Cryptography

History up to 1950

Short interlude from binaries

- We will get back to binaries soon
- Don't want you to get tired of assembly

What is cryptography

- Three main goals of cryptosystems:
  - Confidentiality
  - ~~Integrity~~
  - ~~Authenticity~~
- Old style integrity check:
  - If message makes sense,it probably got through ok
- Old style authentication:
  - If message bearer is someone you know, wax seals, etc

Basic idea

1. **Alice** takes message (called *plaintext*)
2. Uses a function *encrypt*(*key*,*plaintext*) to generate *ciphertext*
3. Sends *ciphertext* to **Bob**, who already knows *key*
4. [**Eve** eavesdrops and hears the message, but doesn't know *key*]
5. **Bob** uses *decrypt*(*key*,*ciphertex*t) to get back *plaintext*

Desire

- Make it impossible for Eve to recover plaintext, given that scenario
  - Is such a thing even possible?
- Make it *really hard* for Eve to recover plaintext
  - How hard?
  - Hard enough that brute force trial of keys is the best way to recover a message
  - [256 bits of security "should be enough for anyone"]

"Classical" Strategy

- Traditionally there are two main approaches
  - Substitution ciphers
  - Transposition ciphers
- Substitution cipher
  - Map each character in plaintext to a new character in the ciphertext
  - Key describes how this mapping works
- Transposition cipher
  - Change the ordering of letters in plaintext
  - Key is usually the description of the transposition

First attempt

- "Caesar Cipher"
  - Simplest substitution imagineable
  - Circularly rotate alphabet by some fixed amount

