Secrets & Lies, Knowledge & Trust. (Modern Cryptography)

COS 116, Spring 2010
Adam Finkelstein

Cryptography

|kripˈtägrəfē| noun

the art of writing or solving codes.

- Ancient ideas (pre-1976)
- Complexity-based cryptography (post-1976)

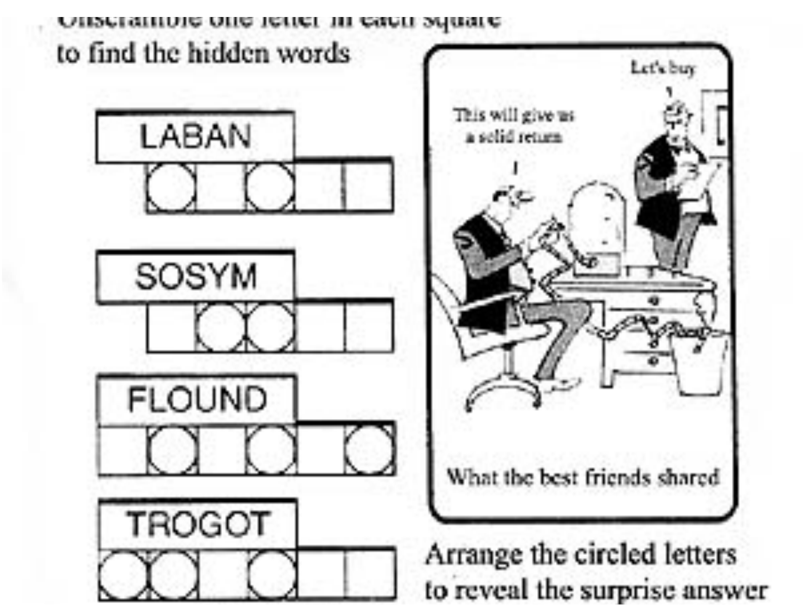
Basic component of Digital World; about much more than just encryption or secret writing.

Main themes of today’s lecture

- Creating problems can be easier than solving them
- Seeing information vs. making sense of it
- Role of randomness in the above
- Two complete strangers exchange secret information

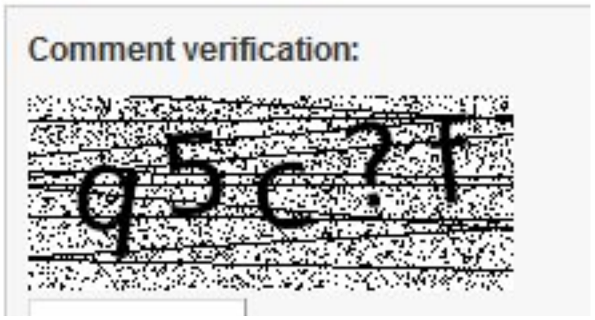
Theme 1: Creating problems can be easier than solving them

Example:


Unscramble one letter in each square
to find the hidden words
LABAN
SOSYM
FLOUND
TROGOT
Let's buy
This will give us a solid return
What the best friends shared
Arrange the circled letters
to reveal the surprise answer

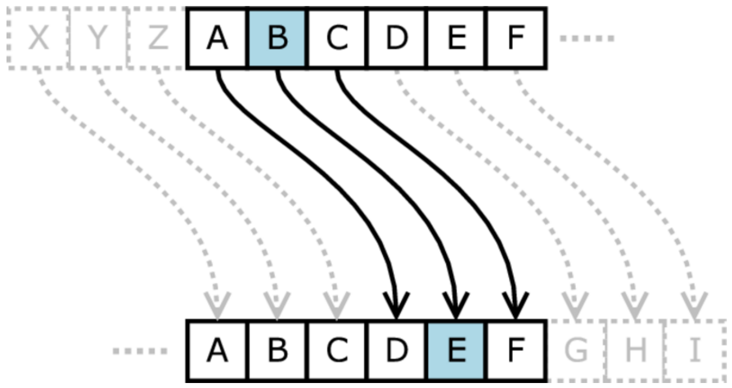
(Aside: This particular problem is trivial for computers!)

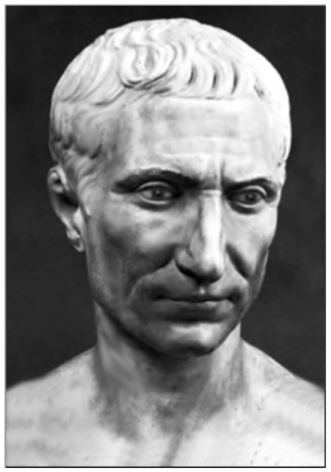
Reminiscent of something similar that is hard for current computers?



