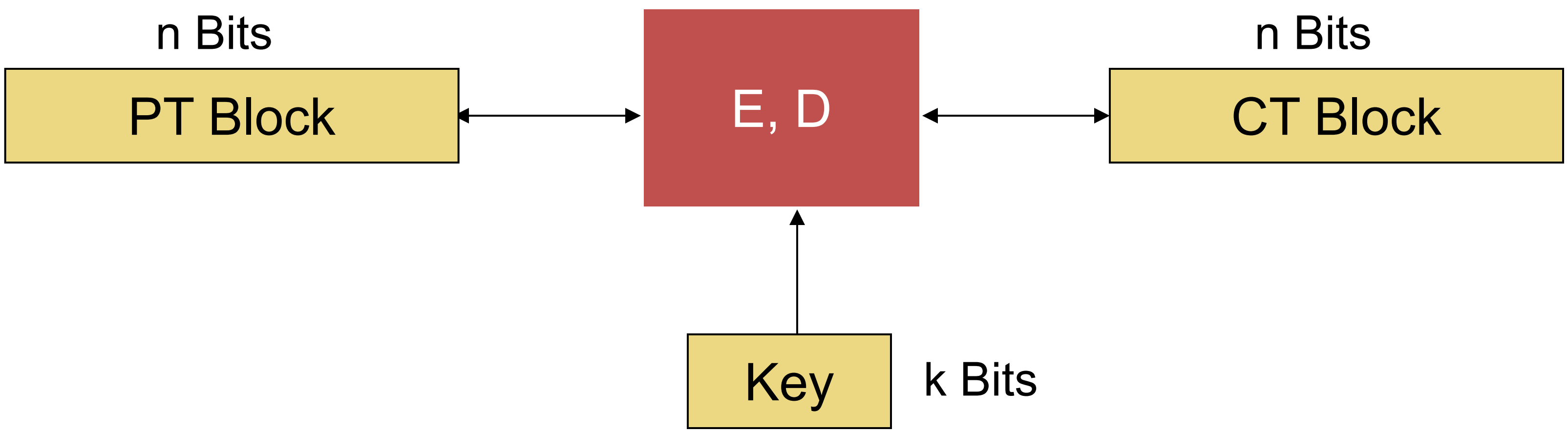
Eavesdropper does:

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

Enough redundancy in English encoding that:

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

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

IV handled as part of PT block

Building a block cipher

Input: (m, k)

Repeat simple "mixing" operation several times

- DES: Repeat 16 times:

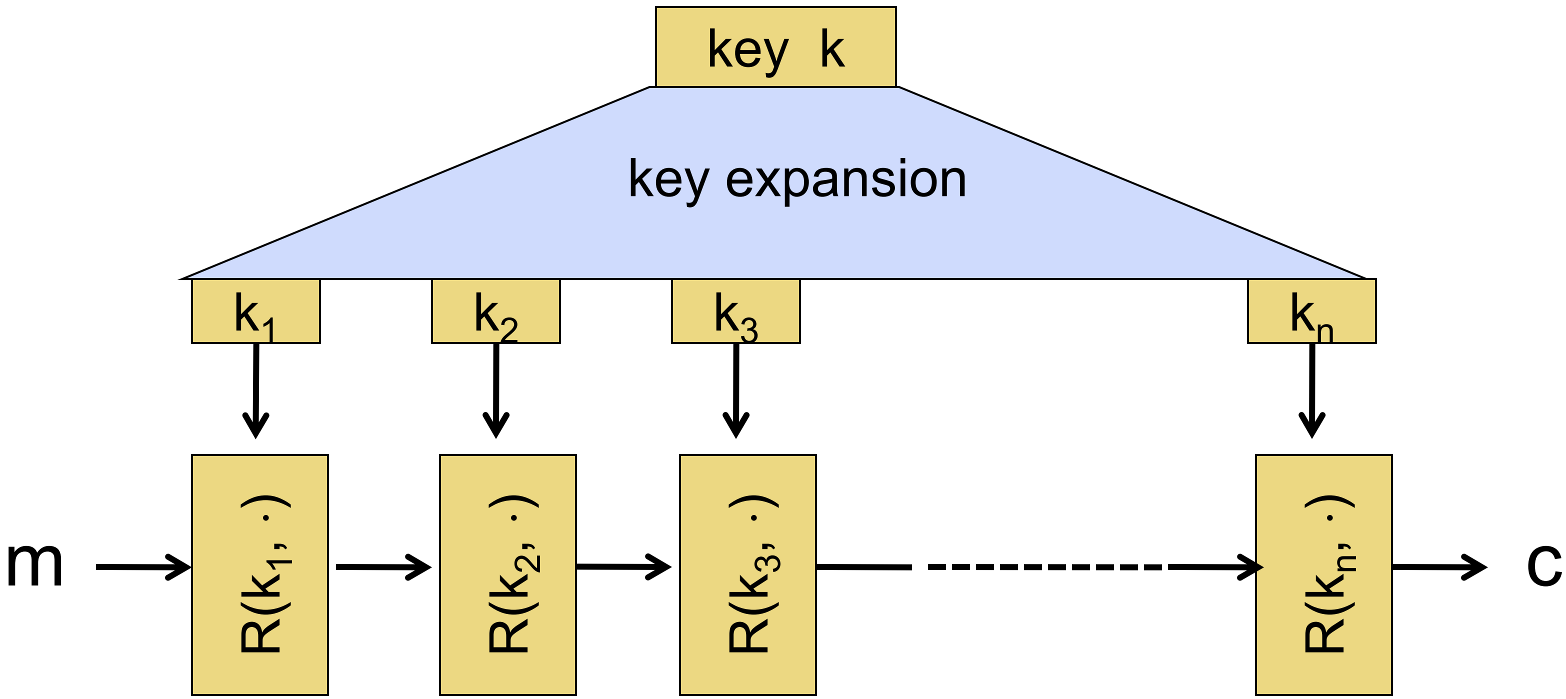
$$\begin{cases} m_L \leftarrow m_R \\ m_R \leftarrow m_L \oplus F(k, m_R) \end{cases}$$