  - Key:a single letter, A-Z

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
DEFGHIJKLMNOPQRSTUVWXYZABC
```

First attempt

- Vulnerabilities:
  - Key space is trivially small
  - Pretty easy to guess and check, not much else needed

Second attempt

- Cryptogram
  - Rather than a simple rotation, use a full permutation of the alphabet
  - Key space is now 26! which is about 88 bits which is pretty good!

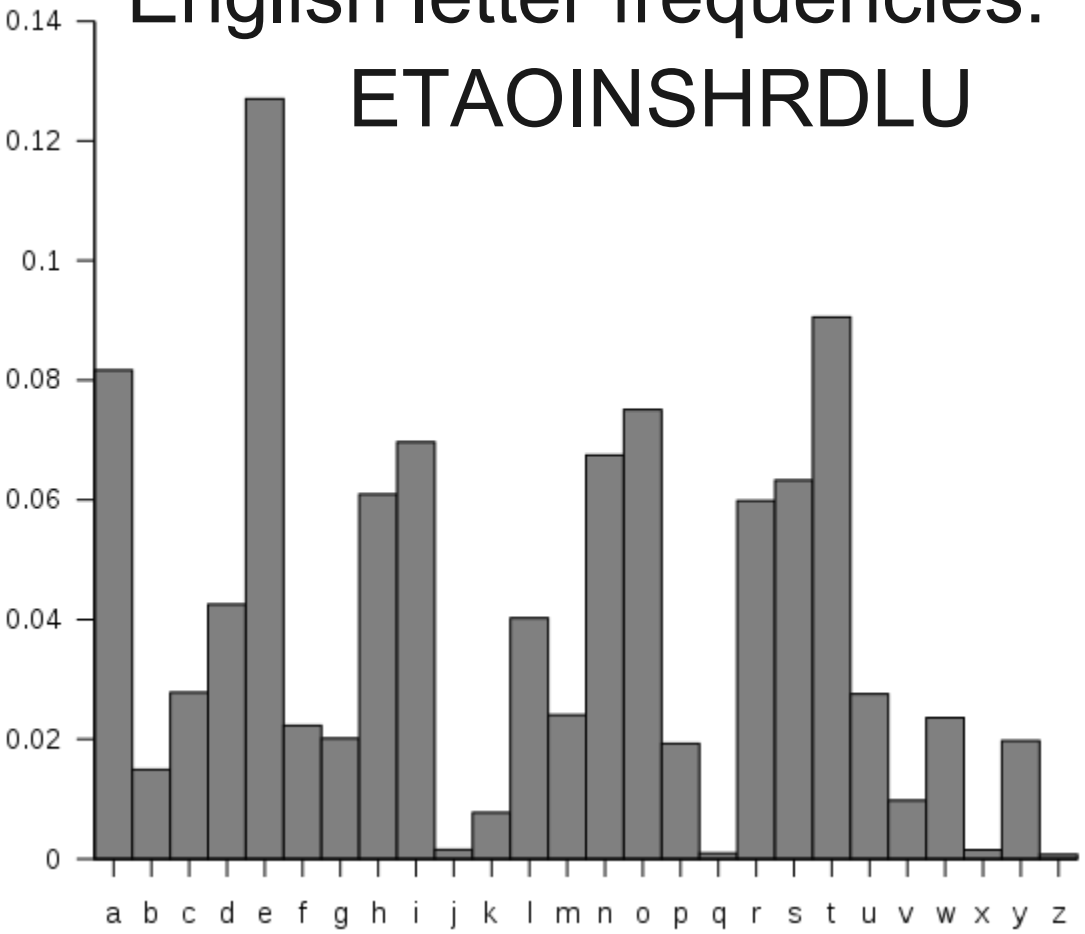
```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
FJXYKVCNOWBLMAPZQSTRDHGIEU
```

Second attempt

- Vulnerabilities:
  - Character *c* always maps to permute[*c*]
  - This lets us do frequency analysis!
- Frequency analysis
  - "real" text is not uniformly random
  - Look for most common encrypted letters, match them up with most common letters in English alphabet [same for bigrams, trigrams, etc]

Frequency Analysis

English letter frequencies:
ETAOINSHRDLU



Second attempt

- Tons of tools exist to solve these automatically
- Can also be done easily by hand

Attempt 3a

- Vigenere cipher
  - Let's use a set of permutations rather than just one!
  - Key is a set of Caesar shifts
  - Keyspace 26^n, where n is number of shifts
    - key must be *random* or else keyspace drops
    - takes ~54 chars for 256 bits of key

Attempt 3a

- Vulnerabilities:
  - First figure out length of the key
    - Search for repeated patterns
    - Find GCD of offsets of patterns

fhkkcfitbjqhgbcewktnmszpgnaznwokorwetzpgiatbvasnwyfhkeqddzpgdevmcfejinat

Δred:50
Δgreen12:15
Δgreen13:35
Δgreen23:20

Attempt 3a

- Vulnerabilities:
  - First figure out length of the key
    - Search for repeated patterns
    - Find GCD of offsets of patterns

fhkkcfitbjqhgbcewktnmszpgnaznwokorwetzpgiatbvasnwyfhkeqddzpgdevmcfejinat

Δred:50
Δgreen12:15
Δgreen13:35
Δgreen23:20

Keylength: 5

Attempt 3a

- Once you have key length, problem reduces to Caeser cipher
  - More ciphertext is useful so that you can get enough info for frequency analysis
  - With a long enough key, this is actually hard to break...
- Again, automatic tools can usually solve these for you!

Attempt 3b

- Polyalphabetic substitution
  - Popularized as the Zodiac cipher
  - Basic idea:
    - Cryptograms fail mostly due to frequency analysis
    - Use multiple replacements for each letter! [many to one encoding]
    - Pick which replacement to use at random
- So E might map to {¥xŁБ} each with 1/4 probability
- To decode, replace any instance of those characters with E

Attempt 3b

- Key can be quite large, depending on how many symbols you use
  - Definitely not bruteforceable
  - Easily extended by adding more symbols

Attempt 3b

- Vulnerabilities:
  - Any ideas?
  - "Cribbing"
    - Guess certain phrases will be in the plaintext [for example:"password","your key is",etc]
    - Make sure ciphertext is consistent with such an assignment [БŴĐБӨƧƐ is consistent with POOPIES, but not BADGERS]
    - Keep guessing and looking to see what this looks like

Attempt 3b

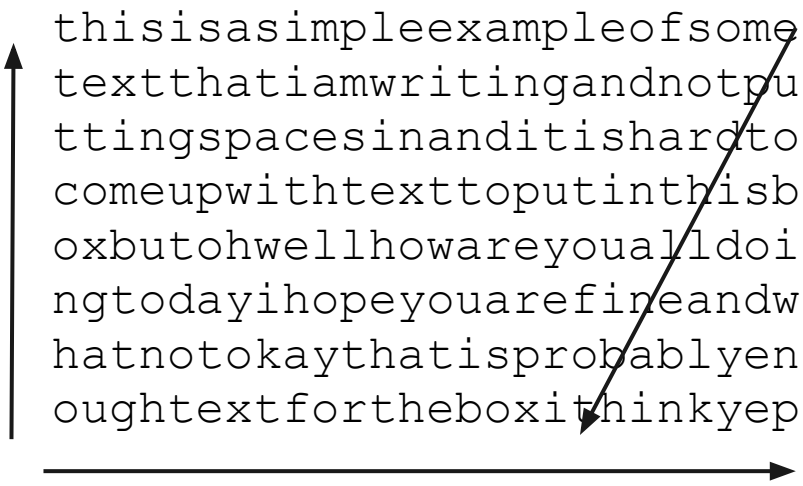
