CSCI E-170 Lecture 03:
Hash functions and Symmetric Ciphers

Simson L. Garfinkel
Center for Research on Computation and Society
Harvard University

October 3, 2005

Administrivia

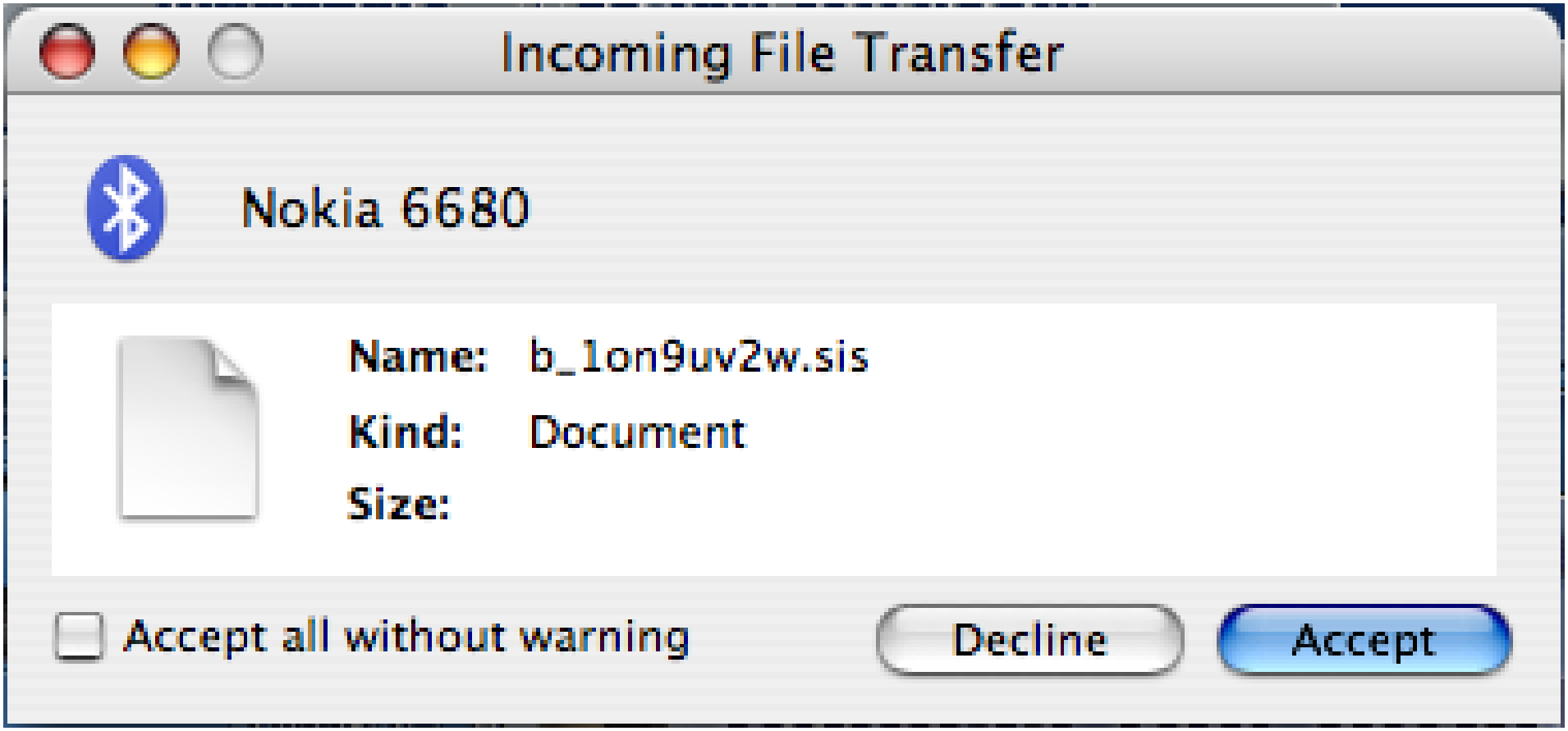
1. LiveJournal — everybody okay?
2. HW1 — Checkpoint
3. Hash Functions
4. Symmetric Ciphers
5. Readings