- 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

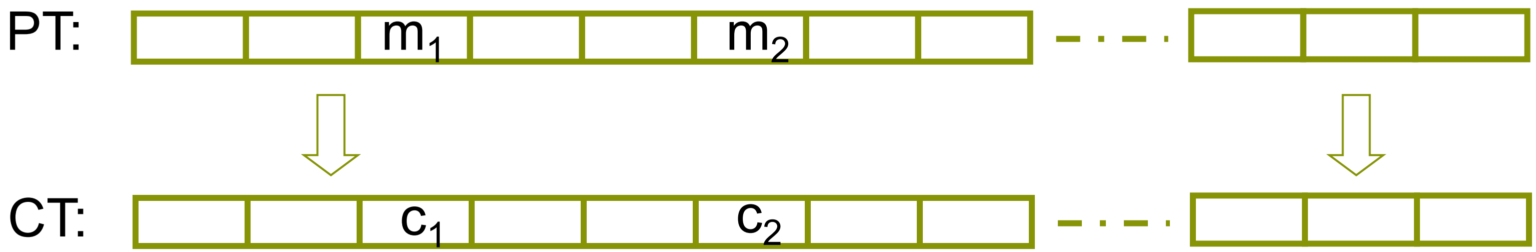


R(k,m): round function

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

Incorrect use of block ciphers

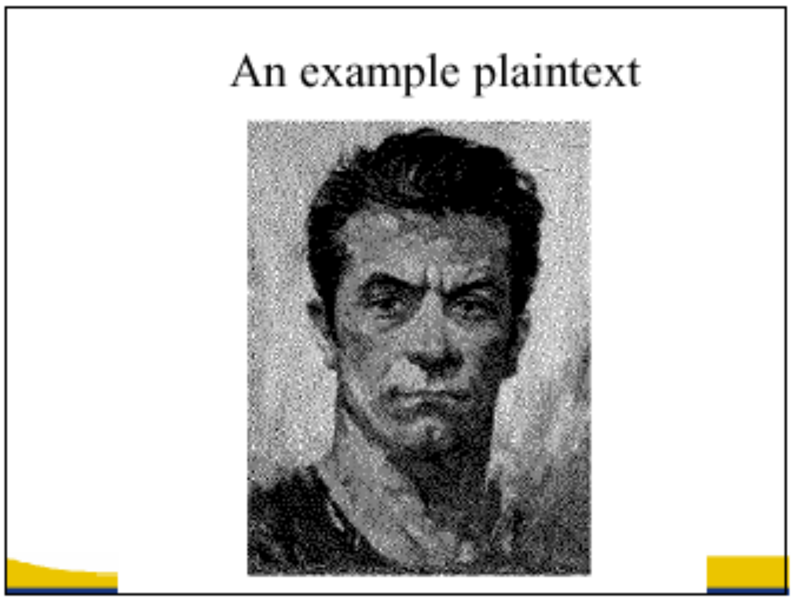
Electronic Code Book (ECB):

