Cryptographic Tools: Shared-Key Cryptography, Hashes, and Digital Signatures

Monday, November, 2012

Reading: S&M Secs. 7.1 – 7.4, 7.6; Ch. 8

Schneier *Applied Cryptography*, Chs. 1, 7-12 (parts, optional)

CS342 Computer Security

Department of Computer Science
Wellesley College

Cast of Characters

Alice

Eve

M = “one if by land two if by sea”

Mallory

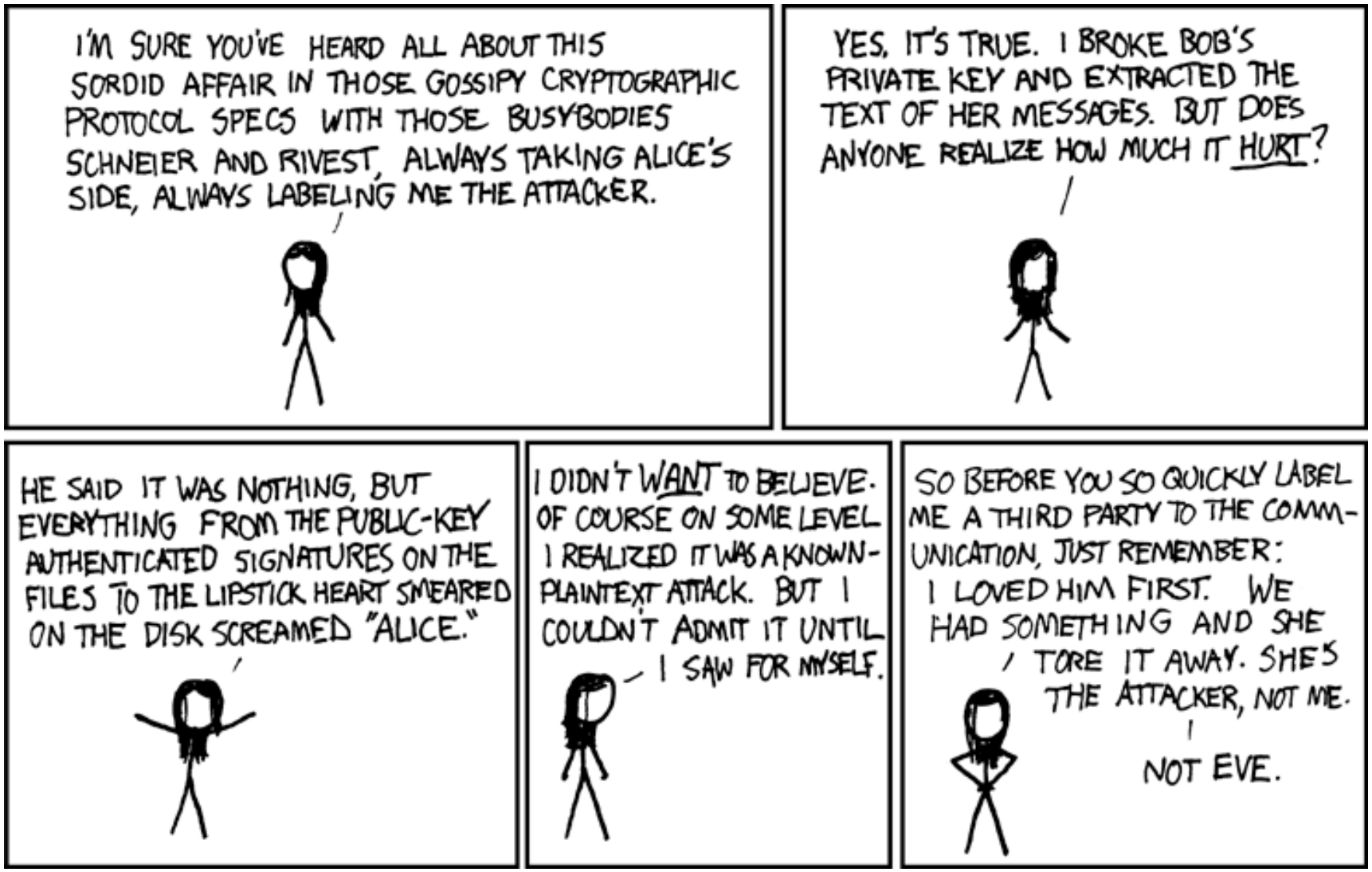
Bob

Alice wants to send a message M to Bob.

Eve may be eavesdropping on the communication (passive attacker)

Mallory may be able to change the message (active attacker)

Eve's Lament



http://xkcd.com/177/

Goals of Cryptography

Confidentiality: No one can read M except Alice and Bob (e.g., whispering, locked box);

Authentication: Bob knows M comes from Alice, not someone masquerading as Alice (e.g., wax seal, watermark paper, signature);

Integrity: Bob can verify that M has not be altered after it was sent by Alice (e.g., tamperproof packaging);

Nonrepudiation: Alice should not be able to falsely deny sending M.

First topic!

encryption/ decryption

message authentication, digital signatures

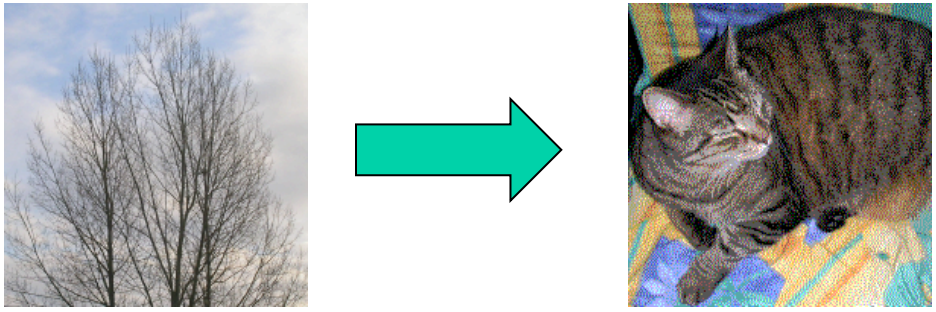
Terminology

Cryptography = scrambling messages: converting messages into "gibberish" that can be converted back to message.

Cryptanalysis = breaking secret codes.

Cryptology = cryptography + cryptanalysis (in practice *cryptography* is often used for *cryptology*).

Steganography = hiding messages, ``security through obscurity''. E.g., under wax, under hair, invisible ink, in lower order bits of images, in whitespace (http://compsoc.dur.ac.uk/whitespace)



Note: Can combine steganography and cryptography.

The Basic Scenario

Alice's key

K_A

Bob's key

K_B

M

Encrypt

$C = E(K_A, M)$

Decrypt

$M = D(K_B, C)$

plaintext
(text,image, voice, …)

ciphertext
(Eve can see
Mallory can see/alter)

plaintext

E(key, msg) is the encryption (enciphering) function

D(key, msg) is the decryption (deciphering) function

Need $D(K_B, E(K_A, M)) = M$ for all M.

Kerkhoffs's Principle

- *Separate cryptography into public algorithms and private keys*
  (Dutch linguist Auguste Kerckhoffs, 1883)

- Private algorithm is an example of **security through obscurity**, which is usually not secure, at least once algorithm is known
  (e.g., Navajo code talkers in WW2).

- Proprietary algorithms often contain holes; public algorithms are analyzed by lots of smart people to find potential problems.

- People (rightly) suspicious of private algorithms, so hard to adopt on widespread basis.

Symmetric vs. Public-Key Cryptography

Symmetric (a.k.a. shared-key, secret-key, private-key):

- Alice and Bob share the same key (or they have different keys that can be easily calculated from one another).
- Communicating keys is a big problem. (How to do e-commerce?)
- Our topic for this lectures

Asymmetric (public-key):

- Bob can publish a public key that anyone (including Alice) can use to encrypt a message, but only Bob has a private key that can decrypt the message. The private key cannot easily be determined from the public key.
- Alice can contact Bob without exchanging keys with him! But public-key management is still a problem.
- Upcoming topic!

Symmetric Cryptography Algorithms

For simplicity, often assume messages & keys use 26-letter alphabet. (in reality, typically use 8-bit = 256-character bytes or raw bits).

In each of the following examples: What is the key? How many keys are there? (Modern crypto relies on computational intractability.)

- Shift cipher
- Vigenere cipher
- One-Time pad
- Substitution cipher
- Transposition cipher
- Rotor Machines
- Block Cipher
- Stream Cipher

Shift Cipher

- **Idea:** "shift" the letter in each plaintext position by the same amount
- Caesar cipher: shift by 3
- Rot13: shift by 13 (doing twice is id)
- How easy is it to break an encrypted message?

shift	message
0	ibm
1	jcn
2	kdo
…	…
25	hal

Shift Cipher: JavaScript Malware, 2010

In a spam .html message sent to Takis:

```
<script type='text/javascript'>
<!--
var s="=nfub!iuuq.frvjw>#sfgsfti#!dpoufou>#1<vsm>iuuq;00cmbdlmfgjmn/dpn0y/iunm#!0?";
m=""; for (i=0; i<s.length; i++) {  if(s.charCodeAt(i) == 28){    m+= '&';} else if
(s.charCodeAt(i) == 23) {     m+= '!';} else {    m+=String.fromCharCode(s.charCodeAt(i)-1);
}}document.write(m);//-->
</script>
```

Shift cipher of 1 for

```
<meta http-equiv="refresh" content="0;url=http://
  blacklefilm.com/x.html" />
```

which redirects to what appears to be a drive-by download site.

