A Tiny Lecture on the Technology and Legal Dangers of Cryptography

Be scared. Be very scared.

A Tiny Lecture on Cryptography

- Fundamental foundational theory for much of security
- **Cryptography**
  encrypting stuff
- **Cryptosystems**
  encryption algorithms and procedures
- **Cryptanalysis**
  decrypting stuff

The Big Three

- Private Key Cyphers
- Public Key Cyphers
- One-way Hash Functions

- Also: Stream vs. Block Cyphers

Bad Approaches to Crypto

- Security through Obscurity
- “Secret” Systems
- Simple Systems with Little Mixing
    - If you had access, you could crack by feeding in plaintext and comparing with the cyphertext
    - You can even break if you have enough cyphertext examples

Roman Cyphers

- “Super-Duper Secret Coder Ring” cyphers
- Pick a random number N (from 1-26)
- **To Encrypt**
  add N to each letter in your message, mod 26
- **To Decrypt**
  subtract N from each letter, mod 26
- Famous example: **rot13**

One Time Pads

- Alice and Bob each have a large stack of cards with random numbers from 1 to 26 written on each one.
- **To encrypt**
  Alice takes one card per letter and adds them, mod 26.
- **To decrypt**
  Bob takes one card per letter and subtracts them, mod 26
- What's good or bad about this?

Private Key Encryption

- Generalization of Roman Cypher
- Bob and Alice both have a secret key with which they encrypt/decrypt their messages
- No one else has access to the secret key or you're screwed.
- Example use: exchanging secret data

Examples

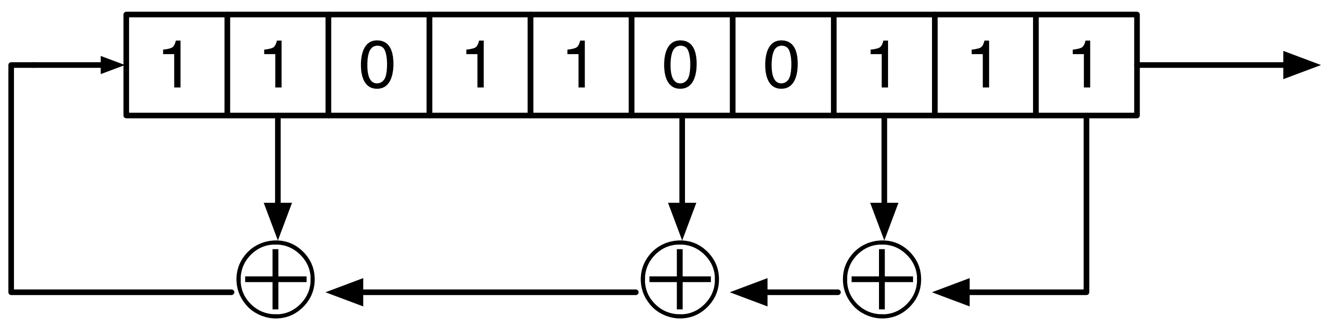
- DES
- GOST
- IDEA
- Blowfish and Twofish
- AES (Rijndael)
- Skipjack

Issues

- The algorithm must be **symmetric**: the encryption function P➔C must have a unique cyphertext C for each plaintext block P, and a unique plaintext block P for each cyphertext C. It's a **bijection**.
- Hardware or Software?
- Stream Cypher or Block Cypher?

Stream Cypher

- Encrypts a stream of data
- Previous stuff that was encrypted may influence how later stuff is encrypted



Block Cypher

- Encrypts a block of data at a time
- The same algorithm is used (and reset) for each block
- Thus two identical blocks will encrypt in the same way
    - Different from what might happen in a stream cypher!

One-Way Hash Functions

- A **hash function** P➔H takes some plaintext P and produces a **hash** H (often a number). Hashes tend to be randomly distributed with regard to the plaintext, and small and easy to compute. *(usually used in "hash tables")*

- A **one-way hash function** is a function P➔H which is easy to do, but such that it's extremely difficulty to compute H➔P

- Issues: it'd be nice to have a function where each P has a unique H

- Example Use: password encryption files

Examples

- MD4
- MD5
- SHA-1, SHA-2 family

Public Key Encryption

- Fundamental problem with private-key encryption: each party (Alice and Bob) have to have the key.