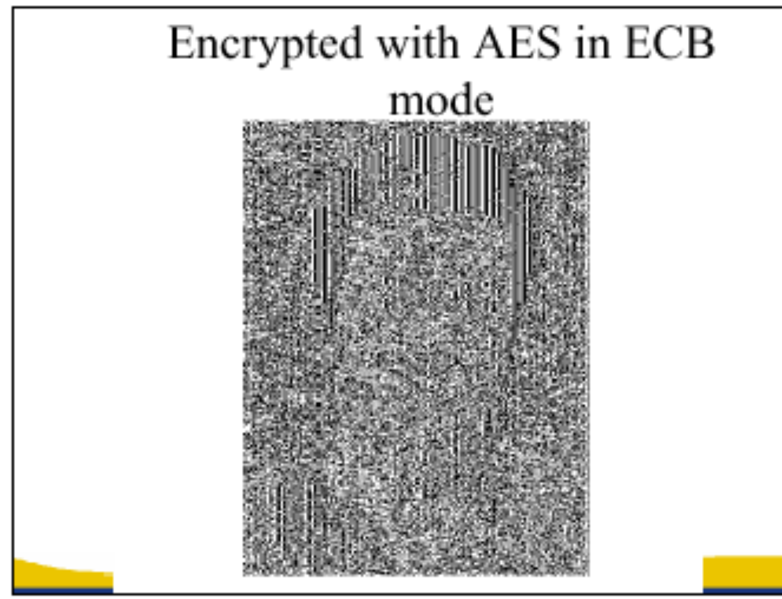


Problem:

