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

COS 116
4/17/2007
Guest Lecturer: Ari Feldman

Cryptography

- Literally means “hidden writing”
- Really is the making and breaking of systems designed to achieve two goals:
  - **Confidentiality** — Keeping information secret
  - **Integrity** — Ensuring that messages are authentic and preventing undetected modifications to messages

Ancient vs. Modern Crypto

- Ancient ideas (pre-1976)
  - More and more complicated letter scrambling
- Modern cryptography (post-1976)
  - Based on computational complexity — the study of what computers can and can't do efficiently

Terminology

- *cipher* — an encryption method
- *plaintext* — the original message before encryption
- *ciphertext* — the encrypted version of the mesage

Cast of characters

Alice

Bob

Eve (a.k.a. Mallory)

(note the devil horns)

Sending an encrypted message

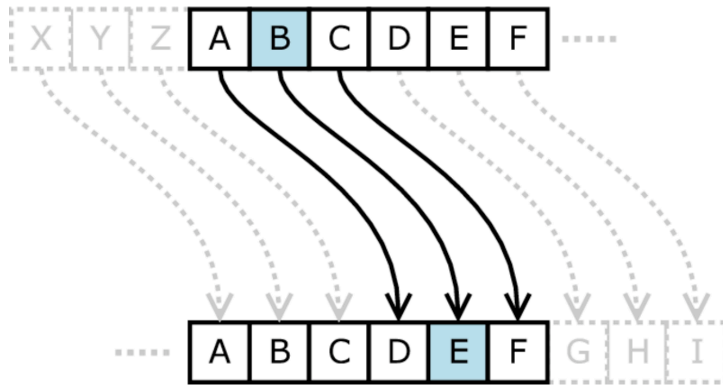
Suppose that Alice wants to send the message

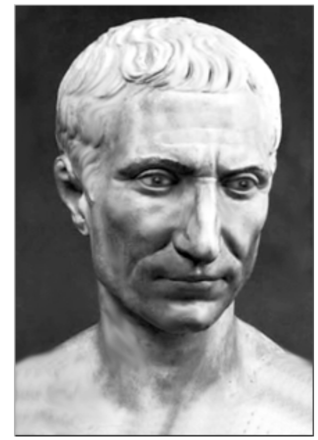
"THE LECTURER SMELLS"

to Bob in encrypted form.

What is the simplest cipher you can think of?

Caesar's Cipher (c. 100BCE)





To encrypt: replace each letter of the plaintext with a letter that is a fixed number of positions further down the alphabet

If Alice shifts by 3 places, then

"THE LECTURER SMELLS" → "WKH OHFWXUHU VPHOOV"

Caesar's Cipher: A closer look

- We can represent each letter A–Z as a number 0–25
- We can represent the size of the shift with a number K which can have values 0–25
- To encrypt, we take each letter L of the original message and calculate:

$$(L + K) \bmod 26$$

- 'mod' gives you the remainder after dividing (e.g. 27 mod 26 = 1)
- 'mod 26' causes numbers greater than or equal to 26 to "wrap around"

K is the "key" — a secret parameter to the cipher that Alice and Bob need to agree on.

Caesar’s Cipher is weak

- Caesar’s Cipher can be broken easily. How?
- There are only 26 possible keys — *you can easily try them all!*

“It will keep your kid sister out, but it won't keep the police out.”

— Bruce Schneier (Cryptographer)

Another idea: One-time Pad

Step 1:

- Alice and Bob meet in advance
- Together they generate an array of random numbers that is as long as the message that Alice will later send Bob
- Each of the numbers in the array is between 0 and 25
- This array is the *one-time pad*

3	5	10	25	16	13	7	6	14	14	22	23	19	21	19	14	9

One-time Pad (cont.)

Step 2:

- To encrypt the message, Alice adds each letter of the message to the corresponding number in the one-time pad and takes the result mod 26.

THE LECTURER SMELLS

19	7	4	11	4	2	19	20	17	4	17	18	12	4	11	11	18

+

3	5	10	25	16	13	7	6	14	14	22	23	19	21	19	14	9

=

22	12	14	10	20	15	0	0	5	18	13	25	5	25	4	25	1

WMOKUPAAFSNZFZEZB

One-time Pad (cont.)

