Basics of Cryptography

Start Guessing

SIGPWNY

Slides by Sam Ruggerio

Meeting Flag

Goal of Cryptography

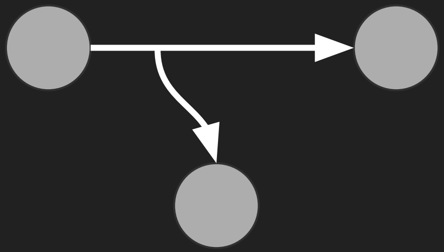
Sending data from point A to point B without it being tampered or viewed from outside attackers.

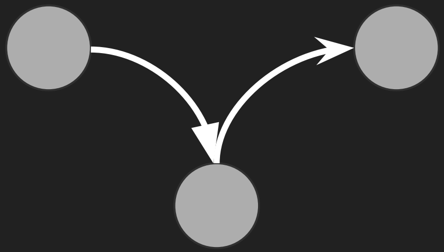
What this boils down to:

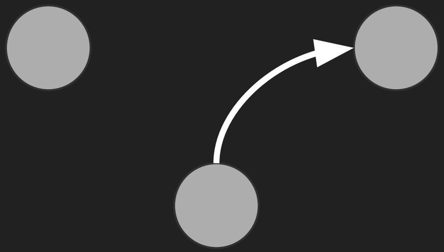
- **Confidentiality:** The recipient is the only one who has received and viewed the data.
- **Integrity:** The data has not been tampered/modified in any way.
- **Authenticity:** The recipient has received data from provably the correct sender.

What this protects against:

- **Interception:** Getting a copy of the sent data.
- **Modification:** Getting the sent data, modifying it, and sending it to the recipient.
- **Fabrication:** Sending data to the recipient under the guise that it is a trusted sender.







Chal:
data obfuscation
EASY

Data in Modern Crypto

Data formats in modern cryptography will be:

1. Complete and utter garbage ciphertext (j�b�l �� r)
2. Base 2, 16, or 64 (keys or other important values)
3. Plaintext (usually our result)

- Easier to send long numbers as succinct hex or b64 strings.
- Garbage ASCII can be interpreted as binary, which *may* be useful depending on context.

Basic Ciphers- Substitution

Caesar Cipher (aka rot substitution) is the simplest of ciphers, rotating the basic alphabet (26 chars), x chars and substituting each character for the rotated one.

rot3:

```
alphabet:  a b c d e f ...

cipher:    D E F G H I ...

plain:  Hello world

cipher: KHOOR ZRUOG
```

This gives way to our first attack vector, brute forcing.

Chal:
caesar cipher
EASY

Brute Forcing

- Try every possible combination of keys and values
- This will *eventually* crack any cryptographic method
- However, it is the goal of any cryptography algorithm to make this take a very ***very*** long time.

Our caesar cipher only has ($|\Sigma|-1$) keys! In non-symbol (a-z) english, 25 total keys.

Can be very quick to brute force a rot cipher.

Σ refers to the set of characters in our alphabet

Improving Substitution

Let’s randomize each letter position instead of a rotation!

This gives us about |Σ|! (26!>4E26) keys!

```
plain:  a b c d e f ...
cipher: Y K L P S B ...
```

But there’s an issue: We can pretty safely guess both the alphabet of the cipher and plaintext.