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

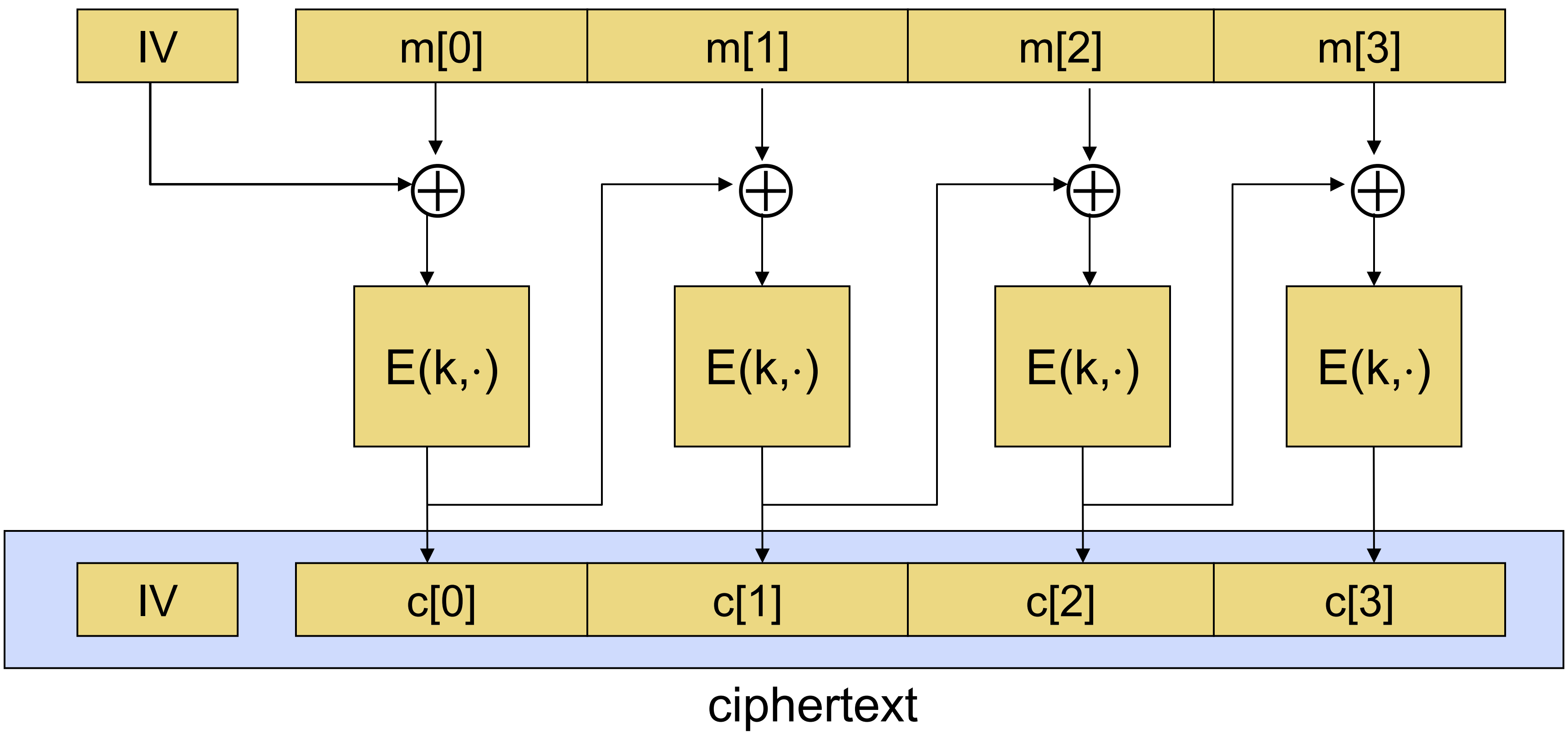
In pictures





Correct use of block ciphers: CBC mode

E a secure PRP. **<u>Cipher Block Chaining</u> with random IV:**



Q: how to do decryption?

Use cases: how to choose an IV

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

Multi use key:

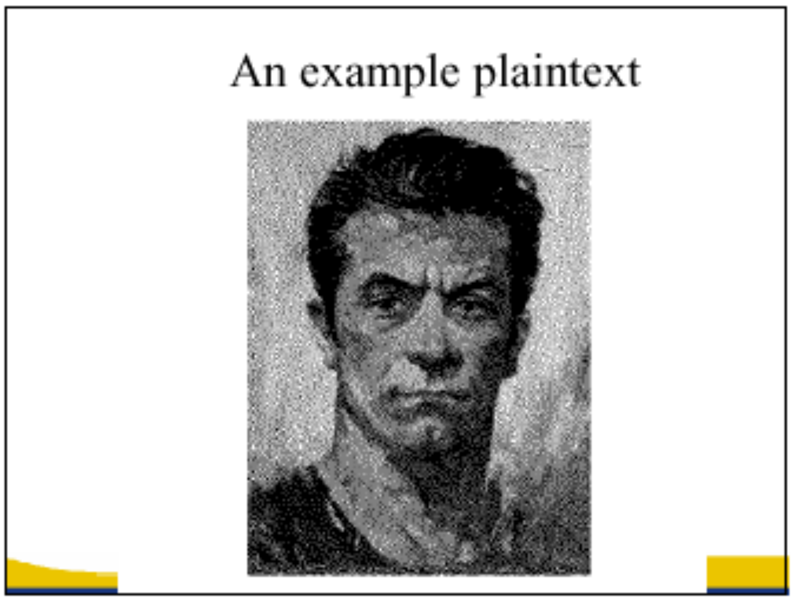
Best: use a fresh <u>*random*</u> IV for every message

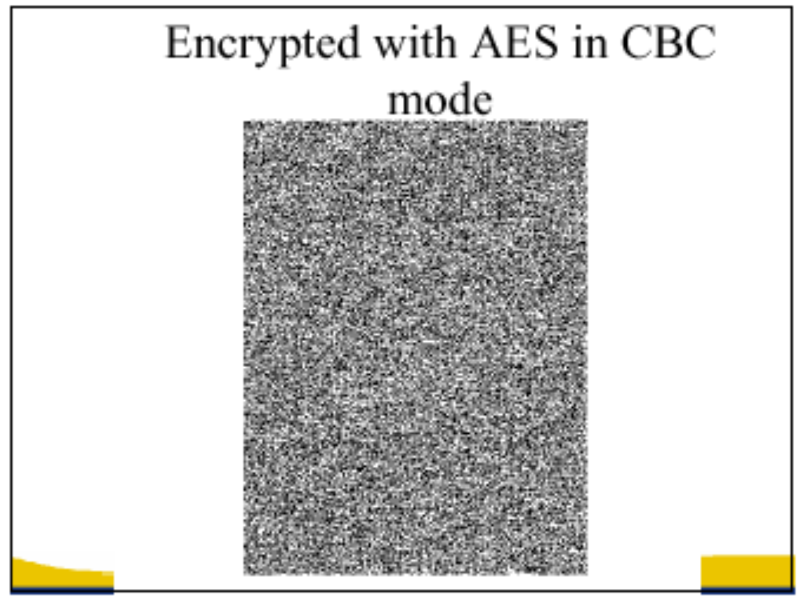
Can use <u>*unique*</u> IV (e.g counter)