Step 3:

- To decrypt the message, Bob subtracts each number in the one-time pad from the corresponding letter of the ciphertext and takes the result mod 26.

WMOKUPAAFSNZFZEZB

⇩

22	12	14	10	20	15	0	0	5	18	13	25	5	25	4	25	1

–

3	5	10	25	16	13	7	6	14	14	22	23	19	21	19	14	9

=

19	7	4	11	4	2	19	20	17	4	17	18	12	4	11	11	18

⇩

THE LECTURER SMELLS

One-time Pad — the good news

Incredibly strong security: the ciphertext "looks random" — it is equally likely to be the encryption of any message of the same length

One-time Pad — the bad news

- Alice and Bob must share a secret as long as the message itself

- Using the same one-time pad more than once compromises security — hence the adjective "*one-time*" (Hopefully, you'll see why in lab)

- The one-time pad must be truly random. How does a computer get randomness?

Random source hypothesis

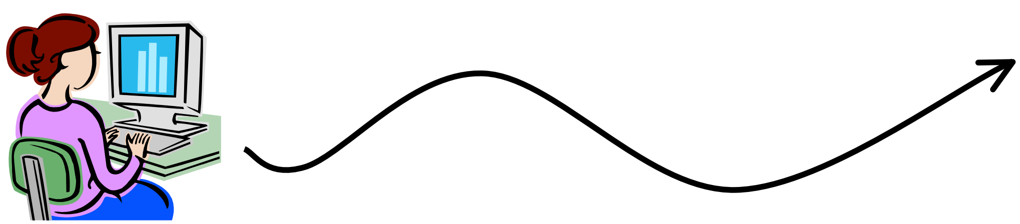
- Integral to modern cryptography

→ 0110101010011010011011101010010010001…

- I and my computer have a source of random bits
- These bits look completely random and unpredictable to the rest of the world.
- Ways to generate: Quantum phenomena in semi-conductors, timing between keystrokes, etc.

Communicating with strangers

- So far, we have assumed that the sender and the receiver of a message have agreed on a secret key in advance
- But sometimes perfect strangers need to exchange encrypted messages
- How can you send your encrypted credit card number to Amazon?



amazon.com

Insecure link (Internet)

(Jeff Bezos ’86)

Public-key cryptography

- **Main idea:** Amazon has 2 keys:
  - A *public key* that everyone knows
  - A *private key* that only it knows

- **Important Property:** A message that is encrypted using the *public key* can only be decrypted using the *private key*

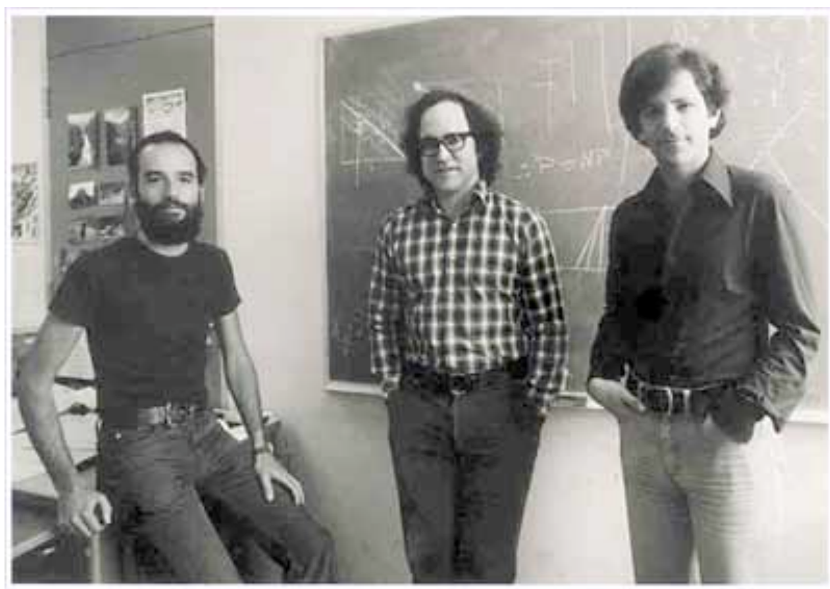
Public-key cryptography at a conceptual level

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

- Example:
  - Enter your credit card number
  - Put it in box, ship it to Amazon
  - Amazon opens box, recovers your credit card number

RSA

- One of the most popular implementations of public-key cryptography
- Rivest, Shamir, Adleman [1977]



RSA (cont.)