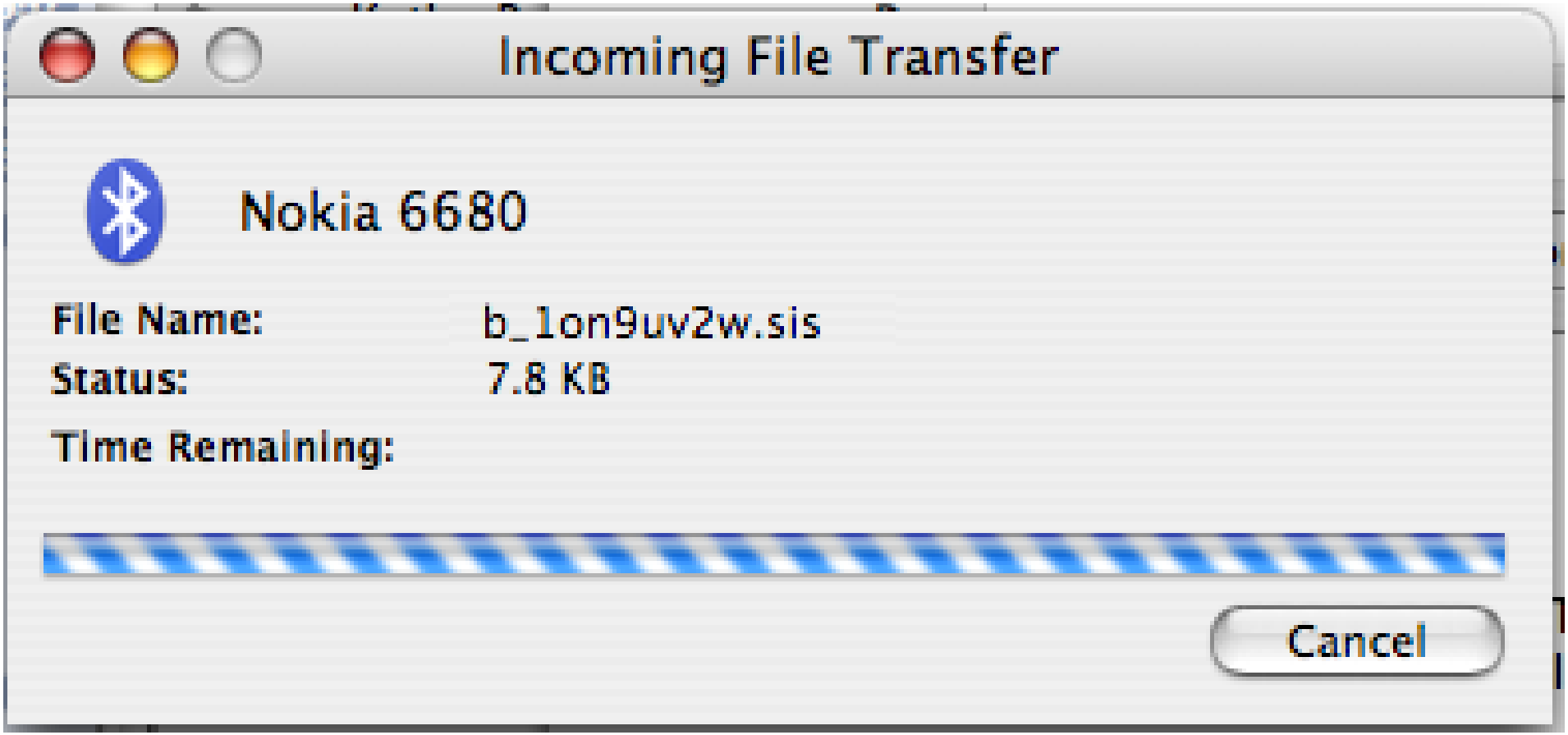
What happened to this lock?