- This means if Alice creates the key, she has to **get the key to Bob safely.** This is close to impossible on the internet.

Public Key Encryption

- Alice makes a **private** key $\mathbf{A_1}$ with a special encryption function using that key P_{A_1}➔C, such that Bob can only decrypt the message with a different **public** key $\mathbf{A_2}$, that is, C_{A_2}➔P.
- And: P_{A_2}➔C is only decrypted with C_{A_1}➔P.
- And: even given A_2, figuring out A_1 is **very hard!**

Public Key Encryption

- Now Alice can post the public key A_2 on a bulletin board for all to see.
- If Bob wants to send Alice a message, he just encrypts using the public key and sends it to Alice.
- Only Alice has the private key, so only Alice can decrypt it.

Proving Identity

- We can also use public key encryption to prove the identity of Alice.
- Alice says: “I'm Alice!”.
- Bob says “Prove it. Encrypt this message (say, the current date).”
- Alice encrypts with the private key.
- Bob is able to decrypt with the public key. It must be Alice!
- A way to defeat this: the **man in the middle attack**.

Issues

- Public Key Encryption is slow.
  - So usually what Alice and Bob do is: use public key encryption to exchange a private key K which Alice just made up, and then Alice and Bob use private key encryption with K to exchange their secret info.
- Public Key Encryption relies on “Hard Problems” (np-complete or worse): bin-packing, prime number factorization

Examples

- Diffie-Hellman
- RSA

What Happened With DVDs?

- The DVD encryption algorithm: secret key encryption where the data can be decrypted with M different keys.
- Each manufacturer gets one key.
- If a manufacturer lets that key get public, no further DVDs will be made that can be unlocked with that key, and the manufacturer is ruined.
- What could go wrong?

DVD John

- It turns out that the keys are easy to figure out once you have one of them.
- DVD Jon (Lech Johnasen) cracked ’em all.
- www.cs.cmu.edu/~dst/DeCSS/Gallery

Smallest C implementation

```c
/*     efdtt.c     Author:  Charles M. Hannum <root@ihack.net>             */
/*                                                                         */
/*     Thanks to Phil Carmody <fatphil@asdf.org> for additional tweaks.    */
/*                                                                         */
/*     Length:  434 bytes (excluding unnecessary newlines)                 */
/*                                                                         */
/*     Usage is:  cat title-key scrambled.vob | efdtt >clear.vob           */

#define m(i)(x[i]^s[i+84])<<
unsigned char x[5],y,s[2048];main(n){for(read(0,x,5);read(0,s,n=2048);write(1,s
,n))if(s[y=s[13]%8+20]/16%4==1){int i=m(1)17^256+m(0)8,k=m(2)0,j=m(4)17^m(3)9^k
*2-k%8^8,a=0,c=26;for(s[y]-=16;--c;j*=2)a=a*2^i&1,i=i/2^j&1<<24;for(j=127;++j<n
;c=c>y)c+=y=i^i/8^i>>4^i>>12,i=i>>8^y<<17,a^=a>>14,y=a^a*8^a<<6,a=a>>8^y<<9,k=s
[j],k="7Wo~'G_\216"[k&7]+2^"cr3sfw6v;*k+>/n."[k>>4]*2^k*257/8,s[j]=k^(k&k*2&34)
*6^c+~y;}}
```

ASCII Art Version

```
/*     efdtt.c     Author:  Charles M. Hannum <root@ihack.net>             */
/*                                                                         */
/*     Thanks to Phil Carmody <fatphil@asdf.org> for additional tweaks.    */
/*                                                                         */
/*     DVD-logo shaped version by Alex Bowley <alex@hyperspeed.org>        */
/*                                                                         */
/*     Usage is:  cat title-key scrambled.vob | efdtt >clear.vob           */


#define m(i)(x[i]^s[i+84])<<

                   unsigned char x[5]      ,y,s[2048];main(
                   n){for( read(0,x,5     );read(0,s ,n=2048
                            ); write(1   ,s,n)        )if(s
                  [y=s      [13]%8+20]  /16%4  ==1      ){int
                  i=m(      1)17  ^256 +m(0)   8,k      =m(2)
                  0,j=      m(4)   17^ m(3)   9^k*     2-k%8
                  ^8,a     =0,c    =26;for   (s[y]    -=16;
                 --c;j  *=2)a=     a*2^i&    1,i=i /2^j&1
                 <<24;for(j=        127;     ++j<n;c=c>
                                    y)
                                    c

                           +=y=i^i/8^i>>4^i>>12,
                  i=i>>8^y<<17,a^=a>>14,y=a^a*8^a<<6,a=a
                >>8^y<<9,k=s[j],k         ="7Wo~'G_\216"[k
                 &7]+2^"cr3sfw6v;*k+>/n."[k>>4]*2^k*257/
                       8,s[j]=k^(k&k*2&34)*6^c+~y
                                  ;}}
```