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

benefit: may save transmitting IV with ciphertext

In pictures





Performance

Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

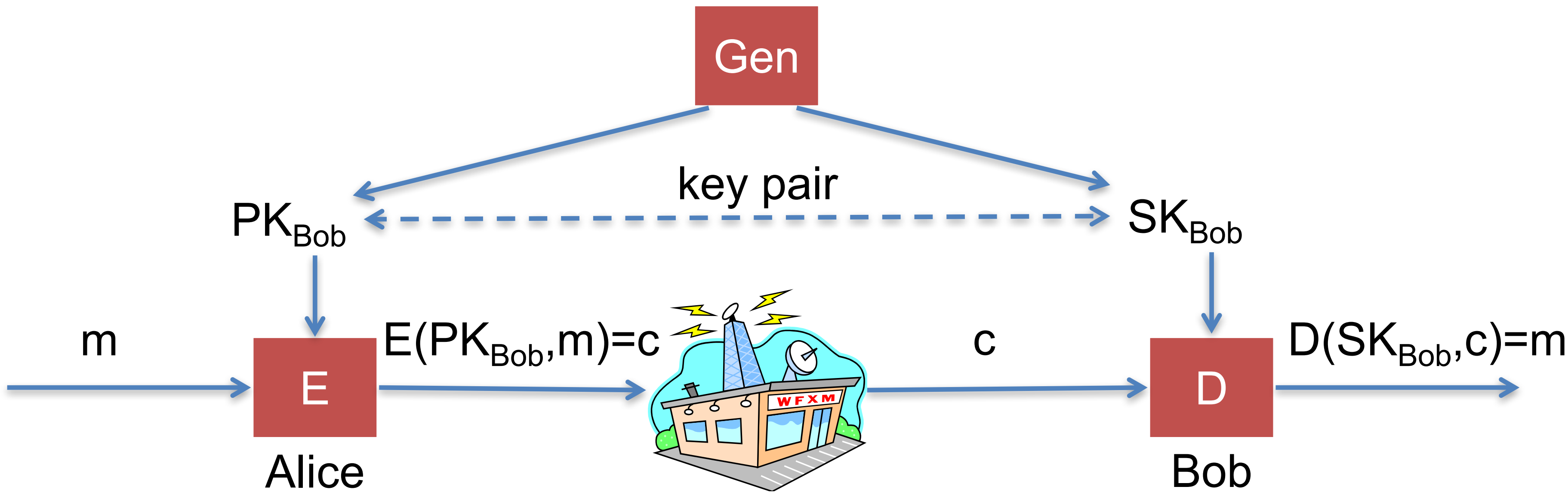
	Cipher	Block/key size	Speed (MB/sec)
stream	RC4		126
stream	Salsa20/12		643
stream	Sosemanuk		727
block	3DES	64/168	13
block	AES-128	128/128	109

Problems with shared key crypto

- Compromised key means interceptors can decrypt any ciphertext they've acquired
  - Change keys frequently to limit damage
- Distribution of keys is problematic
  - Keys must be transmitted securely
  - Use couriers?
  - Distribute in pieces over separate channels?