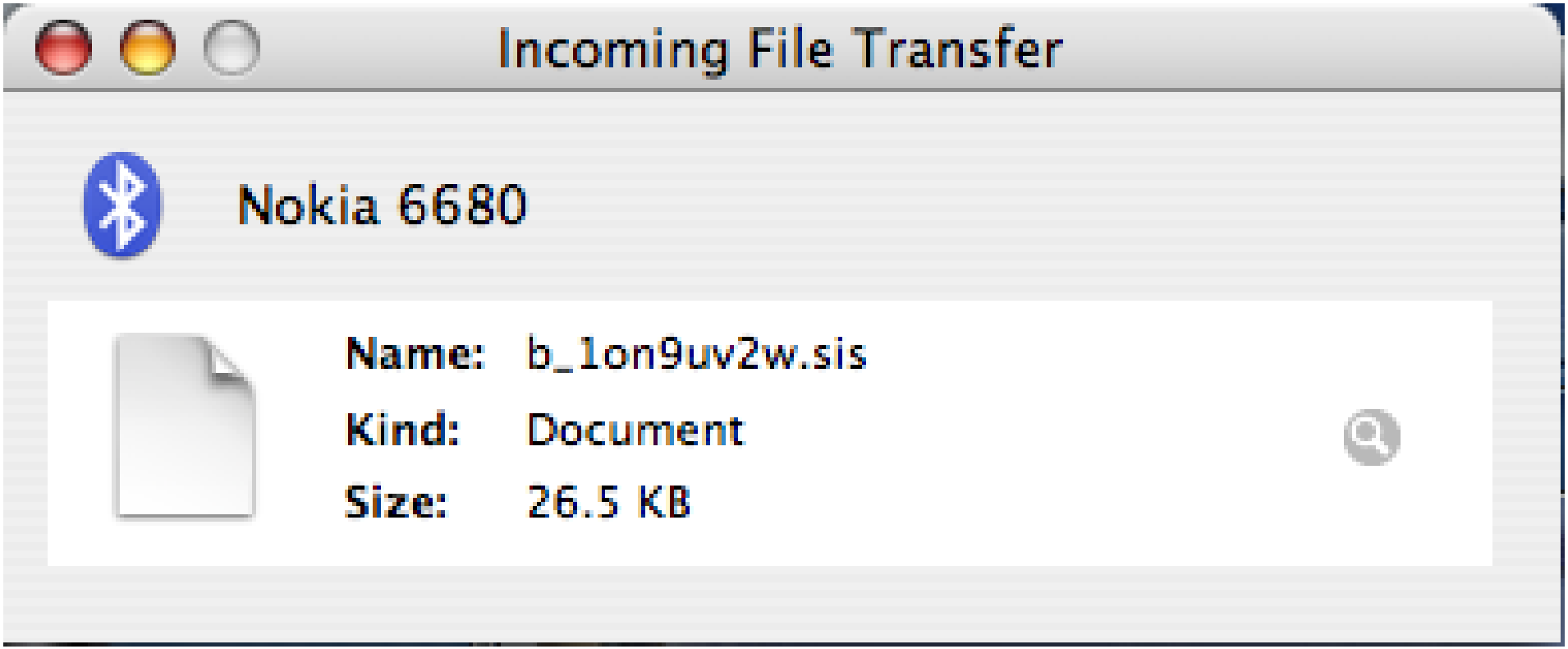
http://theory.csail.mit.edu/~gjw/lock.jpg

LiveJournal

Question: Should we split the discussion into two sections?


Incoming File Transfer
Nokia 6680
Name: b_1on9uv2w.sis
Kind: Document
Size:
Accept all without warning
Decline
Accept


Incoming File Transfer
Nokia 6680
File Name:
b_1on9uv2w.sis
Status:
7.8 KB
Time Remaining:
Cancel

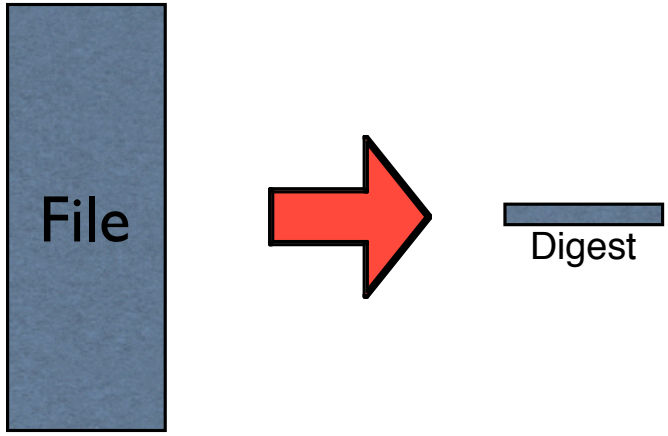


What's going on here?

Hash functions make a “fingerprint” of a file.

Input: 1-264 bytes

Output: 128, 160, 256 or more bits



Also called “message digests” and “one-way functions.

Message Digest Example

Constitution of the United States of America
(In Convention, September 17, 1787)

Preamble
We the people of the United States, in order to form a more perfect union, establish justice, insure domestic tranquility, provide for the common defense, promote the general welfare, and secure the blessing of liberty to ourselves and our posterity, do ordain and establish the Constitution of the United States of America.

Article I.
Section 1. All legislative powers herein granted shall be vested in a Congress of the United States, which shall consist of a Senate and a House of Representatives.
...

$$C = \tag{1}$$

$$\mathrm{MD5}(C) = \mathrm{bab1c005bad1ac7d58d54d0e5d0e5f3f} \tag{2}$$

$$\mathrm{SHA1}(C) = \mathrm{ff3881c932e7591e674e2d9d772817746e8d983f} \tag{3}$$

The output is sometimes called a "residue."