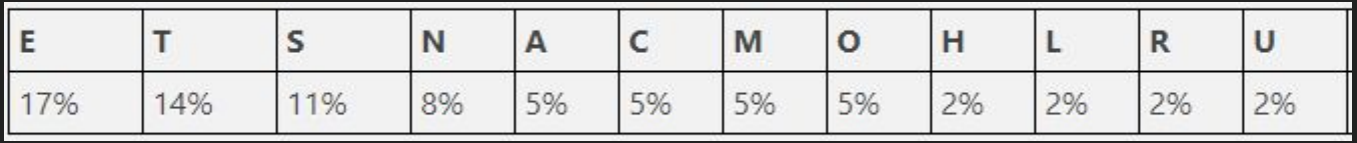
A new attack vector: Frequency Analysis

Frequency Analysis

Goal: Use traits of the language to identify trends in letter usage

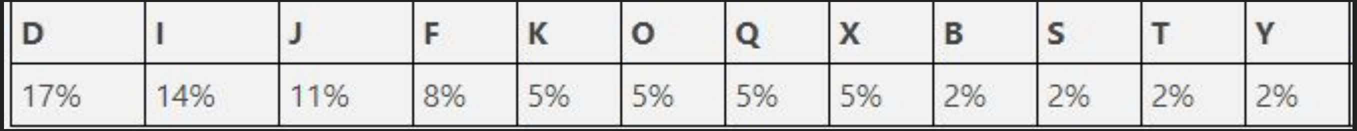
Use it when you know the language of plaintext, and the type of cipher (best for substitution).

```
ptext: a sentence that uses common letters
```



E	T	S	N	A	C	M	O	H	L	R	U
17%	14%	11%	8%	5%	5%	5%	5%	2%	2%	2%	2%

```
ctext: q jdfidfkd isqi yjdj koxxof bdiidtj
```



D	I	J	F	K	O	Q	X	B	S	T	Y
17%	14%	11%	8%	5%	5%	5%	5%	2%	2%	2%	2%

```
cipher: QHKADEWSUCGBXFOMNTJIYVRZLP
```

Frequency Analysis

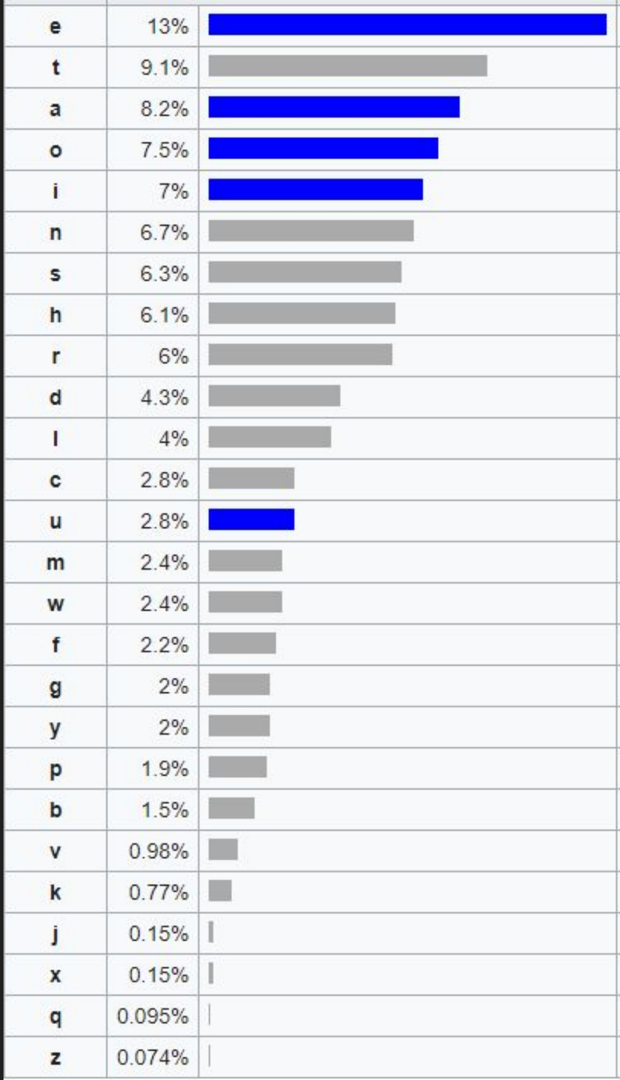
Works best with lots of sample cipher text.

Does require some trial and error with testing letters.

If given spaces, can be significantly easier.