A Small Perl implementation

```
#!/usr/bin/perl
# 472-byte qrpff, Keith Winstein and Marc Horowitz <sipb-iap-dvd@mit.edu>
# MPEG 2 PS VOB file -> descrambled output on stdout.
# usage: perl -I <k1>:<k2>:<k3>:<k4>:<k5> qrpff
# where k1..k5 are the title key bytes in least to most-significant order

s''$/=\2048;while(<>){G=29;R=142;if((@a=unqT="C*",_)[20]&48){D=89;_=unqb24,qT,@
b=map{ord qB8,unqb8,qT,_^$a[--D]}@INC;s/...$/1$&/;Q=unqV,qb25,_;H=73;O=$b[4]<<9
|256|$b[3];Q=Q>>8^(P=(E=255)&(Q>>12^Q>>4^Q/8^Q))<<17,O=O>>8^(E&(F=(S=O>>14&7^O)
^S*8^S<<6))<<9,_=(map{U=_%16orE^=R^=110&(S=(unqT,"\xb\ntd\xbz\x14d")[_/16%8]);E
^=(72,@z=(64,72,G^=12*(U-2?0:S&17)),H^=_%64?12:0,@z)[_%8]}(16..271))[_]^((D>>=8
)+=P+(~F&E))for@a[128..$#a]}print+qT,@a}';s/[D-HO-U_]/\$$&/g;s/q/pack+/g;eval
```

Haiku

(I abandon my
exclusive rights to make or
perform copies of

this work, U. S. Code
Title Seventeen, section
One Hundred and Six.)

Muse! When we learned to
count, little did we know all
the things we could do

some day by shuffling
those numbers: Pythagoras
said "All is number"

long before he saw
computers and their effects,
or what they could do

by computation,
naive and mechanical
fast arithmetic.

It changed the world, it
changed our consciousness and lives
to have such fast math

available to
us and anyone who cared
to learn programming.

Now help me, Muse, for
I wish to tell a piece of
controversial math,

for which the lawyers
of DVD CCA
don't forbear to sue:

that they alone should
know or have the right to teach
these skills and these rules.

(Do they understand
the content, or is it just
the effects they see?)

And all mathematics
is full of stories (just read
Eric Temple Bell);

and CSS is
no exception to this rule.
Sing, Muse, decryption

once secret, as all
knowledge, once unknown: how to
decrypt DVDs.

Arrays' elements
start with zero and count up
from there, don't forget!

Integers are four
bytes long, or thirty-two bits,
which is the same thing.

To decode these discs,
you need a master key, as
hardware vendors get.

(This is a "player
key" and some folks other than
vendors know them now.

If they didn't, there
is also a way not to
need one, to start off.)

You'll read a "disk key"
from the disc, and decrypt it
with that player key.

You'll read a "title
key" for the video file
that you want to play.

With the disk key, you
can decrypt the title key;
that decrypts the show.

Here's a description
of how a player key will
decrypt a disk key.

You need two things here:
An encrypted disk key, which
is just six bytes long.

(Only five of those
are the _key itself_, because
"zero" marks the end.

So that's five real bytes,
and eight times five is forty;
in the ideal case,

forty bits will yield
just short of two trillion
possible choices!

Ian Goldberg once
recovered a key that long
in seven half-hours.

But his office-mate
David Wagner points out that
it's _impossible_

to achieve what the
DVD CCA seems
to want to achieve,

even by making
the key some reasonable,
"adequate" key-length:

There's no way to write
a "secure" software player
which contains the key

and runs on PCs,
yet somehow prevents users
from extracting it.

If the player can
decrypt, Wagner has noted,
users can learn how.)

This is a pointer,
"KEY", to those bytes, and when we're
done, they'll be clear-text.