- "Hill climbing"
  - Pick some scoring function to rate plaintext answers [probably based on frequency of letters, bigrams, trigrams, etc]
  - Randomly assign values to the cipher alphabet
  - Change letters until you make your score higher
  - Can also use evolutionary algorithms to do this...
- Zodiac messages used this cipher
  - One was cracked (mostly using cribbing) in less than a week
  - One has gone unsolved for over 40 years

Attempt 3c

- Transposition cipher
  - Don't change the alphabet, just rearrange the letters

Attempt 3c

- Transposition cipher
  - Don't change the alphabet, just rearrange the letters



Attempt 3c

- Keyspace is complicated...
- Benefits:
  - Frequency analysis won't do you any good
  - Cribbing and hill climbing won't do you any good
  - Can be used in conjunction with other techniques
- Drawbacks:
  - Easy to tell apart short messages
  - Roughly as difficult as solving anagrams [useful tool:"an" linux command]

Examples:

OtxjumQdshmbfxgtwsnsXy.Otxjum,RnhmnIfs,ytOtmsAfqjsynsjfsiAjwtsnhfOfsj(séjGtymfr)Qdshm.
MnxkfymjwhfrjytymjZsnyjiXyfyjxkwtrSjbhfxyqj,Rtdsfqyd,HtzsydRjfym,Nwjqfsi,ns1866,
fsimnxrtymjwbfxgtwsnsRtsywjfq,Vzjgjh,Hfsfif,fsinrrnlwfyjins1856.Ns1887mjjsyjwjiXy.
KwfshnxXjrnsfwdnsRnqbfzpjj,Bnxhtsxns.FkyjwlwfizfynsIkwtrXy.HmfwqjxHtqqjljnsJqqnhtyyHnyd,
Rfwdqfsins1891,QdshmxyzinjiymjtqtldfyXy.Rfwd'xXjrnsfwdnsGfqynrtwj.
MjhmfsljimnxxyzidytqfbfsiymjsuwfhynhjiktwxjajwfqdjfwxsjfwHmnhflt,Nqqnstnx.
QdshmgjhfrjfhvzfnsyjibnymGnxmtuJibfwiOtxjumIzssj,
bmthtsanshjimnrytfgfsitsmnxqjlfqhfwjjwfsiwjxzrjmnxxjrnsfwdxyzinjx.MjfyyjsijiPjswnhpXjrnsfwdnsXy.
Qtznx,Rnxxtzwn,fsibfxqfyjwtwifnsjiytymjuwnjxymttitsOzsj9,1900.
QdshmymjsxjwajifxfhzwfyjfyXfhwjiMjfwyHfymjiwfqnsIfqqfx,Yjcfx,zsynq1902,bmjsmjgjhfrjufxytwtkXy.
XyjumjsxHmzwhmnsBjfymjwktwi]MjjwjhyjihmzwhmjxnsBjfymjwktwifsinsMfsiqjd.
MjbfxsfrjiymjktzsinslufxytwtkXy.Jibfwi'xHmzwhmfyIfqqfxns1903.Mjymjwjjxyfgqnxmjifhmzwhm,wjhytwd,
fsiufwthmnfqxhmttq.NsOzsj1910mjgjhfrjanhfwljsjwfqtkymjInthjxjtkIfqqfx.
MjbfxxmtwyqdfkyjwbfwixrfijFutxytqnhFirnsnxywfytwtkIfqqfxktqqtbnslymjijfymtkGnxmtuIzssjnsFzlzxy191
0.

Examples:

TeciuqdakcfejqpEuzbqz-ot-TqgnjippazUcsdgtgzeIwnxemm,Emmhzkpgk,ejqrkpgneiioqaLmnxoc.
Pgiayipmsyweuazmqratconexwhunzmnxeibwmllqigrkaqrtnmfmy,
ovrmrzqeglgzTucnitpCaudqrrippwnwuqDktgsihcuzazctmenmgpizmfmnjbqihokjteiwpfrojwfejipMdjzgesz
wvteXmcpex.PgussmpfiuvgpitXgbyy'aFuaxg.JqwgaetavtcunzwQdlgvfaBxqfsesip,
mnjcuqdzpgootbcotzwugpvwtfCaudqrripp.HkececnitmczmtusklceaRivutalkzaxqcz.
HkiueoiqcfejekfhPwjzTotnatywpmnjMfiaxlUfirtkzgltgqt,
ovxalbmfiizpvtesippJuppIirskzsovczahwtfibmrdovwumllwtmcuurdenmpeiuvqrpxmunyzmtuatayutnqpfhk
KjgripqrEtonmnj,kqymavkoazmfnyHzkpgkucztuZkohgzfNadbgdatlqfhkzuunkitxy1668.
GvvtotgYaojackszpcfaitwnfuzoqdhgYulqqpetuxtamubgoosxtqhkvuuotcuqdzwoqezivfhk'kjmmhmteolbj
mtmzgmtzzkymkzczdrivutalkzaxqcz,Dx.PgleqqctBazvan.'

Examples:

GEIDIUNOORPKCLCHOCJ-ONXIVAMIBGRLIYNSGEIBOCLLNBCOIFCVHGEIFVPCSIPZNFI.
GEIKVFYNLNGLIOHCYNPNSRGNDIHVFGEIOCGNSKVFY,DIORP,LCNO,
