Intro to Modern Cryptography
– J. KATZ + Y. LINDELL

POIS

FANTASTIC Q When is a problem Infosec one? [Ans] When the problem is impossible to solve logically / perfectly.

Will be shown with standard examples.

eg1: Hashing password.
theoretically impossible to be perfect if length is not infinite.

eg2: secure communication
@ t_0 info(R) = info(eve)
@ $t > t_0$ info_rec(R) = info_rec(eve)

eg3: Data integrity
If m was sent, modified to m' → It is the same to receiver.
and if m' " " ,
└ cannot thus identify.

Eg of non infosec → infosec

- problem in distributed computing ∴ now an Infosec problem.

Solution: use signatures. ⇒ implying signatures are impossible.

FASCINATING Q. How to logically solve / circumvent a logical impossibility?

[Ans] Bring in another impossibility and make it destructively interfere with the original one.

We focus on 4-5 sources of impossibilities in the semester.

Course: See impossibilities
Introduce others
Solve them
1 per month approx.

Random Words
– Hamming Distance

FUNDAMENTAL
- Information security is God
– all nontrivial works of science must include info sec

– logical nogo

Sources of Impossibility.
① Computational Hardness [Resource Complexity]
② Practical Uncertainties (Only happy cases are secure.)
③ Natural Limits (Speed of light, Quantum Stuff)
④ Logical / Philosophical Impossibilities

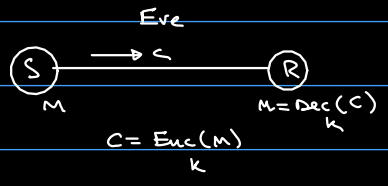
7.1.20

Topics to cover

- Kerckhoff's Principle
- Designing / Breaking classical ciphers.

Starting off with secure communication networks.

- traditional ciphers, and how to break them.
- Shannon next class.
  - defined information
  - path breaking.



Caesar Cipher

$C = (x + 3) \bmod n_c$

M = message
C = coded message

n_c = no. of characters in alphabet.

Brig talk about his perspective of infosec as an 'art', and a bigger rant on what is art and what is science.

Exact words in book.

Kerckhoff's Principle

Security of a system must NOT depend on the OBSCURITY of the algorithm, rather must solely depend on the SECRECY of the KEY.

Kerckhoff's Reasoning

1. Algorithms are reverse engineerable.

   eg: $h(x) = 5x + c$

   Attacker can feed $x_1, x_2, \ldots x_n$ and see that for all outputs $h(x_i)\ldots$

   $8 \mid h(x_i) - h(x_j)$. And then solve for c.

2. Updation / Recovery Complexity.

   if password revealed in secure system: change pass.
   if algo " in obscurity " : !! fixed.

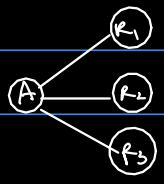
3. Secure Memory is costly.

   Proof in modern times: UTM is algo and just input is key.

   — bad information storage efficiency.

ASK ATHREYA

4. Scalable



Without: Diff algorithm for everyone.

With: only 'key' changes among people.

Additional Reasoning

1. Ethical Hacking

hypothesis: no system is secure.

bug (algo) will be found because

- → bug exists
- → nonethical people exist
- → only " " search for bug
- → " " " will find "
- → big F, take that L

2. Standards

- needed for efficiency
- if an algo is used by every system, it should obviously follow

Thus we can see why Caesar cipher fails.

Next iteration:

Shift Cipher:

$c = (x + k) \bmod n_K$

m = message
c = coded message
k = key
n_c = no. of characters in alphabet.

→ If keyspace is smol, attack ez. (brute force)

first principle learnt:
Principle of large key space

ATTACK

1. Can be broken by humans
2. Autobreaking:
   - frequency analysis

$p_i = P(i^{th} \text{ char in } m)$
$q_i = P(\text{" " " } c)$

Precompute $\sum_{i=0}^{25} p_i^2 \simeq 0.065$

Now compute $\sum_{i=0}^{25} (p_i q_{i+k})$ — wrong guess: $1/26$; correct guess: $\simeq 0.065$

Next iteration:

Monoalphabetic Substitution Cipher

- Diff alphabets shift by different amts.
- no repetitions allowed
- basically, permutation

for brute force: 26! keys to search ← applying first principle

ATTACK

$\forall i \ \exists j : q_j \simeq p_i$ ⇒ 1. Sort q_is, 2. Sort p_is — since distribution is same, eg: $p_a = q_n$, $p_e = q_t$, $p_t = q_b$

Issue: susceptible to frequency attacks.

Next iteration

Vigenere Cipher

1. does not maintain frequency

eg: hello
sease
zilh_

FAILED ATTACKS
- brute force: too many keys
- freq anal: freq. not maintained.

Can be broken if:
1. key length known
2. " " is findable.

ATTACK

PART 1:

if we know length of key, eg: 3 C_0 C_3 C_6 ....

partition ciphertext into ((length)) parts C_1 C_4 C_7 ....

then shift cipher attack on all ((length)) parts. C_2 C_5 C_8 ....

PART 2:

Guess an l.

take a string $C_0 C_l C_{2l} \ldots$

check if $\sum_{i=0}^{25} q_i^2 = \sum_{i=0}^{25} p_i^2$

easy