How to compute a hash function from the command line:

```
% ls -l
-rw-r--r--  simsong  wheel  47990 Const.txt

% md5 Constitution.txt
MD5 (Const.txt) = bab1c005bad1ac7d58d54d0e5d0e5f3f

% sha1 Constitution.txt
SHA1 (Const.txt) = ff3881c932e7591e674e2d9d772817746e8d983

% openssl sha1 < Const.txt
ff3881c932e7591e674e2d9d772817746e8d983f

% openssl sha1 Const.txt
SHA1(Const.txt) =ff3881c932e7591e674e2d9d772817746e8d983f
```

Properties of a good hash function

1. Simple function of input. $H = f(M)$

2. Fast to compute

3. H cannot be predicted from the input

4. Hard or impossible to find $f(M_1) = f(M_2)$
   (two inputs with the same hash)

5. Changing one bit of M changes each bit of output with $p = 0.5$.

Changing one bit of M changes ~~everything~~ lots.

M	$\mathrm{MD5}(M)$
“this is a test”	ff22941336956098ae9a564289d1bf1b
“this is c test”	c5e530b91f5f324b1e64d3ee7a21d573
“this is a test ”	6df4c47dba4b01ccf4b5e0d9a7b8d925

Adding a space, changing a line break, capitalizing a word, changes the output.

There are many different hash functions

Rivest Functions:

- MD2 (RFC 1319, MD4 (RFC 1320), MD5 (RFC 1321)

NIST Functions:

- SHA, SHA-1, SHA-512, SHA-1024

European Function:

- Whirlpool

Other Functions:

- Snerfu, N-Hash, RIPE-MD, HAVAL

Residue sizes of popular hash functions

Digest	Bits
MD5	128
SHA-1	160
SHA-256	256
SHA-512	512
Whirlpool	256(check)

Of course, a hash function seeks to do the impossible

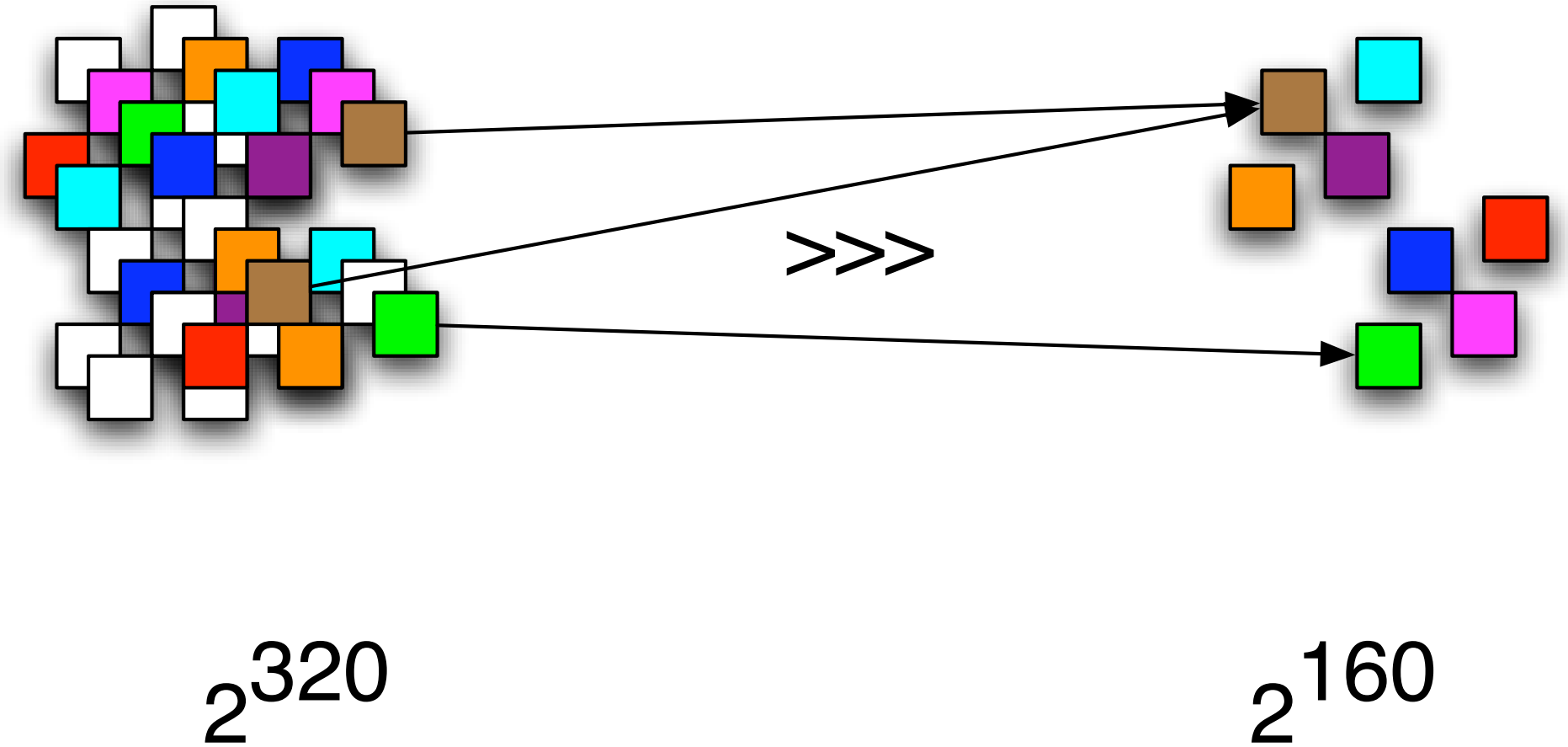
Possible SHA-1 hash residues:

