A

sig fig

Dan Ports <drkp@mit.edu>
6.033 handson #5 - 2004/04/07
Karger TR1

1.1. (2^40 keys / 1 million keys/sec) = ~ 1,099,511 sec = ~ 12.73 days

~ 13 d

1.2. (2^128 keys / 1 million keys/sec) = ~ 1.073 * 10^25 years

1.3. This is many orders of magnitude higher than the age of the universe. ~ 15 OOM

1.4. (2^128 keys / (10^9 processing elements * 10^9 keys/element/sec))
= 1.073 * 10^13 years.
This is still far too long to be effective as a reasonable method for cracking the encryption.

1.5. Turning to Scheme for a solution,

```
1 ]=> (define  (iter time keys rate)
  (let ((new-keys (+ keys (* rate 60 60 24 30 18))))
    (display
     (string-append (number->string time)
                    " months: "
                    (number->string new-keys)
                    " keys"))
    (newline)
    (if (< new-keys (expt 2 128))
        (iter (+ time 18) new-keys (* rate 2)))))

;Value: iter

1 ]=> (iter 0 0 1000000)
0 months: 46656000000000 keys
18 months: 139968000000000 keys
36 months: 326592000000000 keys
54 months: 699840000000000 keys
72 months: 1446336000000000 keys
90 months: 2939328000000000 keys
[...]
1368 months: 705045537999251734688637177600000000 keys
1386 months: 1410091075998503469377279020800000000 keys
1404 months: 2820182151997006938754562707200000000 keys
1422 months: 5640364303994013877509130080000000000 keys
1440 months: 11280728607988027755018264825600000000 keys
1458 months: 22561457215976055510036534316800000000 keys
1476 months: 45122914431952111020073073299200000000 keys

;No value

1 ]=> (/ 1476 12)

;Value: 123
```

OK... I got around 130 yr.

So it takes less than 123 years to break this encryption (less since the last computer system will not be used for the full 18 months).

2.1. dan@AMBULATORY-CLAM:~/sandbox/6.033/handson [5:55pm - 596] echo "Massachusetts Institute of Technology" | openssl sha1
c29518134a6b47563b1e411b7401a52c081996a4

This is the SHA-1 for "Massachusetts Institute of Technology" with the newline.

2.2. There are 2^160 SHA-1 keys, and we'll suppose that each one is

equally likely. Running 'wc /afs/net/admin/hosts/hosts' suggests that there are about 100,000 hosts on the network:

```
dan@AMBULATORY-CLAM:~ [3:56am - 64] wc /afs/net/admin/hosts/hosts
  105168  323794 5819479 /afs/net/admin/hosts/hosts
```

(Of course, many of these hostnames are unused, and other machines might not be listed here. But it gives us a decent first approximation to the order of magnitude.) Sure.

'df -i' indicates that my system has about 6.3 million inodes in use. We'll suppose those are all unique files, and every system has about that many. (These are horrible assumptions to make -- certainly they're not all unique files, and it's hard to estimate the average number of files per host, especially with things like the AFS servers.) It seems a bit high, but we might as well make a conservative approximation, and hopefully the order of magnitude is at least close. Then the probability is: Reasonable enough.

((100,000 * 6,300,000) / 2^160) = ~ 1/(231984386877921098127569021066076697), which is a horribly small probability. ack! give me 10^ Well, most of that time was process creation overhead...

2.3. Executing 'openssl sha1' 100 times on a file containing "Massachusetts Institute of Technology\n" took 6.76 seconds on my system. This is about 15 executions per second. Assuming the distribution is uniform, it would take 2^160 attempts to find another file whose hash code matched this one. So the time required is (2^160 / 15) seconds. This is on the order of 10^47 seconds. I didn't start this handson until the day before it was due, so I don't have the 10^47 seconds for the calculation available. I thought about asking for an extension. *chuckle*

3.1. Signing encrypts a hash of the message with the sender's private key so that any recipient can verify the integrity and authenticity of the message. Sealing encrypts the message with the recipient's public key, so that only the recipient will be able to decode it. The unencrypted message is included in plaintext in the signed message. good

3.2.

```
dan@clamshell:~ [4:20pm - 154] gpg --list-keys E3232682
pub  1024D/E3232682 2004-03-27 MIT 6.033 - Spring 2004 - Hands on #5 (This key
is for hands-on #5) <6.033-tas@mit.edu>
sub  1024g/68853FD5 2004-03-27 [expires: 2004-05-26]
```

3.3. The first message was modified after being signed:

```
dan@clamshell:/tmp [4:23pm - 161] fetch -o - http://web.mit.edu/6.033/handout
s/handson/gpg-message1.txt.asc | gpg --verify
Receiving - (276 bytes): 100%
276 bytes transferred in 0.0 seconds (409.60 kBps)
gpg: Signature made Sat Mar 27 14:09:34 2004 PST using DSA key ID E3232682
gpg: BAD signature from "MIT 6.033 - Spring 2004 - Hands on #5 (This key is f
or hands-on #5) <6.033-tas@mit.edu>"
```

3.4. The PGP signatures must hold a hash of the message encrypted with the sender's private key.

3.5. The key is apparently signed by Simson Garfinkel:

```
dan@clamshell:/tmp [4:43pm - 167] gpg --check-sigs E3232682
pub  1024D/E3232682 2004-03-27 MIT 6.033 - Spring 2004 - Hands on #5 (This ke
y is for hands-on #5) <6.033-tas@mit.edu>
sig!3        E3232682 2004-03-27   MIT 6.033 - Spring 2004 - Hands on #5 (This
 key is for hands-on #5) <6.033-tas@mit.edu>
sig!3        903C9265 2004-03-27   Simson L. Garfinkel <simsong@vineyard.net>
sig!3        D1BA664D 2004-03-27   Simson L. Garfinkel <simsong@acm.org>
sub  1024g/68853FD5 2004-03-27 [expires: 2004-05-26]
```

DAM 10/15

sig! E3232682 2004-03-27 MIT 6.033 - Spring 2004 - Hands on #5 (This key is for hands-on #5) <6.033-tas@mit.edu>

✓ We know that the signatures are valid because gpg --check-sigs verifies the signatures. However, we do not know that the signing key is valid; we would need to receive it from a trusted source or follow the web of trust in order to verify that it is in fact Simson Garfinkel's PGP key. right.

3.6. This key claims to be George Bush's PGP key. But anyone could have generated a key with that identity and submitted it to the key server, so we can't trust that it's valid. (In any case, considering how few people actually understand how to use PGP, we certainly can't trust that Bush is among them.) Indeed.

$Id: handson5.txt,v 1.1 2004/04/08 17:06:07 dan Exp $