Vigenere Cipher

Idea: "shift" the letter in each plaintext position as determined by repeating key

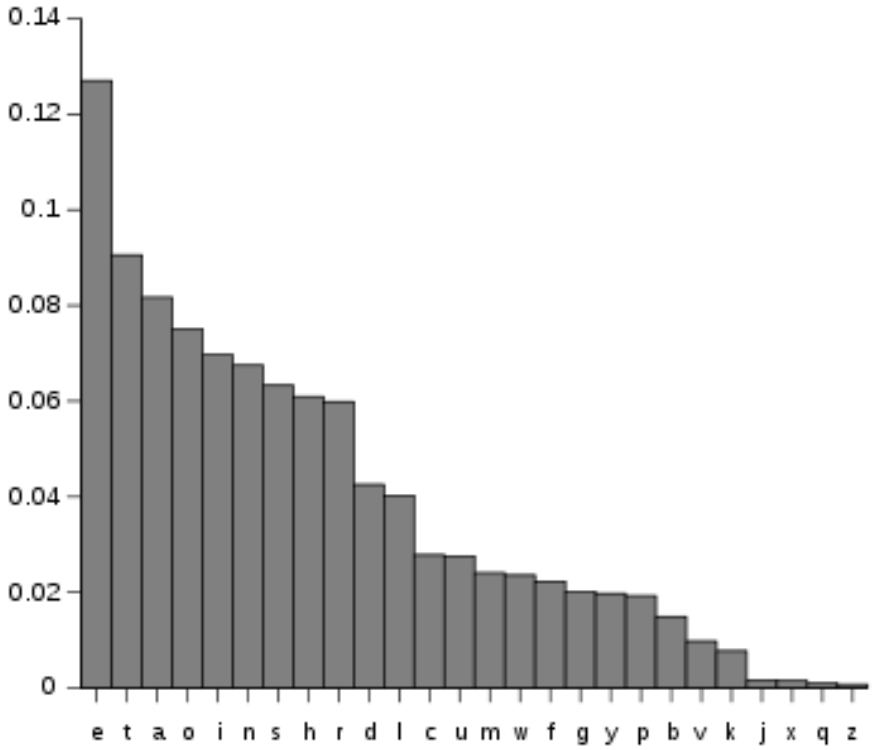
```
Plaintext: oneifbylandtwoifbysea
      Key: cabcabcabcabcabcabcab
Ciphertext: qnfkfcalbpduyojhbzueb
```

- The same plaintext at different positions may be encrypted to different ciphertext (e.g. "ifby").
- How secure is Vigenere? Check out

  http://cs.wellesley.edu/~fturbak/codman-archive/codman-nov-02-2007/

Frequency Analysis

English letter frequencies:



Most common digrams (in order):
th, he, in, en, nt, re, er, an, ti, es, on, at, se, nd,
or, ar, al, te, co, de, to, ra, et, ed, it, sa, em, ro.

The most common trigrams (in order):
the, and, tha, ent, ing, ion, tio, for, nde, has, nce, edt, tis, oft, sth, men

See http://pages.central.edu/emp/LintonT/classes/spring01/cryptography/letterfreq.html

XOR Operation (⊕) On Bits

⊕	0	1
0	0	1
1	1	0

Properties

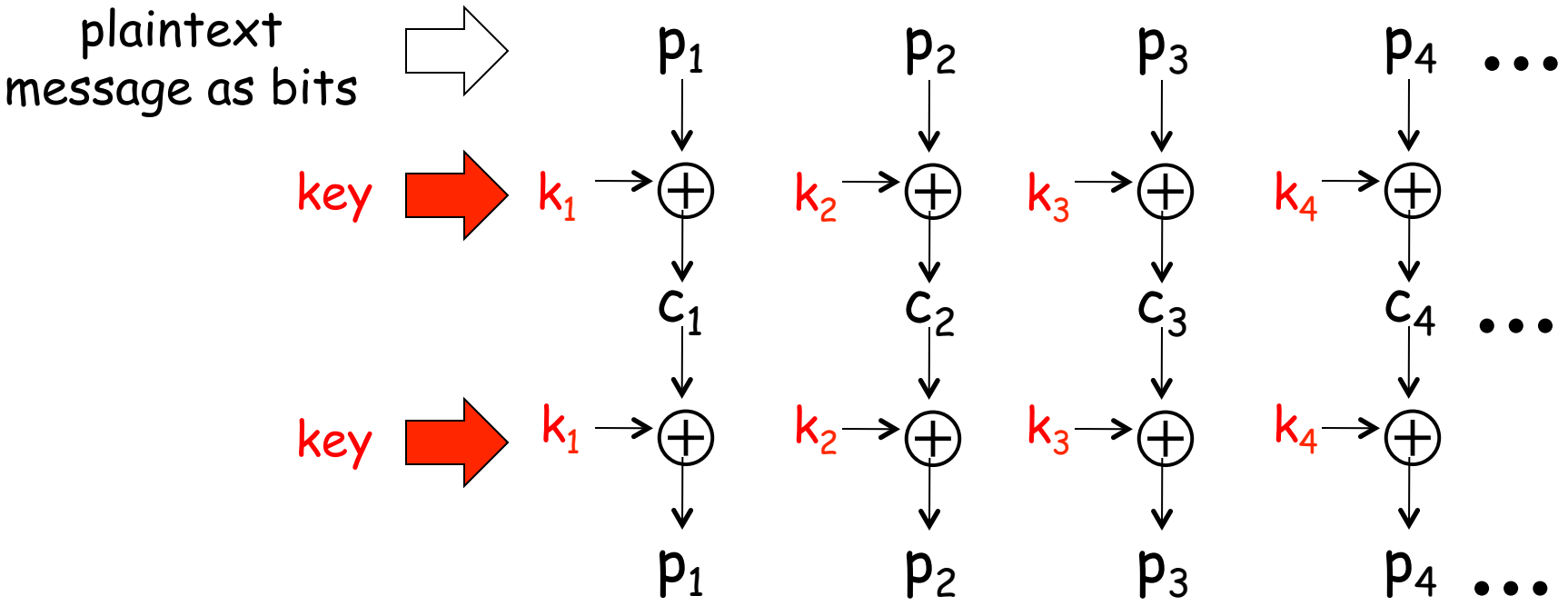
- *Associativity:* (x ⊕ y) ⊕ z = x ⊕ (y ⊕ z)
- *Commutativity*: x ⊕ y = y ⊕ x
- *Identity:* x ⊕ 0 = x = 0 ⊕ x
- *Invertability:* x ⊕ x = 0

Consequence: (p ⊕ k) ⊕ k = p

plaintext	o	n	e	i	f
ascii	111	110	101	105	102
bits	0110 1111	0110 1110	0110 0101	0110 1001	0110 0110
key	1011 0101	0101 1010	1110 1111	0100 0000	0110 1101
ciphertext	1101 1010	0011 0100	1000 1010	0010 1001	0000 1011
key	1011 0101	0101 1010	1110 1111	0100 0000	0110 1101
bits	0110 1111	0110 1110	0110 0101	0110 1001	0110 0110
ascii	111	110	101	105	102
plaintext	o	n	e	i	f

One-Time Pad

View message as n bits and have n-bit random "pad" as key.
Effectively Vigenere with key as long as message.



Evaluating One-Time Pad

- If used properly, one-time pad is **perfectly secure**.
  Ciphertext C for message M can decrypt to *any* message M' with *some* key.

- New key must be chosen for every message:

  - Reusing keys breaks security by opening messages to analysis (e.g., Russian Venona ciphers).