Public key cryptography

Public Key Encryption

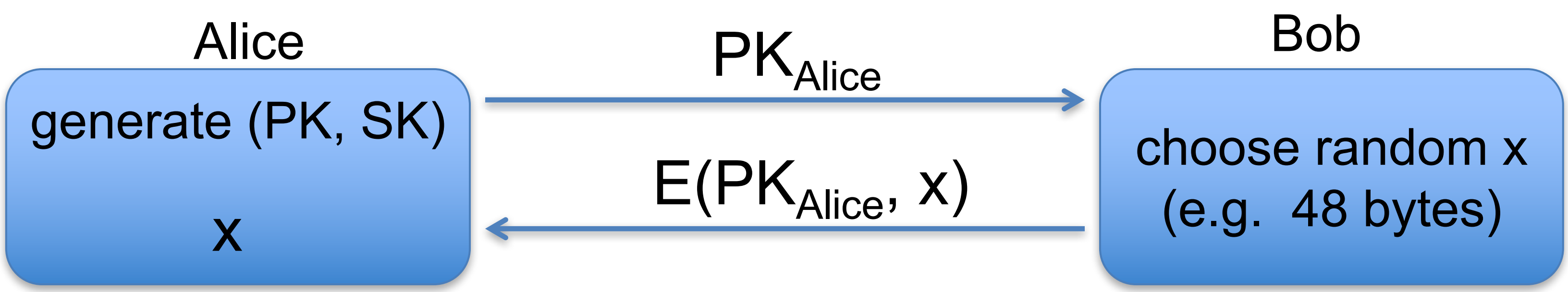


PK: public key , SK: secret key (e.g., 1024 bits)

Example: Bob generates (PK_{Bob}, SK_{Bob}) and gives PK_{Bob} to Alice

Applications

Session setup (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)

(Simple) RSA Algorithm

- Generating a key:
  - Generate composite **n = p * q**, where p and q are secret primes
  - Pick public exponent **e**
  - Solve for secret exponent **d** in $d \cdot e \equiv 1 \pmod{(p-1)(q-1)}$
  - Public key = (e, n), private key = d

- Encrypting message m: $c = m^e \bmod n$
- Decrypting ciphertext c: $m = c^d \bmod n$

- **Security** due to cost of factoring large numbers
  - Finding **(p,q)** given **n** takes $O(e^{\log n \log \log n})$ operations
  - n chosen to be 2048 or 4096 bits long

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

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

Establishing a shared secret

Alice

$(pk, sk) \leftarrow G()$

"Alice", pk →

Bob

choose random $x \in \{0,1\}^{128}$

← "Bob", $c \leftarrow E(pk,x)$

$D(sk,c) \rightarrow x$

x shared secret

Insecure against man in the middle

The protocol is insecure against **active** attacks

Alice | MiTM | Bob

Alice: $(pk, sk) \leftarrow G()$

MiTM: $(pk', sk') \leftarrow G()$

Alice → MiTM: "Alice", pk

MiTM → Bob: "Alice", pk'

Bob: choose random $x \in \{0,1\}^{128}$

Bob → MiTM: "Bob", E(pk', x)

MiTM: $x \leftarrow D(sk', E(pk', x))$

MiTM → Alice: "Bob", E(pk, x)

Authenticated channel

- You should always expect a **man-in-the-middle**
  - e.g. on the internet, your messages go through many intermediaries

- Solution: Use an authenticated channel
  - For instance, Alice and Bob have certificates that contain a public key, and exchange them prior to the msg exchange
  - They use them to authenticate the values in the session setup phase

Trade-offs for Public Key Crypto

- More computationally expensive than symmetric (shared) key crypto
  - Algorithms are harder to implement
  - Require more complex machinery
- More formal justification of difficulty
  - Hardness based on complexity-theoretic results
- A principal needs 1 private key and 1 public key
  - Number of keys for pair-wise communication is O(n)

Cryptographic hash functions

Hash Algorithms

- Take a variable length string
- Produce a fixed length digest

$$h:\{0,1\}^{*} \xrightarrow{\text{hash}} \{0,1\}^{n}$$

- (Non-cryptographic) Examples:
  - Parity (or byte-wise XOR)
  - CRC
- Realistic Example:
  - The NIST Secure Hash Algorithm (SHA) takes a message of less than 2^{64} bits and produces a digest of 160 bits

Cryptographic Hashes

- Create a hard-to-invert summary of input data
- Like a check-sum or error detection code
  - Uses a cryptographic algorithm internally
  - More expensive to compute
- Sometimes called a Message Digest
- Examples:
  - Secure Hash Algorithm (SHA)
  - Message Digest (MD4, MD5)