Letter scrambling: ancient cryptographic idea

Example 1: "Caesar cipher" (c. 100BC)





Example 2: Cipher used in conspiracy plot involving Queen Mary of Scots, 1587

From Discovery Channel, Apr 17 2006

Mafia Boss's Messages Deciphered

- "Boss of bosses" Bernardo Provenzano, captured after 40 years
- Sent "pizzini" (little messages on scraps of paper) using variant of Caesar cipher
- "...I met 512151522 191212154 and we agreed that we will see each other after the holidays...,"
- 5 = B, 12 = I, 15 = N, etc.

"It will keep your kid sister out, but it won't keep the police out." - Bruce Schneier (Cryptographer)

Letter scrambling (cont.)

- Example 3: Enigma
  - Used by Nazi Germany (1940's)
  - Broken by British (Turing), Polish
  - "Won us the war." – Churchill

Moral: Use of computer necessitates new ideas for encryption.

Integer factoring

Easy-to-generate problem

- Generation

  Pick two 32-digit prime numbers *p, q*, and multiply them to get *r* = *pq*

Hard to solve

- Factoring problem

  Given *r*: find *p* and *q*

Suggest an algorithm?
Running time?

Status of factoring

Despite many centuries of work, no efficient algorithms.

Believed to be computationally hard, but remains unproved ("almost–exponential time")

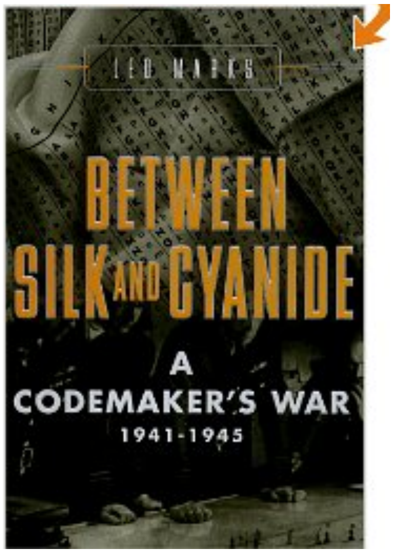
You rely on it every time you use e-commerce (coming up)

Note: If quantum computers ever get built, this may become easy to solve.

- Theme 2:
  Seeing information vs. making sense of it

- Theme 3:
  Role of randomness.

Simple example that illustrates both:
one-time pad ("daily codebook.")



Random source hypothesis

- Integral to modern cryptography

$\longrightarrow$ 0110101010011010011011101010010010001…

- We have a source of random bits
- They look completely unpredictable
- Possible sources:
  Quantum phenomena,
  timing between keystrokes, etc.

One-time pad (modern version)

- Goal: transmit *n*-bit message

- One-time pad: random sequence of *n* bits (*shared* between sender and receiver)

Using one-time pad

- Encryption:
  Interpret one-time pad as "noise" for the message
  - 0 means "don't flip"
  - 1 means "flip"

- Example:

Encryption

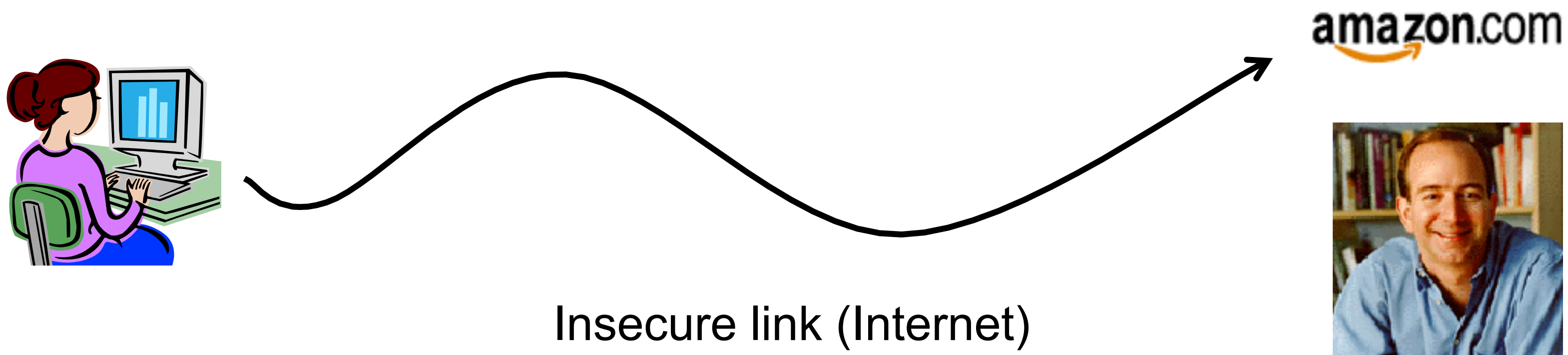
Message	0110010
Pad	1011001
Encrypted	1101011