$2^{160} =$ 1,461,501,637,330,902,918,203,684,832,716,283,019,
655,932,542,976

Two lines of 80-characters have 160 characters = 320 bits.

$2^{320} =$ 20,815,864,389,328,798,163,850,480,654,728,171,077,
230,524,494,533,409,610,638,224,700,807,216,119,346,
720,596,024,478,883,464,648,369,684,843,227,908,562,
015,582,767,132,496,646,929,816,279,813,211,354,641,
525,848,259,018,778,440,691,546,366,699,323,167,100,
945,918,841,095,379,622,423,387,354,295,096,957,733,
925,002,768,876,520,583,464,697,770,622,321,657,076,
833,170,056,511,209,332,449,663,781,837,603,694,136,
444,406,281,042,053,396,870,977,465,916,057,756,101,
739,472,373,801,429,441,421,111,406,337,458,176

The pigeonhole principle says that there *must* be hash collisions.



But 2^{160} is still a very big number.
In practice, collisions are very hard to find.

Being able to find a collision means that the message digest is "broken."

There are two kinds of attacks:

1. *Brute force attack* — hash until you find a match.

2. *Algorithmic attack* — reverse engineer; less hashing.

Best attack: create H for any M.

Acceptable attack: Create M_1 and M_2 such that $f(M_1) = f(M_2)$

MD5 “broken”

“Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD,” Xiaoyun Wang and Dengguo Feng and Xuejia Lai and Hongbo Yu, August 16, 2004

`http://eprint.iacr.org/2004/199/`

The MD5 evidence

```
file1.dat:
  00000000  d1 31 dd 02 c5 e6 ee c4  69 3d 9a 06 98 af f9 5c
  00000010  2f ca b5 87 12 46 7e ab  40 04 58 3e b8 fb 7f 89
  00000020  55 ad 34 06 09 f4 b3 02  83 e4 88 83 25 71 41 5a
  00000030  08 51 25 e8 f7 cd c9 9f  d9 1d bd f2 80 37 3c 5b
  00000040  96 0b 1d d1 dc 41 7b 9c  e4 d8 97 f4 5a 65 55 d5
  00000050  35 73 9a c7 f0 eb fd 0c  30 29 f1 66 d1 09 b1 8f
  00000060  75 27 7f 79 30 d5 5c eb  22 e8 ad ba 79 cc 15 5c
  00000070  ed 74 cb dd 5f c5 d3 6d  b1 9b 0a d8 35 cc a7 e3

  MD5(file1.dat) = a4c0d35c95a63a805915367dcfe6b751

file2.dat:
  00000000  d1 31 dd 02 c5 e6 ee c4  69 3d 9a 06 98 af f9 5c
  00000010  2f ca b5 07 12 46 7e ab  40 04 58 3e b8 fb 7f 89
  00000020  55 ad 34 06 09 f4 b3 02  83 e4 88 83 25 f1 41 5a
  00000030  08 51 25 e8 f7 cd c9 9f  d9 1d bd 72 80 37 3c 5b
  00000040  96 0b 1d d1 dc 41 7b 9c  e4 d8 97 f4 5a 65 55 d5
  00000050  35 73 9a 47 f0 eb fd 0c  30 29 f1 66 d1 09 b1 8f
  00000060  75 27 7f 79 30 d5 5c eb  22 e8 ad ba 79 4c 15 5c
  00000070  ed 74 cb dd 5f c5 d3 6d  b1 9b 0a 58 35 cc a7 e3

  MD5(file2.dat) = a4c0d35c95a63a805915367dcfe6b751
```

Just 6 bits are different.

Finding which 6 bits to change was a challenge.