KENBEBVSHNFPLGEIENLGVFNBCOIDNYISBI(HFVPBVNSLCSYLBROZGRFI)
GECGDIUNOOCKIFIONGIFCOOT"ONGGOILCNOL"N.I.HOCJ-ONXILGCSYCFYL.
NSGEIDIUNOORPGEIBOVGEKCLYFCZIYHFVPCEVFNQVSGCOBFVLLACFLRLZISYIYHFVPGEIL
GCHH;
GENLNLRSONXIPVLGPVYIFSHOCJLNSKENBEGEI'EVNLG'VHGEIBOVGENLCGGCBEIYYNFIBGO
TGVGEIDIFGNBCOLGCHH.
GEIAICFIFVHCDIUNOORPKCLXSVKSCLCDIUNOOCFNRLVFDIUNOONHIF.
MRLGCLNSGEIBCLIVHGEIFIJNPISGCOBVOVFLVFHOCJVHKILGIFSFIJNPISGL,
GEIDIUNOORPKCLCGFICLRFIYLTPAVOVHGEIPNONGCFTRSNGGECGNGFIZFILISGIYCSYNGK
CLBOVLIOTYIHISYIYNSBVPACG.SICFOTCOOVHGEIZFILISG-
YCTFIJNVSLVHNGCOTZFILIFDIGEIRLIVHDIUNOOC.
PCSTBEFNLGNCSZFVBILLNVSCOACSSIFLCFINSGEIDIUNOORPHVFP;
RLRCOOTGEILIACSSIFLCFIGIFPIYOCACFCCHGIFGEILGCSYCFYCYVZGIYATGEIHNFLGBEFNL
GNCSFVPCSIPZIFVFBVSLGCSGNSINKENBEFIZOCBIYGEINPZIFNCOICJOIKNGEGEI"BEN-FEV"
LTPAVO

Other ciphers

- Book cipher
  - Secret key is a book
  - Use offsets into book as ciphertext..
- Bacon cipher
  - Not really cryptography, but steganography
  - Highlight certain letters in an otherwise innocuous message
    - Highlighted letters spell out some secret message
- Vernam cipher (aka xor)
  - Probably the single most important cipher in all of cryptography

Xor

1. Take a message, represent it as binary
2. Generate a random key, also represented as binary
3. xor the two together to get your ciphertext
4. xor again to get your plaintext!

Xor

- Why is this the most important cipher in cryptography?
  - Forms the basis of the "one time pad"
  - If your secret key is as long as your message, it is *impossible* to recover the plaintext from the ciphertext
    - Literally impossible from an information theoretic standpoint (see Claude Shannon)
      - Why?
    - So why isn't cryptography "solved"?
      - How are you going to securely distribute a key as long as your plaintext?

Xor

- Many cryptosystems try to mimic the behavior of the one time pad
  - RC4, block ciphers in CTR mode
- Many systems can be modelled and analysed somewhat like OTP use
  - block ciphers in CBC mode, block ciphers in ECB mode to some extent
- In the real world (and in competitions) xor is one of the most common forms of cryptography
  - And **not** in the secure, one time pad sort of way
  - Dirt easy to implement and efficient in C

Xor

```
void xor(char* str, int key, int len) {
  int i;
  for (i = 0; i < len; i+=4) { //ignore those 3 bytes
    *((int*)&str[i]) ^= key;
  }
}
```

How many bits of security? 32, or less?

Xor

- "many time" pad xor loses it's magical security properties!
  - Everyone still uses it anyway
  - Again, so easy to implement...
  - It also works on *any* data that can be put into binary, so a lot more useful than other ciphers!
- How do you do cryptanalysis with a *file* encrypted with xor?
  - Pretty much identical to Vigenere
  - What is the most common byte?
    - Binary data: 0x00 or 0xff

Summary

- Algorithms seem insecure (bits of key > bits of security)
- Still used today, because people are lazy/stupid
- Great model even for many *modern* cryptosystems
- Analysis and techniques will come up again