- Pick 2 large random prime numbers p and q — *random source hypothesis!*
- Let N = p • q
- "Derive" values e and d from p and q such that e and d are mathematical inverses — *leaving out many details!*

public key = (e, N)

private key = (d, N)

RSA and integer factoring

- The security of RSA depends on a problem that is easy to generate, but seemingly hard to solve: integer factoring

- If you could efficiently derive p and q from N (i.e. factor N), you would be able to derive e and d
- And once you know d, you know Amazon's private key!

Integer factoring (cont.)

- **Easy to generate:**
  Just multiply two prime numbers (N = p • q)
- **Seemingly hard to solve:**
  Given N, find p and q
  - What algorithm could you use?
  - What if p and q are each hundreds or even thousands of bits long?

(Aside: factoring is also **easy to verify** because given a potential solution p and q, you can efficiently verify that N = p • q. Indeed, factoring is in **NP**.)

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

(Aside: If quantum computers ever get built, may become easy to solve.)

Last theme

Suppose you observe something

What does it mean to learn *nothing* from it?

Suggestions?

One-time pad revisited

WMOKUPAAFSNZFZEZB

THE LECTURER SMELLS

⇩

WMOKUPAAFSNZFZEZB

Eve

WMOKUPAAFSNZFZEZB

⇩

THE LECTURER SMELLS

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

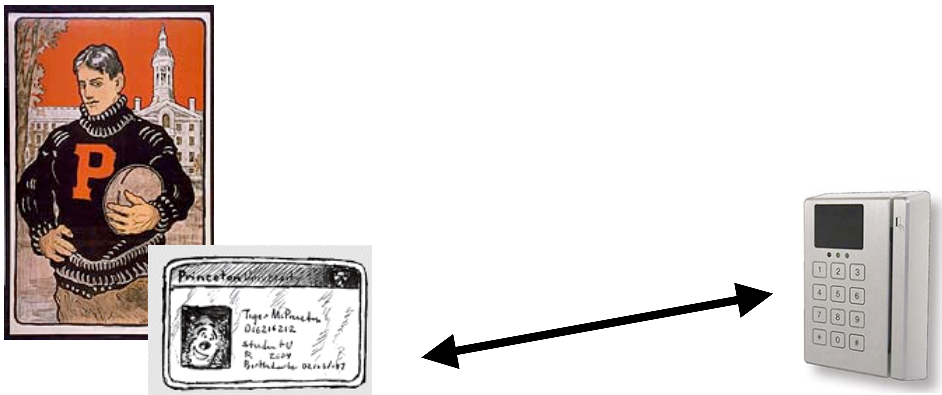
↑

Eureka! moment for modern cryptography

Zero Knowledge Proofs

[Goldwasser, Micali, Rackoff ’85]

Student



prox card reader

prox card

What we want:

- Prox card reader should accept real prox cards and reject fake ones
- But it should learn nothing about the prox card except that it *is* a prox card (e.g. to preserve privacy, it shouldn’t learn which prox card it is)

“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

- Suppose that I know what distinguishes sock A from sock B, but you don't
- Now suppose that I want to prove to you that I know what distinguishes them
- Normally, I would just tell you: "Look, sock A has a tiny hole and sock B doesn't!"

Illustration: Zero-Knowledge Proof that "Sock A is different from sock B" (cont.)

Sock A

Sock B

- But what if I don't want to give away the distinguishing feature?
- I could use the following 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 do you learn "nothing"? (Except that the socks are indeed different.)

Main themes of today's lecture

- Creating problems can be easier than solving them
- Difference between seeing information and making sense of it
- Role of randomness in the above
- Ability of 2 complete strangers to exchange secret information