$512^6 = 2^{54} = 18,014,398,509,481,984$

Documents with Tunable Digests

We the people citizens of the US United States, in order to form make a more perfect union, establish justice, insure domestic tranquility, provide for the common defense, promote the general welfare, and secure the blessing of liberty to ourselves and our posterity children, do ordain and establish the Constitution of the United States of America.

3 choices = $2^3 = 8$ different SHA-1 codes.

This could be done at the bit level, of course.

Just how big is 2^{128}?

$$2^{56} = \text{72 million billion}$$
$$2^{64} = \text{18 billion billion}$$
$$2^{128} = \text{10,790} \ \frac{\text{billion billion}}{sec} \times \text{trillion years}$$

Hash functions have many uses

Integrity

- Verifying downloaded code
- Determine if two files are identical
- Verifying SSL streams

Authentication

- Verifying a shared secret w/o encryption

Misc.

- Conditioning random number generators
- Storing passwords

MD5s for Downloaded Code

The ProFTPD Project: MD5 sums and PGP signatures of release files

http://proftpd.linux.co.uk/md5_pgp.html

pgp md5

ProFTPD

Highly configurable GPL-licensed FTP server software

Current Versions

Current: **1.2.8** [NEWS]
[gz] [bz2]

Candidate: **none**

Newsroom

- News Flashes
- Critical Bugs
- Security Updates
- Contrib module news

Information

- What is ProFTPD?
- Features
- Supported Platforms
- Bug reporting system
- Reporting Guidelines
- Other sites
- Sites powered by

MD5 sums and PGP signatures of release files

MD5 Digest Hashes

```
35e669cb085879eea21c6db9e7af2040 proftpd-1.2.8.tar.bz2
9064ac430730c792b13910bd7c8b2060 proftpd-1.2.8.tar.gz
```

PGP Signatures

proftpd-1.2.8.tar.bz2.asc

```
-----BEGIN PGP SIGNATURE-----
Version: PGP 6.5.8

iQA/AwUAPmQ7bbeOiT+lEZdqEQIHPgCcCG5NzYoXijvheNnhsCg7Kzca4hQAoKXL
RrZOR5LvwF3oFLU6t/CuoRta
=RPaN
-----END PGP SIGNATURE-----
```

proftpd-1.2.8.tar.gz.asc

```
-----BEGIN PGP SIGNATURE-----
Version: PGP 6.5.8

iQA/AwUAPmQ7ZbeOiT+lEZdqEQJJ6QCeJcBgl19C3ErrshM3j7z5K049UU8AoKHG
L79MJrvRvQ0oiECdtGbuRfwe
=u41Z
-----END PGP SIGNATURE-----
```

ProFTPD Verification

```
ftp> get proftpd-1.2.8.tar.gz
local: proftpd-1.2.8.tar.gz remote: proftpd-1.2.8.tar.gz
227 Entering Passive Mode (81,223,20,36,149,0).
150 Opening BINARY mode data connection for proftpd-1.2.8.tar.gz (966281
bytes)
226 Transfer complete.
966281 bytes received in 00:16 (56.25 KB/s)
ftp> quit
221 Goodbye.
% md5 proftpd-1.2.8.tar.gz
MD5 (proftpd-1.2.8.tar.gz) = 9064ac430730c792b13910bd7c8b2060
%
```

MD5 sums and PGP signatures of release files

MD5 Digest Hashes

```
35e669cb085879eea21c6db9e7af2040 proftpd-1.2.8.tar.bz2
9064ac430730c792b13910bd7c8b2060 proftpd-1.2.8.tar.gz
```

Class Discussion:

What are the practical implications of MD5 being “broken?”

Is it really “broken?”

Storing passwords with hashes

Instead of storing the password, store the hash of the password.

Stealing passwords now requires “cracking” the password file.

Unix originally used a DES-based hash, now it uses an MD5 hash.

```
gigawalt:fURfuu4.4hYOU:129:129:Walter Belgers:/home/gigawalt:/bin/csh

root:$1$zlC9.Vfl$9rXSaQqe1HWDaNNOSTJzh.:0:0::0:0:Nitroba Root$:/root:/
```

Unix passwords are stored with a “salt.”

Old style:

```
gigawalt:ba4TuD1iozTxw:129:129:Walter Belgers:/home/gigawalt:/bin/csh
```

password	foo
salt	ba
hashed password	ba4TuD1iozTxw

FreeBSD:

```
root:$1$zlC9.Vfl$9rXSaQqe1HWDaNNOSTJzh.:0:0::0:0:Nitroba Root$:/root:/
```

algorithm	1
salt	zlC9.Vfl
hashed password	9rXSaQqe1HWDaNNOSTJzh.