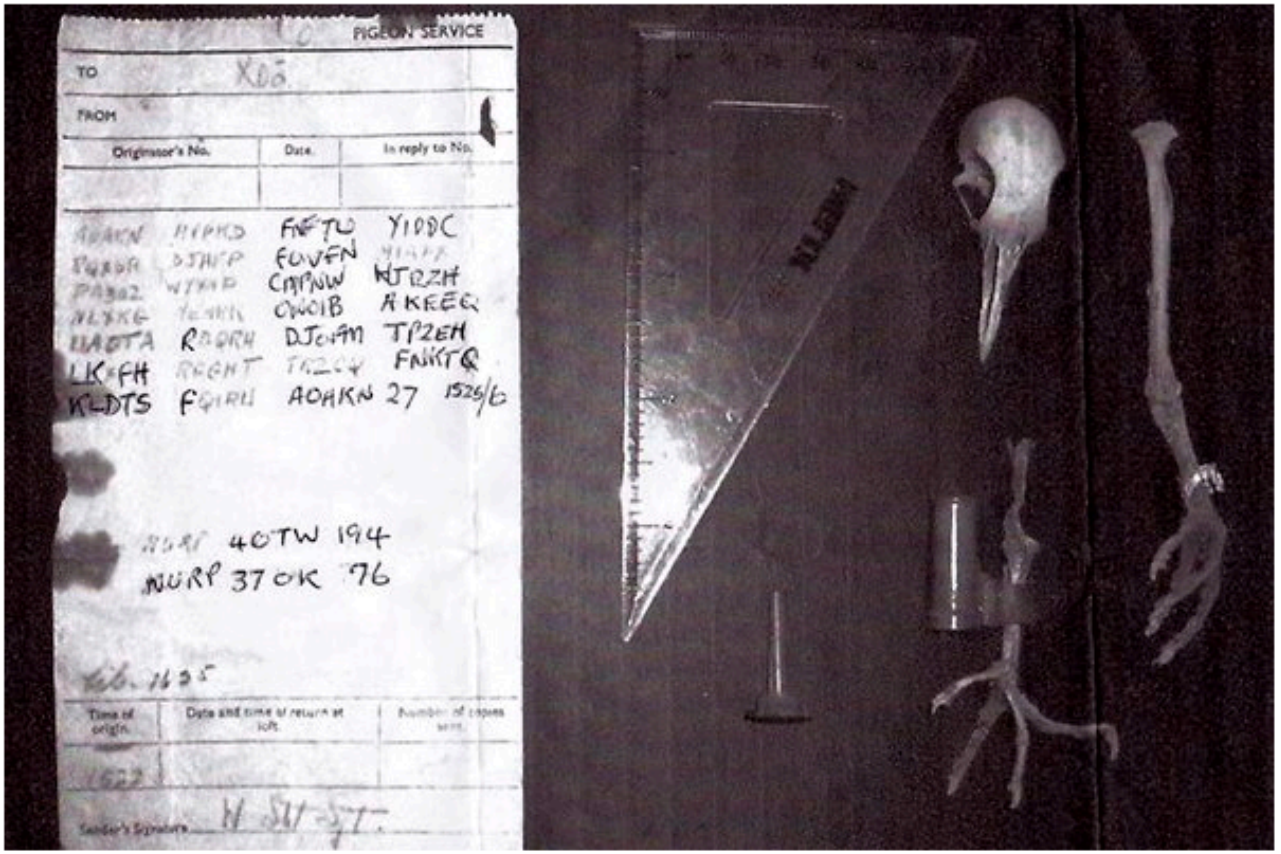
  - How to communicate such long keys securely?
    (Why not send messages the same way?)

- If pad isn't really random but psuedo-random generated by seed, this is a *stream cipher* .

One-Time Pads in the News this Month

MEMO FROM EUROPE

A Bird Skeleton, a Code and, Maybe, a Top Secret



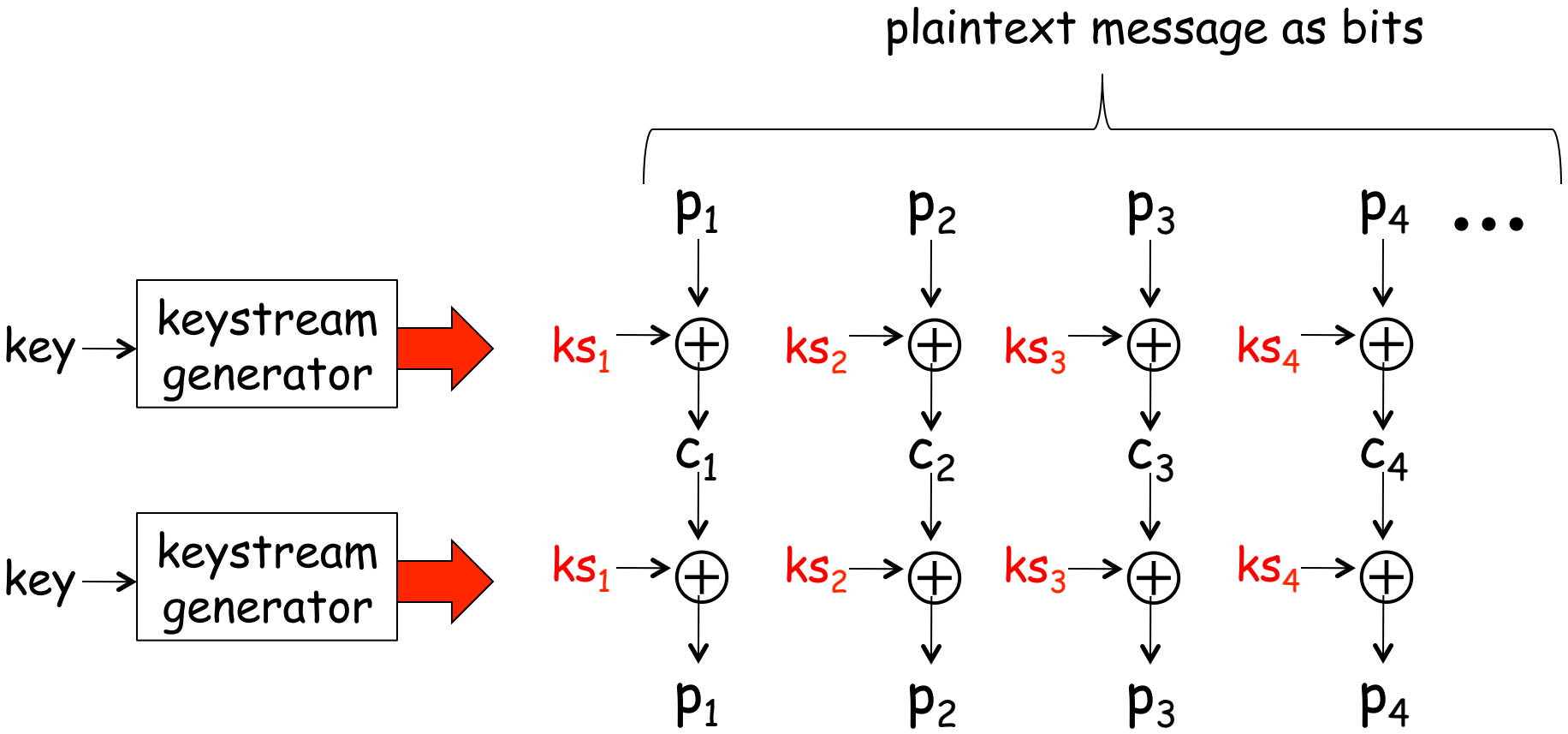
SWNS.com

A chimney in a home in Surrey, England, was found in 1982 to hold the remains of a carrier pigeon bearing a World War II coded message. An effort is now under way to find out what it says.

http://www.nytimes.com/2012/11/02/world/europe/world-war-ii-pigeons-message-a-mystery.html

Stream Cipher

Basic architecture is like one-time pad, except XORing is done with key stream generated by key. E.g., key is seed to psuedo-random number generator (PRNG). Example: RC4.



Evaluating Stream Ciphers

A simple way to encrypt long/infinite streams of data (e.g., a network link).

Security of stream cipher depends entirely on details of keystream generator.

Substitution Cipher

Idea: replace plaintext letters according to alphabet permutation.

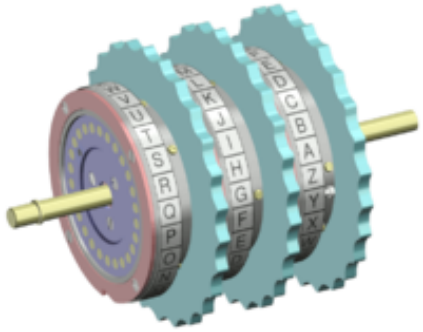
A "key" is one such permutation. E.g.:

plaintext:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
ciphertext:	j	d	u	s	c	l	n	f	r	h	a	g	z	w	k	y	i	p	t	m	x	b	v	q	o	e

Encrypt a message via this permutation. E.g.:

plaintext:	o	n	e	i	f	b	y	l	a	n	d	t	w	o	i	f	b	y	s	e	a
ciphertext:	k	w	c	r	l	d	o	g	j	w	s	m	v	k	r	l	d	o	t	c	j

- Same plaintext at different positions encrypts to same ciphertext.
- Can implement by rotors (e.g. WWII Enigma)
- How many keys are there?



Breaking Substitution Ciphers

There are 26! = 4×10^{26} keys.