Desired Properties

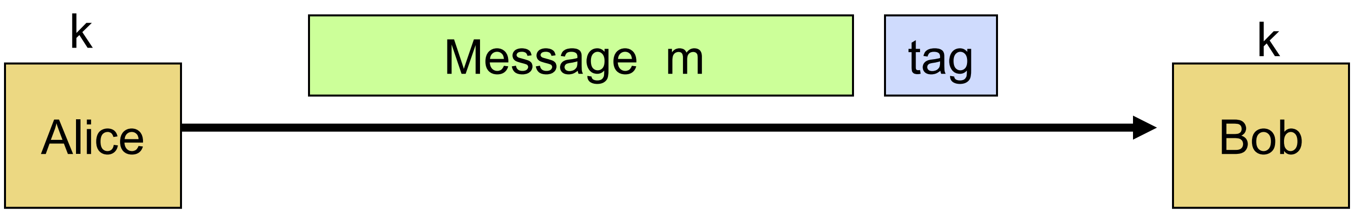
- One way hash function
  - Given a hash value y, it should be infeasible to find m s.t. $\mathrm{h}(m)=y$
- Collision resistance
  - It should be infeasible to find two different messages m_1 and m_2 s.t. $\mathrm{h}(m_1)=\mathrm{h}(m_2)$
- Random oracle property
  - $\mathrm{h}(m)$ is indistinguishable from a random n-bit value
  - Attacker must spend a lot of effort to be able to modify the message without altering the hash value

Data integrity

Message Authentication Codes

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's power: 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.

Standardized method: 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:

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

Maintains performance of the original hash function

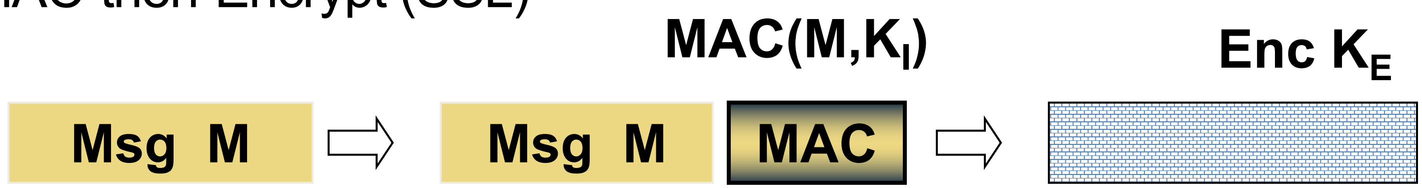
Authenticated Encryption

Encryption + MAC

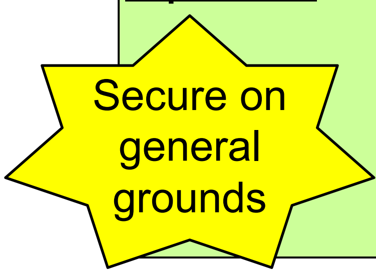
Combining MAC and ENC

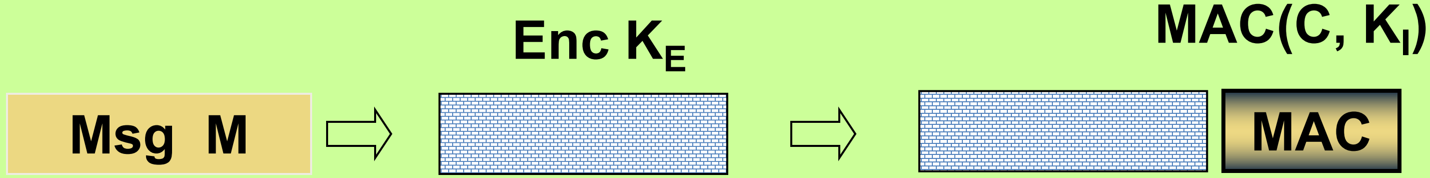
Encryption key K_E MAC key = K_I

Option 1: MAC-then-Encrypt (SSL)

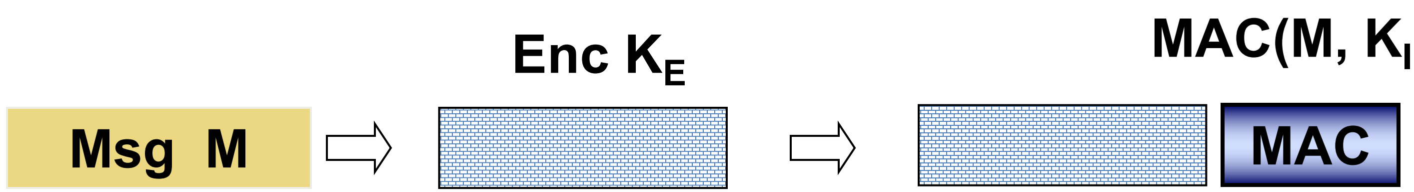


Option 2: Encrypt-then-MAC (IPsec)