What’s the point of the salt?

MACs and HMACs use a shared secret for message authentication

MAC	“Message Authentication Code”
HMAC	“Keyed Hashing for Message Authentication” (RFC 2104)

`http://www.ietf.org/rfc/rfc2404.txt`

`http://www.cs.ucsd.edu/users/mihir/papers/hmac.html`

IETF HMAC (RFC 2104) is a standard way of turning any hash function into a keyed MAC.

$$\mathrm{HMAC}(f, K, M) = f(K \otimes (\mathrm{0x5c})^{64} \cdot f(K \otimes (\mathrm{0x36})^{64} \cdot M))$$

More complicated than concatenating the key and taking the hash, but more secure!

HMACs are widely used in network protocols.

- SNMPv3
- BGP
- IPsec Authentication Header and Encapsulating Security Payload
- RFC 3567 Intermediate System to Intermediate System (IS-IS) Cryptographic Authentication

There are three common ways to represent a hash:

binary	8XM-v"0M-,<M-^Q_0^LfC^RM-F?				
hex	3858f62230ac3c915f300c664312c63f				
base64	0Fj2IjCsPJFfMAxmQxLGPw				
octal	034130	173042	030254	036221	057460
	006146	041422	143077	005000	

MD5 API: Perl

```
% man Digest::MD5
  # Functional style
  use Digest::MD5  qw(md5 md5_hex md5_base64);

  $digest = md5($data);
  $digest = md5_hex($data);
  $digest = md5_base64($data);

  # OO style
  use Digest::MD5;

  $ctx = Digest::MD5->new;
  $ctx->add($data);
  $ctx->addfile(*FILE);

  $digest = $ctx->digest;
  $digest = $ctx->hexdigest;
  $digest = $ctx->b64digest;
```

Using the perl API:

```
#!/usr/bin/perl

use Digest::MD5 qw(md5);
use strict;

open J,$ARGV[0] || die "Cannot open $ARGV[0],";
my $ctx = Digest::MD5->new;
$ctx->addfile(*J);
print "md5($ARGV[0]) = ",$ctx->hexdigest,"\n";
```

Using the python MD5 API

```
#!/usr/bin/python

import md5
import sys

m = md5.new()
data = open(sys.argv[1],"r").read()
m.update(data)
print "md5(%s) = %s" % (sys.argv[1],m.hexdigest())
```

Note: be careful not to call this file md5.py!

Using the perl HMAC API

```
NAME
  Digest::HMAC_MD5 - Keyed-Hashing for Message Authentication

SYNOPSIS
  # Functional style
  use Digest::HMAC_MD5 qw(hmac_md5 hmac_md5_hex);
  $digest = hmac_md5($data, $key);
  print hmac_md5_hex($data, $key);

  # OO style
  use Digest::HMAC_MD5;
  $hmac = Digest::HMAC_MD5->new($key);

  $hmac->add($data);
  $hmac->addfile(*FILE);

  $digest = $hmac->digest;
  $digest = $hmac->hexdigest;
  $digest = $hmac->b64digest;
```

OpenSSL is a popular open-source crypto library.

OpenSSL includes:

- Berkeley-style license
- Written in C
- Most crypto algorithms
- Full S/MIME implementation
- Key management
- SSL

`http://www.openssl.org`
/usr/include/openssl/

OpenSSL MD5 API

```
#include <openssl/md5.h>

unsigned char *MD5(const unsigned char *d,
                   unsigned long n,
                   unsigned char *md);

void MD5_Init(MD5_CTX *c);
void MD5_Update(MD5_CTX *c, const void *data,
                   unsigned long len);
void MD5_Final(unsigned char *md, MD5_CTX *c);
```

OpenSSL SHA-1 API

```
#include <openssl/sha.h>

unsigned char *SHA1(const unsigned char *d,
                    unsigned long n,
                    unsigned char *md);

void SHA1_Init(SHA_CTX *c);
void SHA1_Update(SHA_CTX *c, const void *data,
                    unsigned long len);
void SHA1_Final(unsigned char *md, SHA_CTX *c);
```

The “context” allows multiple hashes to be computed at once.

```
void MD5_Init(MD5_CTX *c);
void SHA1_Init(SHA_CTX *c);
```