Oh, the other thing!
Called "im", a pointer to six
bytes: a player key.

(Now those six bytes, the
DVD CCA says
under penalty

of perjury, are
its trade secret, and you are
breaking the law if

you tell someone that,
for instance, the Xing player
used the following:

Eighty-one; and then
one hundred three -- two times; then
two hundred (less three);

two hundred twenty
four; and last (of course not least)
the humble zero.)

We will use these few
internal variables:
t1 through t6,

unsigned integers.
k, pointer to five unsigned
bytes. i, integer.

So here's how you do
it: first, take the first byte of
im -- that's byte zero;

OR that byte with the
number 0x100
(hexadecimal --

that's two hundred and
fifty-six to you if you
prefer decimal).

Store the result in
t1. Take byte one of im.
Store it in t2.

Take bytes two through five
of im; store them in t3.
Take its three low bits

(you can get them by
ANDing t3 with seven);
store this in t4.

Double t3, add
eight, subtract t4; store the
result in t3.

Make t5 zero.
Now we'll start a loop; set i
equal to zero.

i gets values from
zero up to four; each time,
do all of these steps:

Use t2 for an
index into Table Two:
find a byte b1.

Use t1 for an
index into Table Three:
find a byte b2.

Take exclusive OR
of b1 with b2 and
store this in t4.

Shift t1 right by
a single bit (like halving);
store this in t2.

Take the low bit of
t1 (so, AND it with one),
shift it left eight bits,

then take exclusive
OR of that with t4; store
this back in t1.

Use t4 for an
index into Table Four:
find a byte and store

it back in t4.
Shift t3 right by three bits,
take exclusive OR

of this with t3,
shift this right by one bit, and
take exclusive OR

of this with t3,
shift this right by eight bits, and
take exclusive OR

of this with t3,
shift this right by five bits, and
(No exclusive OR!

Orange you glad I
didn't say banana?) take
the low byte (by AND

with two hundred and
fifty-five); now store this
into t6. Phew!

Shift t3 left eight
bits, take OR with t6, and
store this in t3.

Use t6 for an
index into Table Four:
find a byte and store

it in t6. Add
t6, t5, t4; store
the sum in t5.

Take t5's low byte
(AND t5 with two hundred
fifty five) to put it

in the ith byte of
the vector called k. Now shift
t5 right eight bits;

store the result in
t5 again. Now that's the
last step in the loop.

No sooner have we
finished that loop than we'll start
another; no rest

for the wicked nor
those innocents whom lawyers
serve with paperwork.

Reader! Think not that
technical information
ought not be called speech;

think not diagrams,
schematics, tables, numbers,
formulae -- like the

terrifying and
uniquely moving, though cliche,
Einstein equation

"Energy is just
the same as matter, but for
a little factor,

speed of light by speed
of light, and we are ourselves
frozen energy."

Einstein's formula
to convert from joules into
kilogram-meters

squared per second squared,
for all its power, uses
just five characters.

But Einstein wrote to
physicists: formal, concise,
specific, detailed.

And sometimes we write
to machines to teach them how
tasks are carried out:

and sometimes we write
to our friends to show a way
tasks are carried out.

We write precisely
since such is our habit in
talking to machines;

we say exactly
how to do a thing or how
every detail works.

The poet has choice
of words and order, symbols,
imagery, and use

of metaphor. She
can allude, suggest, permit
ambiguities.

She need not say just
what she means, for readers can
always interpret.

Poets too, despite
their famous "license" sometimes
are constrained by rules:

How often have we
heard that some strange twist of plot
or phrase was simply

"Metri causa", for
the meter's sake, solely done
"to fit the meter"?

Programmers' art as
that of natural scientists
is to be precise,

complete in every
detail of description, not
leaving things to chance.

Reader, see how yet
technical communicants
deserve free speech rights;

see how numbers, rules,
patterns, languages you don't
yourself speak yet,

still should in law be
protected from suppression,
called valuable speech!

Ending my appeal
on that note, I will describe
the second loop. Store

T-Shirt

- Is this protected speech?

Protocols

- Built On Top of Crypto. Mechanisms for:
  - Proving who you are, and that you've obeyed certain rules
  - Exchanging information in secret
- Examples:
  - Authentication Schemes
  - GPS Encryption
  - HTTPS, SSL, etc.