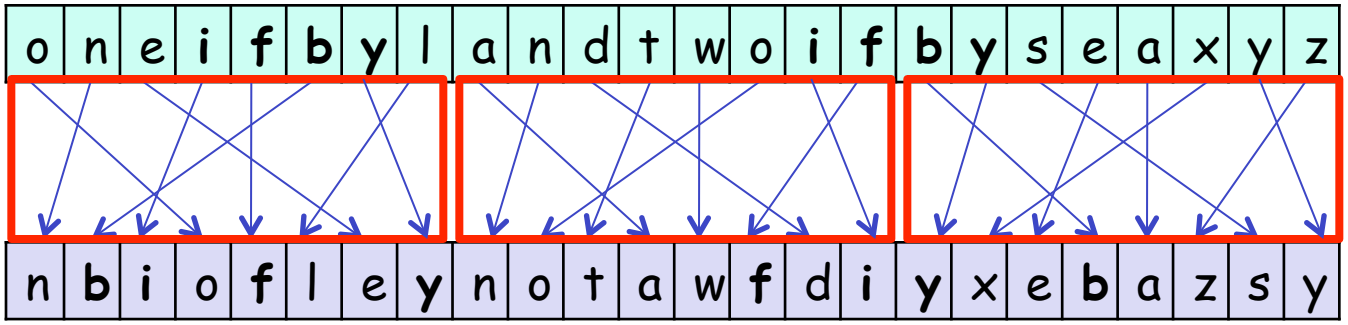
So substitution ciphers must be very secure, right? ☺

☹ Oops! How are substitution ciphers broken?

Transposition Cipher

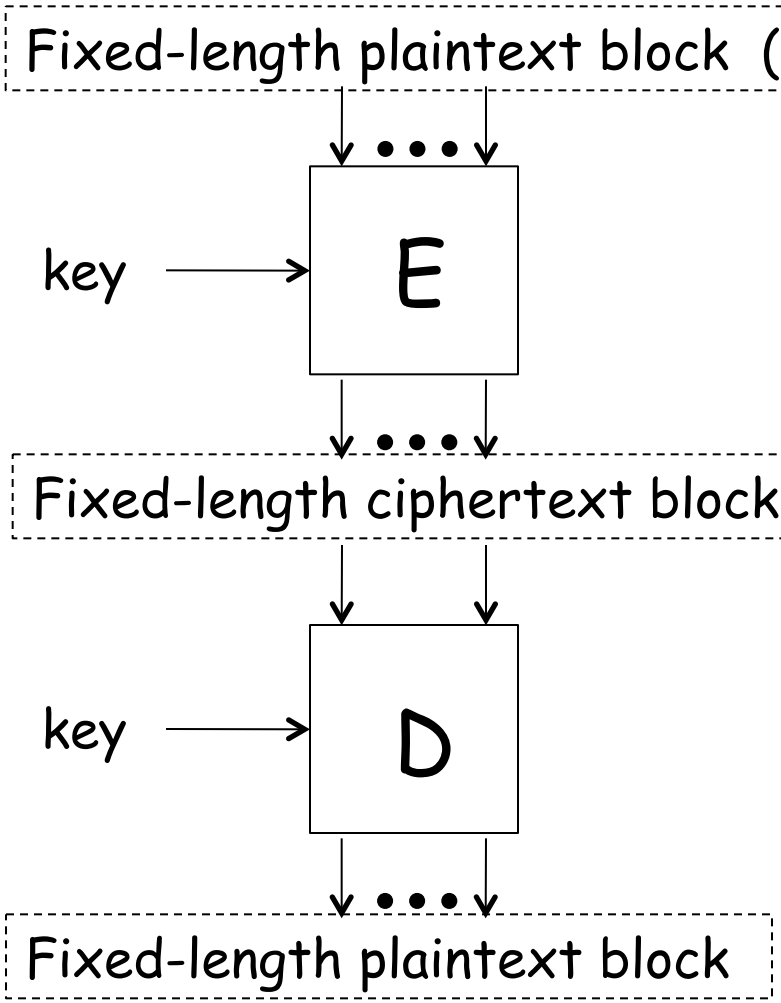
Idea: shuffle chunks of the plaintext message

A "key" is one such shuffling. E.g.:



- Same plaintext at different positions can be shuffled differently.
- May need to "pad" end of message. Pads must be chosen carefully!
- How many keys are there for shuffle blocks of length n?
- How can transposition ciphers be broken?

Block Cipher



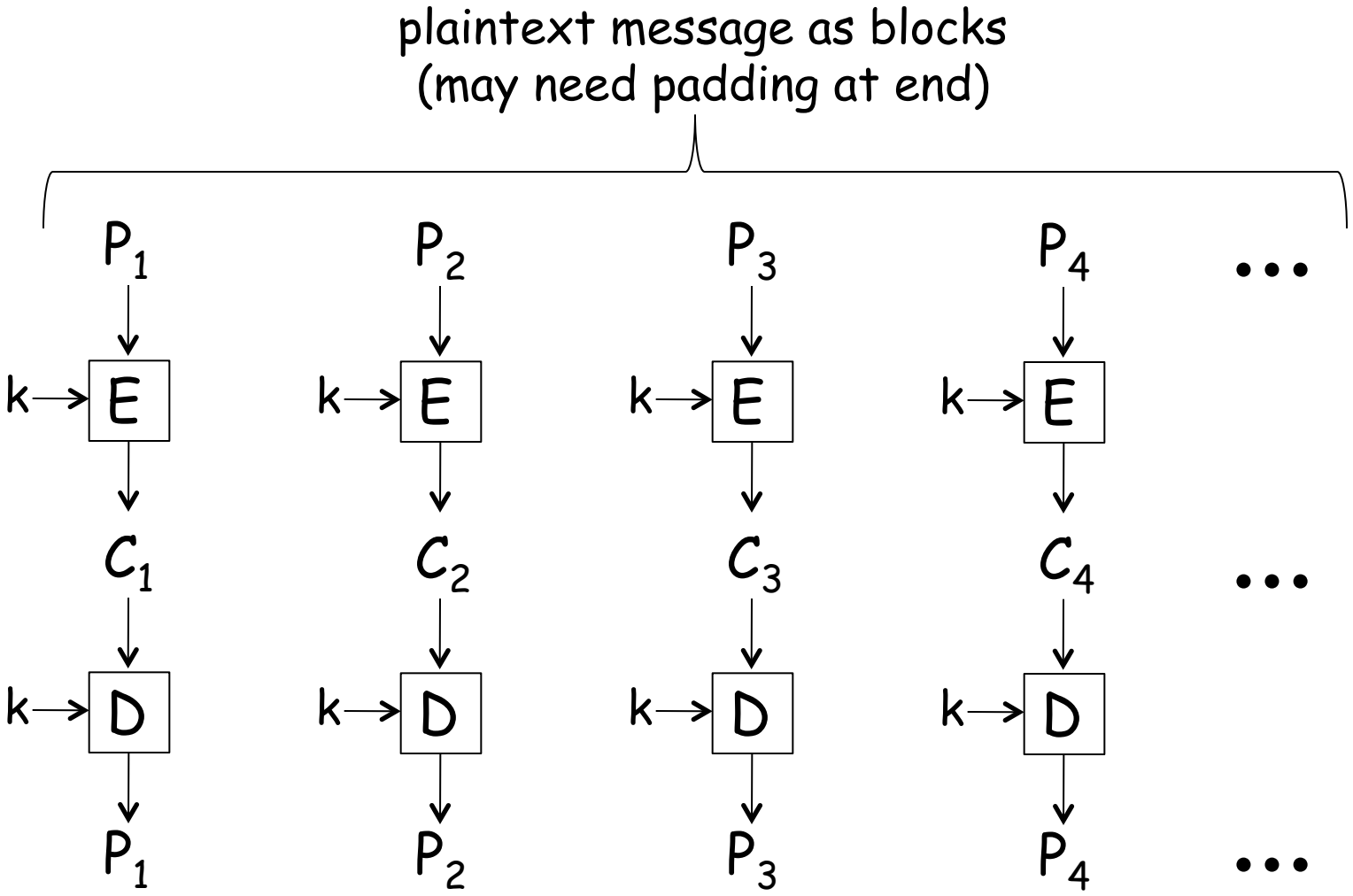
How many different blocks with n bits?

In theory, how many different encryption functions?

In practice far fewer, depending on key size. How many encryption functions for key size k?