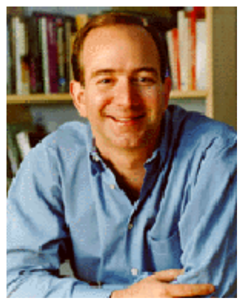
Decryption

Encrypted	1101011
Pad	1011001
Message	0110010

Musings about one-time pad

- Incredibly strong security:
  encrypted message “looks random” …
  equally likely to be encryption of *any* *n*-bit string



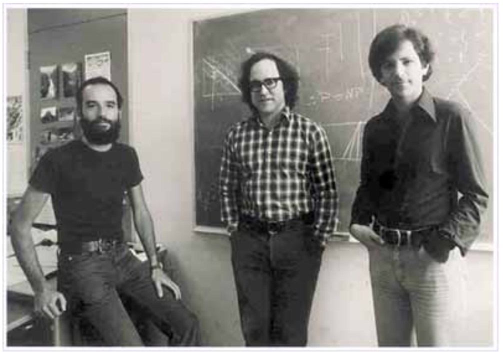

(Jeff Bezos ’86)

- How would you use one-time pad?
- How can you and Amazon agree on a one-time pad?

Theme: How perfect strangers can send each other encrypted messages.

Powerful idea: public-key encryption

- Diffie-Hellman-Merkle [1976]



- Rivest, Shamir, Adleman [1977]

Public-key cryptography

c = Encrypt(m, K_{pub})

$$\xrightarrow{\hspace{10em}}$$

amazon.com

Message m

Public key K_{pub}
(512 bit number, publicly available, e.g. from Verisign Inc)

Private key K_{priv}
(512-bit number, known only to Amazon.)

m = Decrypt(c, K_{priv})

- **Important**: encryption and decryption algorithms are *not* secret, only private key!

Public-key encryption at a conceptual level

- "Box that clicks shut, and only Amazon has the key to open it."

01011
amazon.com
01011

- Example: Key exchange [Diffie-Hellman]
  - User generates random string ("one-time pad")
  - Put it in box, ship it to Amazon
  - Amazon opens box, recovers random string

Public-Key Encryption at a mathematical level (RSA version)

Random Source Hypothesis!

Key generation: Pick random primes p, q.

Let $N = p ¢ q$

Find k that is not divisible by p, q. (“Public Key”)

Encryption: m   is encrypted as m^k (mod N)

Decryption: Symmmetric to Encryption; use “inverse” of k (this is private key)

“Modular” math

Last theme:

What does it mean to learn nothing?

Suggestions?

One-time pad revisited

m = 100100
pad = 110110

c = 010010 $\longrightarrow$

Eve

c = 010010
pad = 110110
m = 100100

- In what sense did Eve learn nothing about the message?
- Answer 1: Transmission was a sequence of random bits
- Answer 2: Transmission looked like something she could easily have generated herself

$\uparrow$

Eureka! moment for modern cryptography

Zero Knowledge Proofs

[Goldwasser, Micali, Rackoff ’85]

Student

prox card

prox card reader

- Desire: Prox card reader should not store “signatures” – potential security leak
- Just ability to recognize signatures!
- Learn nothing about signature except that it **<u>is</u>** a signature

“ZK Proof”: Everything that the verifier sees in the interaction, it could easily have generated itself.

Illustration: Zero-Knowledge Proof that “Sock A is different from sock B”

Sock A

Sock B

- Usual proof: “Look, sock A has a tiny hole and sock B doesn’t!”
- ZKP: “OK, why don’t you put both socks behind your back. Show me a random one, and I will say whether it is sock A or sock B. Repeat as many times as you like, I will always be right.”
- Why does verifier learn “nothing”? (Except that socks are indeed different.)

Actual ZK Proofs

- Use numbers, number theory, etc.

(From Lecture 1): Public closed-ballot elections

- Hold an election in this room
  - Everyone can speak publicly (i.e. no computers, email, etc.)
  - At the end everyone must agree on who won and by what margin
  - No one should know which way anyone else voted
- Is this possible?
  - Yes! (A. Yao, Princeton)

    "Privacy-preserving Computations" (Important research area)