Option 3: Encrypt-and-MAC (SSH)

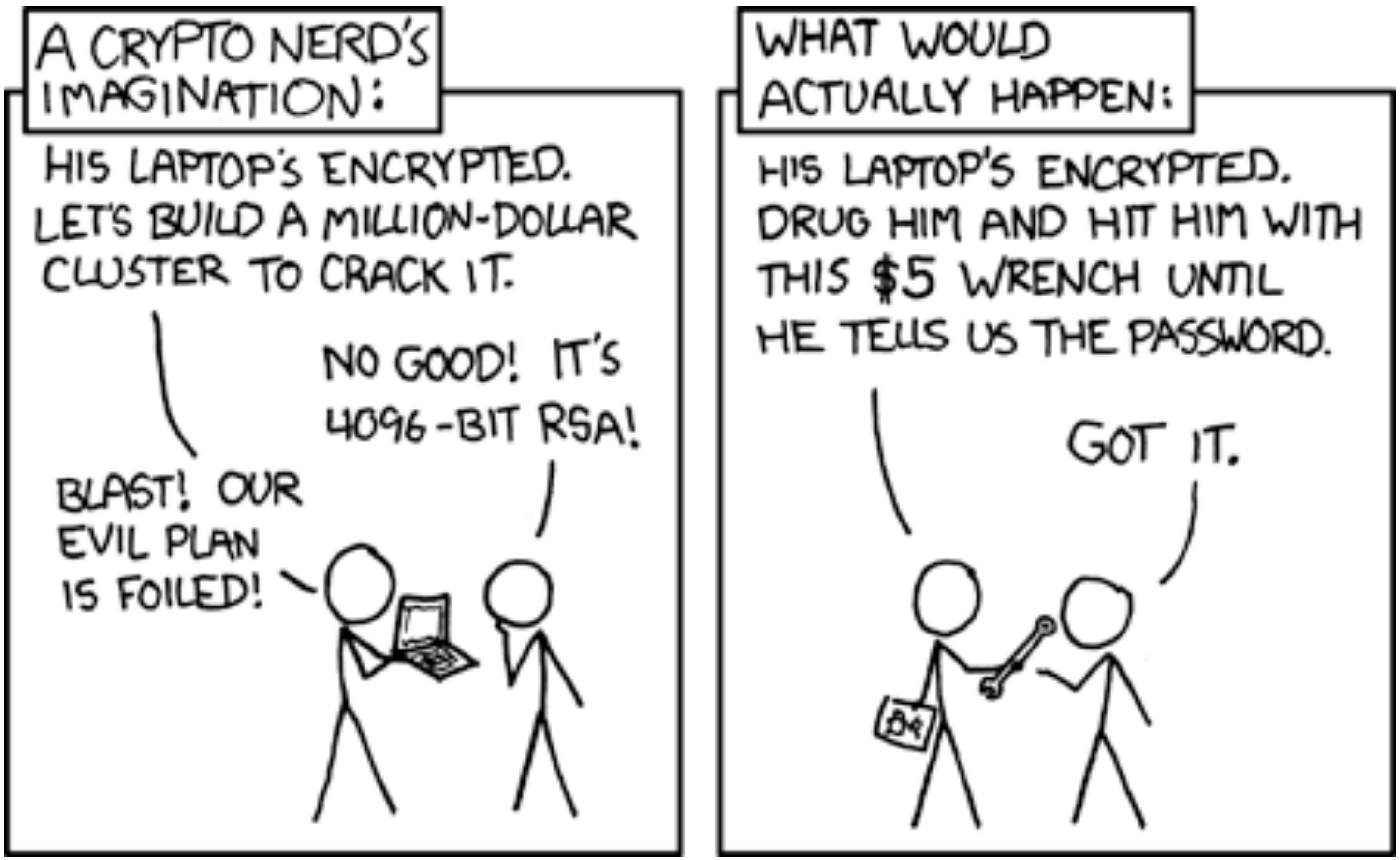


To remember

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

In reality



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.

Wednesday topic:

Bitcoin
and blockchains
and consensus