Examples: DES, AES. Skipjack, IDEA, Blowfish, RC5

Block Cipher: Electronic Codebook Mode (ECB)



Evaluating ECB

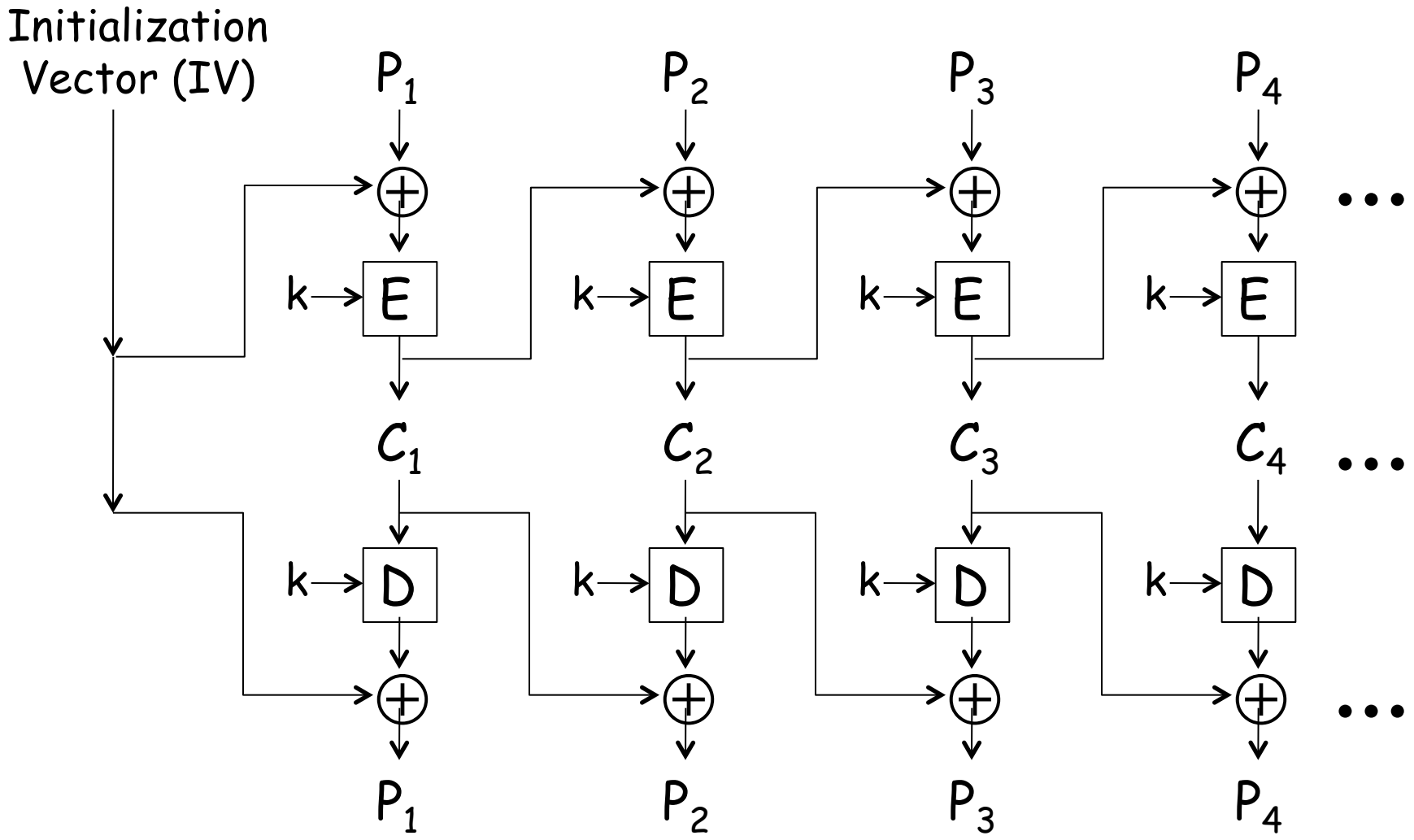
Advantages:

- Easy to parallelize
- Can modify part of encrypted file without re-encrypting whole file.
- Ciphertext bit errors in one block don't propagate to other blocks.

Disadvantages:

- Can make "codebook" of any decrypted blocks.
- Can perform statistical attacks on blocks. Stereotyped beginnings and endings of messages especially problematic.
- Adding or losing a ciphertext bit (synchronization error) throws everything off if no frames.
- Block replay: Mallory can replace, remove, repeat, interchange blocks (e.g. modify bank transfers to other accounts to move money to his account).

Block Cipher: Cipher-Block Chaining Mode (CBC)



Evaluating CBC

Why Initialization Vectors?

Same message (or message prefix) won't yield same ciphertext for different IVs (like password hashing salt). IVs chosen randomly.

Advantages of CBC:

Resistant to "codebooks", statistical attacks, and block replay.

Disadvantages of CBC:

- Inherently sequential.
- Error problems:
  - Changed bit in ciphertext block garbles current plaintext block and changes one bit of subsequent block .
  - Adding/removing single bit in ciphertext block garbles rest of plaintext message.

Data Encryption Standard (DES)

- In 1972, National Bureau of Standards issued request for standard crypto algorithm.
- Data Encryption Standard (DES) was adopted as federal standard in 1977 and ANSI standard in 1981.
- Grew out of IBM Lucifer system and evaluated by the National Security Agency (NSA). People worried that NSA reduced key size from 128 to 56 bits and may have introduced a trapdoor.
- DES is a block cipher --- maps 64-bit plaintext block to 64 bit ciphertext block controlled by 56-bit key. Uses 16 rounds of the same substitution/permutation operation.

Encryption/Decryption with openssl Command

```
[cs342@puma] cat secret.txt
One if by land, two if by sea.

[cs342@puma] openssl enc -des -nosalt -pass pass:foobar -in secret.txt -out
secret.enc

[cs342@puma] cat secret.enc
S\3725\337*|^T\341\345^^\366\316\207\370\261\351d\371^D^A#
\245^V1>\240Gy\230\256

[cs342@puma] openssl enc -d -des -nosalt -pass pass:foobar -in secret.enc
One if by land, two if by sea.
```

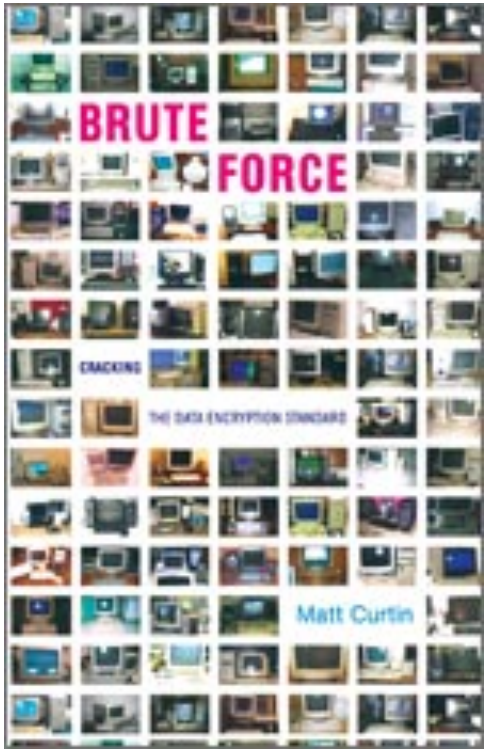
Brute Force Attacks on DES

Although denied by US govt. (esp. NSA) people suspected DES was breakable by brute force (i.e., try all possible keys) with enough computing power.

Shown in series of RSA-Labs-sponsored challenges to break DES keys:

- Jun. 1997: 140 days by DESCHALL project (distributed.net), as described in Matt Curtin's *Brute Force: Cracking the Data Encryption Standard*
- Feb. 1998: 39 days by distributed.net
- Jul. 1998: 56 hours by Electronic Frontier Foundation's (EFF) "Deep Crack" machine
- Jan. 1999, key broken in 22 hours and 15 minutes by Deep Crack + distributed.net

Moral: Key size matters!



Other Block Ciphers

National Institute of Standards and Technology (NIST) requested proposals for new encryption standard in 1997; winner in 2000 was dubbed *Advanced Encryption Standard (AES)* . Minimum of 128-bit keys; up to 256-bit keys.

Other block ciphers:

- Skipjack (Clipper chip/Fortezza program, 80-bit keys)
- Triple-DES (TDES) = three rounds of DES with 3 different keys; 2*56 = 112 effective bits of security with 3*56 = 168-bit keys.
- IDEA (128-bit keys)
- Blowfish (Schneier, up to 448-bit keys).
- RC5 (Rivest, parameterized over key & block size)

Breaking Cryptography (S&M Ch. 8)

Brute force attack: Try all possible keys. Feasability depends on key size, available computrons.

Information theoretic attacks: Letter frequency information can break shift & substitutions cipher quickly. (To reduce redundancy of messages, some crypto implementations first compress message.)

Algorithm/implementation attacks: take advantage of details in algorithm or its implementation – e.g. choice of randomness, padding functions . Examples:
- Netscape SSL "random" session key easily guessable with time/process info.
- Kerberos v.4 DES 56-bit session key only had 20 bits of info.

Side channel attacks: determine key from timing, memory usage, power, electromagnetic field, etc.

Keyjacking: Hijack calls to cryptographic API.

Other approaches we' ve seen: *information leakage* (find keys on stack, in memory pages, etc.) and *social engineering* (shoulder surfing, dumpster diving, etc.)