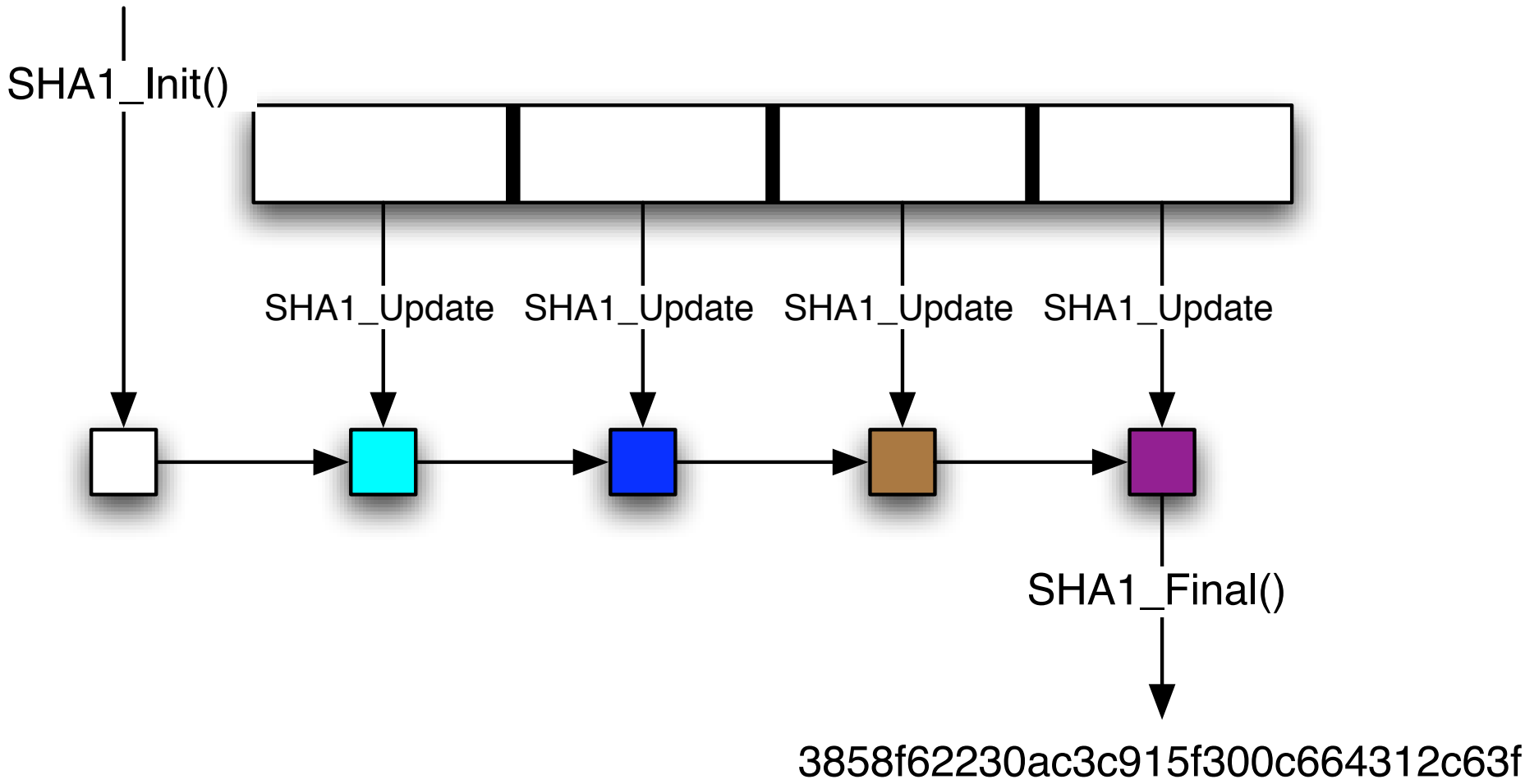
Contexts are used by:

- Hash functions
- Encryption algorithms
- Other large computations requiring “state”

This is similar to instance variables in object oriented programming.

The context holds state between each block that's processed.

char buf[] vs.
unsigned char buf[]



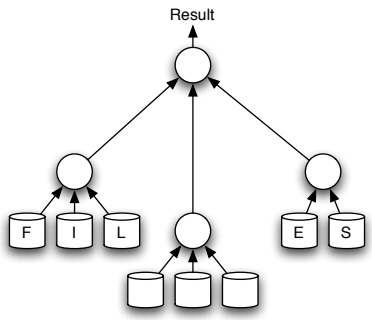
Be careful about `char buf[]` vs. `unsigned char buf[]`

Technically, binary data should be `unsigned char`

Many C routines take char (e.g. `read()`, `write()`).

Expect frequent casts or compiler warnings

Other uses of MACs



Hash Trees — Shurety digital notary

S/KEY

SecureID 

Password Challenge-Response (RFC 1939 APOP)

```
S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C: APOP mrose c4c9334bac560ecc979e58001b3e22fb
S: +OK maildrop has 1 message (369 octets)
```