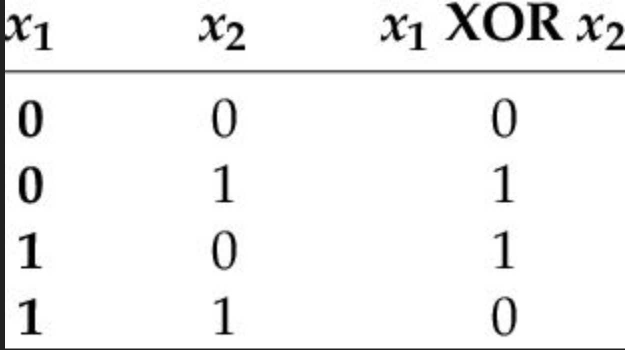
Find vowels and most common consonants.

Chal:
frequent frequencies
HARD



e	13%
t	9.1%
a	8.2%
o	7.5%
i	7%
n	6.7%
s	6.3%
h	6.1%
r	6%
d	4.3%
l	4%
c	2.8%
u	2.8%
m	2.4%
w	2.4%
f	2.2%
g	2%
y	2%
p	1.9%
b	1.5%
v	0.98%
k	0.77%
j	0.15%
x	0.15%
q	0.095%
z	0.074%

Chal:
xor pad
EASY



x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Math in Crypto: XOR

Foundation a lot of modern encryptions.

XOR is DESTRUCTIVE but REVERSIBLE

You lose information of its parts after the operation.

You can reverse it by xoring it again with one of the original parts.

Seeing a 1 in the result means either the plaintext or key had a 1, but you don’t know which. Seeing a zero means both were either 1 or 0, but you don’t know which value.

Encoding a length n message with a length n key requires 2^n guesses to crack.

Math in Crypto: modulo

The “remainder” operator.

Also destructive! 1 mod 2 = 1 = 9 mod 2

Sort-of reversible (you will see in RSA).

Recall: Congruence is when x mod n == y mod n

modulo also stays within operation space: add/sub/mul/exp are all valid.

Modern Encryption

Classical Encryption usually has a major pitfall:

It is almost all symmetric encryption, you have to distribute one key.

Public Key cryptography (asymmetric) solves this problem:

- Generate a pair of keys.
- Distribute one key that can encrypt data.
- The 2nd key can decrypt data and is kept private.

You can encrypt as much data as you want with the “public” key, but only the “private” key can decrypt it, so you can send secure data to the recipient. (who generated both keys)

RSA key generation

Pick two prime numbers: p, q (usually ***very*** large, 10E100)

Calculate n = $p \times q$

Calculate “Euler’s Totient Function”* for n: $\phi(n) = (p\text{-}1)(q\text{-}1)$

Select e such that $\gcd(\phi(n),e)=1$; $1<e<\phi(n)$

Calculate d such that $ed=1 \bmod \phi(n)$; $1<d<\phi(n)$

e, n is your public key; d, n is your private key

*You can use Carmichael’s totient function: lcm(p-1,q-1)

(Most people pick e=65537 for regular RSA, since it is prime it is inherently coprime to any $\phi(n)$).

Chal:
technically rsa
MEDIUM

RSA encryption/decryption

Convert our plaintext to a number less than n (do it in chunks if its larger than n)

Ciphertext $C = P^e \pmod{n}$

Plaintext $P = C^d \pmod{n}$

Factoring very large numbers takes an extraordinary amount of time. RSA is ***very*** hard to attack without knowing the private values to begin with.

Chal:
prime god
HARD

The Process

Classical Encryption

- Identify the Cipher (frequency analysis of bigrams can help too)
- Use freq analysis or brute force, or other weaknesses in cipher to decode text

Modern Encryption

- Usually impossible without external information or a quantum computer.
- If given information, find mathematical overlap to find other private values or plaintext. Remember that most modern encryption is based on XOR or modular arithmetic.

The Tools

Frequency Analysis Tool - Tool to see breakdown and make guesses on plaintext

Cipher Detector - Does the work for you and identifies the ciphertext

Cyber Chef - an **excellent** swiss army knife cryptography tool

Dcode.fr - a verbose source to encrypt and decrypt in many ciphers and explains how most of them work

Pwntools - a python framework to help automate responses or exploits through netcat connections

Pycrypto - a python package for cryptographic tools and encryption/decryption processes

Mod Multiplicative Inverse - solves $ed = 1 \bmod \phi(n)$