Cryptanalytic Attack Classification

Add ciphertext-only: given only encrypted messages, deduce messages/key.

known-plaintext: given plaintext messages & encryptions, deduce key/algorithm

chosen-plaintext: deduce key from black-box encrypter.

chosen-ciphertext: deduce key from black-box decrypter.

purchase-key/rubber-hose: bribe/threaten key holder until s/he gives up key

Change of Focus: Message Authentication

Confidentiality: No one can read M except Alice and Bob (e.g., whispering, locked box);

Authentication: Bob knows M comes from Alice, not someone masquerading as Alice (e.g., wax seal, watermark paper, signature);

Integrity: Bob can verify that M has not be altered after it was sent by Alice (e.g., tamperproof packaging);

Nonrepudiation: Alice should not be able to falsely deny sending M.

encryption/ decryption

message authentication, digital signatures

Message Authentication via Digital Signatures

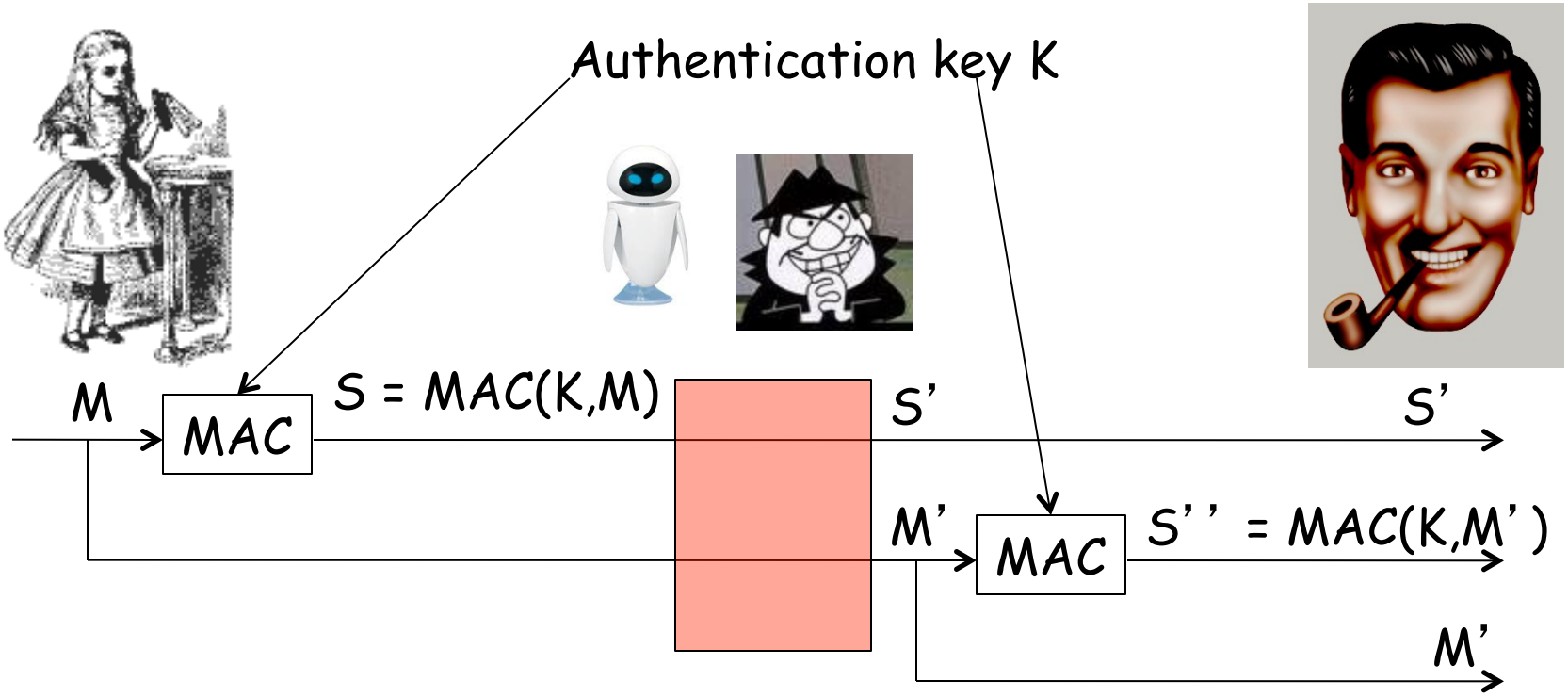
With encryption scenario, nothing prevents Eve from recording and replaying messages and Mallory from modifying messages (e.g. splicing parts of different messages together).

Want a way to determine that an plaintext message is authentic -- it's really from who it says it's from, and has not been changed.

If Alice sends Bob (unencrypted) M, want it accompanied with a **digital signature $S_{A,M}$** that has the following properties:

- **Authenticity**: Bob confident M is from Alice (only she signs $S_{A,M}$).
- **Integrity** of signed document. Changing M invalidates $S_{A,M}$
- **Unforgeability**: no one but Alice knows how to sign $S_{A,M}$.
- **Unreusabilty**: $S_{A,M}$ depends on M, so can't be used with another msg.
- **Nonrepudiation** of signature: Alice can't claim she didn't send M.

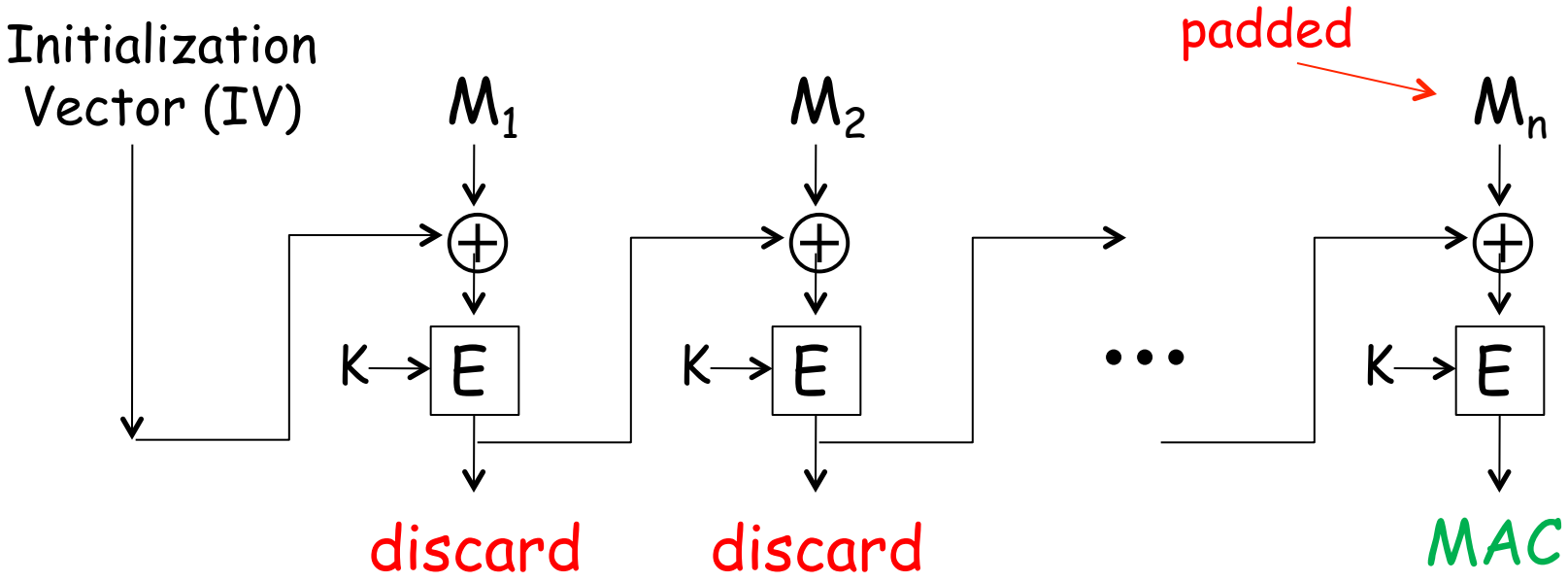
Message Authentication Code (MAC)



- For untampered message, M = M' and S = S' = S''. Otherwise, Bob thinks message has been changed and/or is not from Alice.
- Replay still a problem, but can be solved with sequence #s/timestamps.
- MAC provides no confidentiality by itself. M could be plaintext.
- MAC can be combined with encryption, but authentication key should be unrelated to encryption key.

CBC-MAC

Idea: Break message into n fixed-sized blocks (padding if necessary) and encrypt with block cipher in CBC mode. MAC is final cipher block.

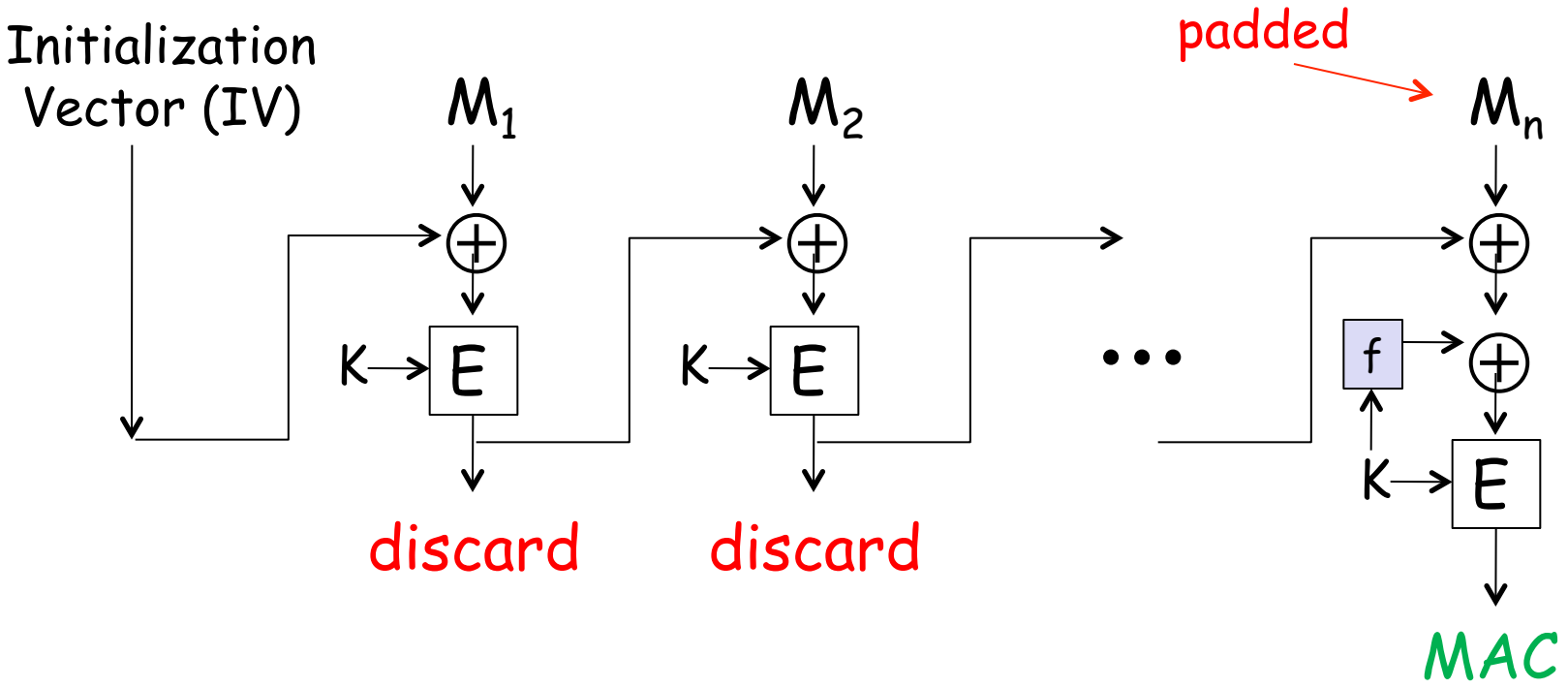


Ferguson, Schneier, and Kohno (FSK) warn:

- Never use same key for encryption and authentication.
- Collision attacks limit security to half the length of block size.
- Not recommended, because difficult to use correctly.

CMAC

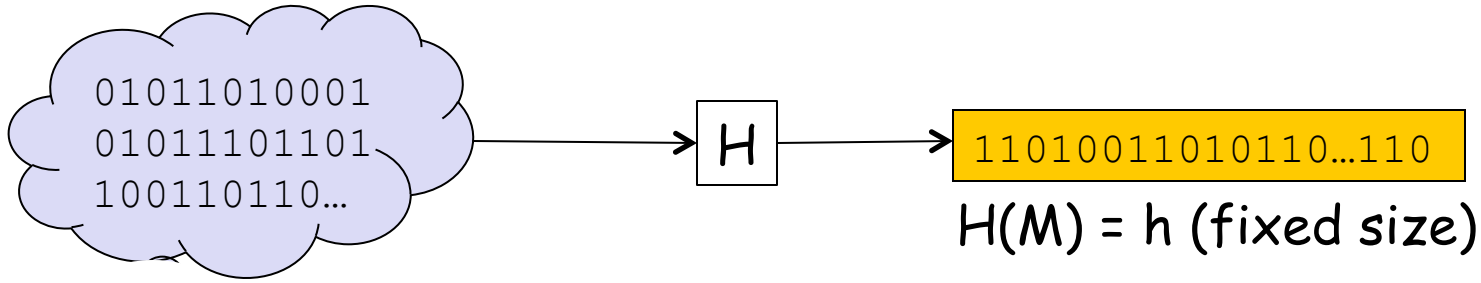
Idea: Like CBC-MAC, but XOR key-dependent value in input to final block to disrupt some CBC-MAC attacks



Cryptographic Hash Functions

A cryptographic hash function H(M) returns a fixed-length hash value h for any size message M. This value h is known as a ***cryptographic checksum***, ***fingerprint***, ***message digest***.

Well see they' re very handy to use in MAC.s



Hash Function	Bits in Fingerprint
MD5 (Message Digest, Rivest)	128
SHA-1 (Secure Hash Algorithm, NSA)	160
SHA-2 (NSA)	224, 256, 384, 512
SHA-3 (NIST)	under design

Desirable Crypto Hash Function Properties

One-way (a.k.a. Pre-image Resistance):
•Easy to compute H(M) = h.
•Given h', hard to find an M' such that H(M') = h' (even though there are ∞ such values, by the pigeon-hole principle).
•Implies that, given M, hard to find M' such that H(M) = H(M').

Collision Resistance:
•Practically difficult to find (M_1, M_2) that collide: $H(M_1) = H(M_2)$.

Seemingly Random Behavior:
•H behaves like a random mapping: changing a single bit in M should change about half the bits in H(M), in an unpredictable way

Note: do note confuse cryptographic hash functions with much simpler hash functions used in hash tables!

Hashing Examples Using openssl

[cs342@puma] echo "One if by land, two if by sea." | openssl dgst -md5
fe66cbf9d929934b09cc7e8be890522e

[cs342@puma] echo "One if by land, two if by sea." | openssl dgst -md5 -c
fe:66:cb:f9:d9:29:93:4b:09:cc:7e:8b:e8:90:52:2e

[cs342@puma] echo "One if by land, two if by sea" | openssl dgst -md5 -c
78:4b:61:4a:66:98:17:82:18:d9:25:ca:c9:64:c5:56

[cs342@puma] echo "One if by land, Two if by sea." | openssl dgst -md5 -c
96:07:e8:69:cd:97:59:98:ad:21:8e:46:a8:c0:4f:0e

[cs342@puma] echo "One if by land, two if by sea." | openssl dgst -sha1 -c
28:47:d1:e5:9a:96:83:bf:2f:2a:91:b8:f3:ec:21:63:d3:be:5a:6b

[cs342@puma] echo "One if by land, two if by sea" | openssl dgst -sha1 -c
24:e2:d1:19:44:c2:17:49:1f:d8:9c:23:d0:9d:d2:d9:87:87:11:f1

More Hashing Examples

[cs342@puma ~] echo "One if by land, two if by sea." | openssl dgst -sha256 -c
67:82:97:b4:e2:4f:95:28:b7:f1:cf:37:dc:8b:49:83:3f:94:d6:45:50:eb:1c:4f:79:86:
56:0a:59:8e:e1:ed

[cs342@puma ~] echo "One if by land, two if by sea" | openssl dgst -sha256 -c
94:09:b6:88:06:44:df:e8:47:28:e2:9c:5e:99:0c:16:76:5c:ad:1d:32:36:25:ef:2b:c3:
0e:d8:7a:ed:38:95

[cs342@puma ~] echo "One if by land, two if by sea." | openssl dgst -sha512 -c
93:fc:e3:a3:07:6a:90:ec:51:29:be:71:da:bf:47:dd:d0:67:ed:89:c6:b4:f9:27:ba:f6:
c8:8c:a1:78:d2:53:ac:92:bc:3d:a5:52:06:98:d5:40:14:9f:2e:ad:fe:ab:55:ae:6f:d7:
67:cf:1e:b4:85:0a:01:9c:78:8f:97:22

[cs342@puma ~] echo "One if by land, two if by sea" | openssl dgst -sha512 -c
8c:89:d6:ad:9e:30:09:53:00:70:bd:a0:4c:1a:62:34:7c:5e:f2:a3:c0:25:02:99:e2:7a:
89:b0:ac:2f:a5:fc:7c:44:72:a7:a6:69:75:45:c9:3f:18:f9:12:e0:a7:50:bf:73:c4:f8:
61:42:fd:90:78:3d:06:28:7b:f0:48:8c

Hashes in Practice: Passwords

One-wayness of hashes is useful for storing passwords.

Example: suppose user gdome has password foogle.

- Crypt (DES) style /etc/shadow entry for gdome (first 2 chars are "salt"):

  gdome:MATk8HeMV.5yk:14151:0:99999:7:::

  [cs342@puma ~] openssl passwd -salt Wf foogle
  rNR3vCevfquRw

  [cs342@puma ~] openssl passwd -salt Wf foogle

  Wf1x7fz3kpBrg

  [cs342@puma ~] openssl passwd -salt MA foogle
  MATk8HeMV.5yk

- MD5 style /etc/shadow entry for gdome ("salt" between 2nd and 3rd $)

  gdome:1TCClG4D0$GTUC6geaYRIq8BnhIo5n81:14151:0:99999:7:::

  [cs342@puma ~]$ openssl passwd -1 -salt TCClG4D0 foogle

  1TCClG4D0$GTUC6geaYRIq8BnhIo5n81

Hashes in Practice: Safe Downloads

Safe downloads from untrusted sites (as long as have file hash from trusted site).

Can use hashes to guarantee file integrity.

E.g., http://www.safer-networking.org/en/download/index.html

```
[lynux@localhost ~]$ openssl dgst -md5 Desktop/spybotsd160.exe
MD5(Desktop/spybotsd160.exe)= 0e7fbf50f87b3b7c384a2471154a7558
```

Hashes in Practice: Intrusion Detection

- Intrusion detection: use hash (possibly combined with key) to fingerprint all important system files.

Can use hashes to guarantee file integrity.

E.g., http://www.safer-networking.org/en/download/index.html

```
[lynux@localhost ~]$ openssl dgst -md5 Desktop/spybotsd160.exe
MD5(Desktop/spybotsd160.exe)= 0e7fbf50f87b3b7c384a2471154a7558
```

Hashes Aren't Digital Signatures!

One-way hashes can be used for integrity in some cases (e.g., code fingerprint from ``reputable'' website), but by themselves they're not suitable for signing messages from Alice to Bob.

Why?

Hash-Based MACs

A MAC combine hashes with keys to so that only key-holders can authenticate. Useful for authenticating files between users and determining if user files have changed

Basic idea: send pair <M, S> of message M and signature S, where the S is calculated from M and key K. Some possible signatures:

- encrypt hash value of M with key: S = E(K, H(M)).
- hash encrypted value of M: S = H(E(K, M)).
- hash result of concatenating key and message: (1) S = H(K @ M) or (2) S = H(M @ K).

Schneier & FSK mention some vulnerabilties of these approaches.

Generic Hash Attacks

Recall: want hash function H to have both the following properties:

Pre-Image Resistance: given h, hard to find M s.t. H(M) = h

Collision Resistance: hard to find M_1 and M_2 s.t. $H(M_1) = H(M_2)$

For *k*-bit hashes:

- given h, about how many messages M do we expect to examine in a brute force attack on h?

- about how many messages M do we expect to examine to find a collision?

To figure these out, let's think about birthdays …

Birthday Problems

The following two problems are very different:

1.Alice is in a room of n people. What's the smallest n for which there's a 50% probability that someone else shares Alice's birthday?

2. **(Birthday Paradox)** There are n people in a room. What's the smallest n for which there's a 50% probability that at least one pair share the same birthday?

Birthday Problem #1: Details

Solution to Birthday Problem #1:

Let B(n) = at least one of n people shares Alice's birthday
and NB(n) = not one of the n people shares Alice's birthday

Then p(B(n)) + p(NB(n)) = 1, so p(B(n)) = 1 – p(NB(n)).

$p(NB(n)) = (364/365)^n$

Note: *expected* number of people to have first match is 365.

Consequence: for k-bit hash, expect to find a message with a given hash value h after 2^k hashes.

n	p(NB(n))	p(B(n))
50	.872	.128
100	.760	.240
150	.663	.337
200	.578	.422
250	.504	.496
253	**.500**	**.500**
300	.439	.561
400	.334	.666
500	.254	.746
750	.128	.872
1000	.064	.936

Birthday Paradox: Details

Solution to Birthday Problem #2 (Birthday Paradox)

Let B(n) = at least two of n people share same birthday
and NB(n) = no two people share the same birthday.

Then p(B(n)) + p(NB(n)) = 1, so p(B(n)) = 1 – p(NB(n)).

p(NB(n)) = (365/365)*(364/365)*(363/365)* … * ((366 – n)/365)

= 365! / ((365 – n)! * 365^n)

n	p(NB(n))	p(B(n))
10	.883	.117
15	.747	.253
20	.589	.411
23	**.493**	**.507**
30	.294	.706
40	.109	.891
50	.03	.97
60	.006	.994
70	.001	.999

Birthday Paradox: Results

Suppose a system generates a random sequence of n values in [1 .. L].

There are (n * (n-1))/2 pairs of sequence elements (ignoring order).

There is a 1/L chance that any pair has equal values.

So the probability of a collision is (n * (n-1))/2L ~ $n^2/2L$.

When is the collision probability about 50%?

What does this imply about the number of messages that need to be examined to find collisions for a hashing function with a k-digit fingerprint?

Birthday Paradox: Practical Consequences

MD5 hash (128 bits)

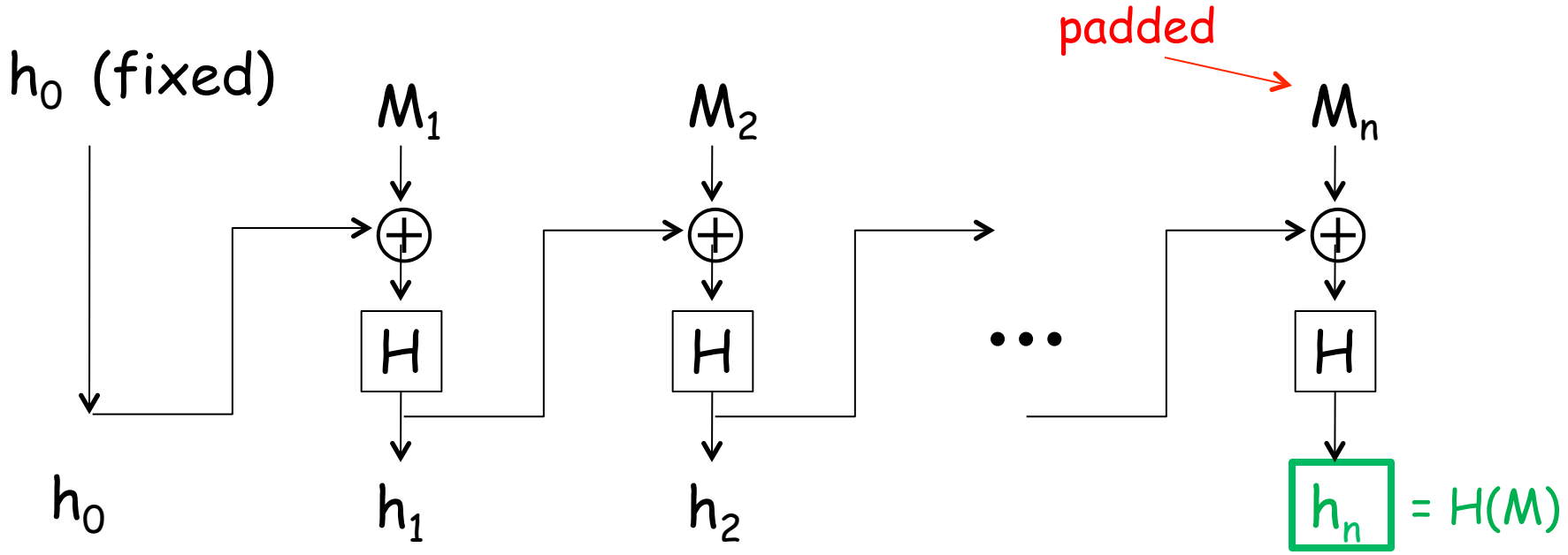
- Expect to find collisions after $2^{64} \sim 18 \times 10^{18}$ messages. Once unthinkably large, but now very thinkable.
- Cryptanalytic advances starting in 2005 allow finding collisions in much fewer than 2^{64} computations.
- "While the existence of such efficent collision finding attacks may not immediately break all uses of MD5, it is safe to say that MD5 is very weak and should no longer be used." (FSK, Ch. 5)

SHA-1 hash (160 bits)

- Expect to find collisions after $2^{80} \sim 1.2 \times 10^{24}$ messages. Within the realm of thinkability.
- Algorithm details make it possible to finding collisions in much fewer than 2^{80} computations.
- "It is no longer safe to trust SHA-1". (FSK, Ch. 5)

Iterative Hash Functions

In practice, a hash function H on fixed size blocks (typically 512 bits) is applied iteratively to message blocks starting with a fixed value.



Properties:

- Easy to implement.
- Can compute as soon as part of message received, so works well on stream of data.
- But, subject to some attacks.

Attacks on Iterative Hash Functions

Length Extension Attack:

- Knowing $H(M_1 \mathbin{@} M_2 \ldots \mathbin{@} M_n)$, we know a lot about $H(M_1 \mathbin{@} M_2 \ldots \mathbin{@} M_n \mathbin{@} M_{n+1})$.

- E.g., Consider $MAC(K,M) = H(K \mathbin{@} M) = h$. Mallory can add an extra block to the end of M without changing h.

Partial-Message Collisions:

- Consider $MAC(K,M) = H(M \mathbin{@} K) = h$, where K is a multiple of a block size. Then if Mallory finds M' such that $H(M') = H(M)$, she can replace M by M' without changing h.

FSK suggest the hash function $H'(M) = H(H(M) \mathbin{@} M)$ to fix these problems.