§ 4.6 Frequency analysis and the Vignière cipher.

The Vignère cipher remained "uncracked" for about 300 years. It was believed to be very resistant to frequency analysis because of how it effectively "smooths out" the distribution of letters in the English language.

The key to cracking the whole thing turns out to be determining the length of the key, which can be done as follows:

The Kasiski Test (~1863)

Idea: look for repeated strings in the ciphertext (three or more characters in length). If we find these the distances between them are likely to be a multiple/multiples of the key ~~length~~. E.g.

plaintext	<u>maths</u> is short for <u>math</u>ematics
key	keykeykeykey . . .
ciphertext	$\underline{\text{WERR}}\underbrace{\text{WGCWFYVRPSP}}_{\text{11 letters}}\underline{\text{WERR}}\text{IKKXGMW}$

Since the distance from the end of one occurrence of "math" to the first letter of the next is

$$12 = 3 \times 4$$

we get a repeated string.

So if you saw:

$\underline{\text{WERR}}\,\underbrace{\text{WGCWFYVRPSPW}}_{\text{12 letters.}}\underline{\text{ERR}}\text{IKKXGMW}$

You would guess "perhaps the key length is a divisor of 12 – so 2, 3, 4, 6, 12".

Of course, there's a chance that repeated strings arise for reasons unrelated to key length – but this happens less frequently than repeated strings arising from key length, especially if the strings are long.

E.g. The book gives an example of a very long ciphertext w/ repeated strings. The repeated strings are:

Repeated string	distance between occurrences
VOGYVQBND (×2)	50 = 2×5×5 ~~×10~~
CCBPDBEL (×2)	120 = 2×3×4×5
KTPCZ (×2)	190 = 2×5×19
HLNL (×2)	70 = 2×5×7
VOG (×4)	50, 55, 15 (all mult. of 5)
OGH (×2)	334 = 2×167
ZVS (×2)	120 = 2×3×4×5
VTA (×2)	165 = 3×5×11
NLC (×2)	89 = prime
	⋮
	many more.

Note the frequency of 2 and 5 above, then we start getting oddball numbers like 89, 334, etc. These are likely just coincidence, the first few in the table (especially the long strings) are likely not.

So we guess a key length of 2, 5, or 2×5 based on the first few repeated strings — and rule out 2 since this is probably too short for any serious cryptographer to have used (also would probably result in many more repeated strings).

Remark: There are formulas for which key lengths you should guess based on the string lengths, distances between them and factorizations. We will not focus on learning these, as they are all ad-hoc anyways.

Now, suppose you have the key length: what next?

Say, as in our case, the key length is 5. Set:

L_0 = set of all letters that occur in positions 5, 10, 15, 20, 25 (with repetition!!)

L_1 = set of all letters that occur in positions 1, 6, 11, 16, ... etc. (with repetition!)

and in general:

L_i = set of all letters that occur in positions congruent to i mod 5 (or in general, i mod keylength)

Now suppose our length 5 key is "abcde". The key observation is:

All the letters in L_i were encrypted using a Caesar cipher with shift equal to the ~~(i-1)st~~ i-th letter in the key! (Where we take "zeroeth" to mean last)

E.g. Letters in positions $\overbrace{2, 7, 12, 17, 22}^{L_2}$, etc were all encrypted by adding 1, corresponding to "b" in the key.

Letters in positions 3, 8, 13, ... etc were all encrypted by adding 2, corresponding to "c" in the key.

Conclusion: For each set L_i, since it's a Caesar cipher the distribution of letters should match the distribution of letters in english.

I.e. If L_i contains 13% Q's, then "Q" probably corresponds to "E". So the i^{th} letter in the key must be:

$$E + ? = Q$$
$$4 + \underset{\uparrow}{?} = 16$$

12! So the ith letter must be the 12th letter — M.

Then do the same for all "i" characters in the key.

HCATC.

There are easy defences against a frequency analysis attakk like the ones above, e.g.

(1) changq xax phew litturs aynd txeh myyning xis clur (at least for native english speakers).

However, this is time-consuming and it opens up the possibility of the intended recipient misinterpreting the message.

(2) Deliberately choose strange words. E.g. there's an entire novel "Gadsby" by Ernest Vincent Wright that never uses the letter "e". Again, this has the same problems as above (more labour to prepare the message, plus possibility of misinterpreting).

Also:

(3) Realistically, we are not going to use cryptography to send cute messages. We are going to send things like banking info and credit card numbers, ~~which~~ where we cannot "add a few letters/digits" to thwart eavesdroppers.

§5.1. Breaking the permutation cipher.

First, note that the permutation cipher is <u>designed</u> to be impervious to frequency analysis.
Specifically: We made a cipher for which the letters of the ciphertext are the same as the letters of the plaintext, just in a different order.
Therefore if we did a frequency analysis we would learn nothing.

<u>Breaking the cipher</u> is easy:

- First, the length of the block has to be a ~~multiple~~ divisor of the length of the ciphertext.
- Next, we can either:
  - (i) For each divisor, focus on trying permutations of the first block that "make sense", ie test until we find something that looks like the start of a message.
  - (ii) Look for blocks that contain combinations of letters whose order we can guess based on our knowledge of English. Test relevant permutations.

<u>Example</u>: Suppose we have

RIBNT HGEES MSGEA TTHOE RODPO IPNRLTH.

Ciphertext length is 32, so block sizes are one of:

2, 4, 8, 16.

Approach (ii) :

If we guess block length 4, then the first block is RIBN. "English intuition" probably tells you these letters will only fit together as "BRIN", so the permutation is

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \end{pmatrix}.$$

This works, and reveals the message
"BRING THE MESSAGE TO THE DROP POINT $\underbrace{\text{RLH}}_{\text{padding.}}$

Approach (i) is pretty self explanatory so we don't go into detail here, but it is very easy to implement on a computer (try permuting first letters, compare against a dictionary).

---

§5.2 Breaking the Hill cipher

This is a significant security improvement, but is still vulnerable. As before, we can use divisors of the length of the ciphertext to guess the block size. Having guessed the block size, do a frequency analysis on "bigrams" (if block size 2)
or "trigrams" (if block size 3)
or "n-grams" (if block size n).

For example, suppose we guess a block size of 2. Then the Hill cipher corresponds to multiplying each block by $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, some choice of a, b, c, d,

where a, b, c, d are chosen so that the matrix has an inverse. In other words, we apply a function A that is 1-1 and onto

$$\{\text{pairs of letters}\} \longrightarrow \{\text{pairs of letters}\}$$
$$\begin{pmatrix} x \\ y \end{pmatrix} \longmapsto A\begin{pmatrix} x \\ y \end{pmatrix}$$

So it's just a permutation of pairs! We know how to crack permutations. Make a list of common pairs:

Most common pairs

th ↑ more common
he
in
en
nt
re
⋮ ↓ less common.

Then look for pairs in our ciphertext that repeat:

this counts

$\overline{FR}\,\overline{AQ}\,\overline{R\ T}\,\overline{FR}\,\overline{LZ}\ \overline{KE}\,\overline{FP}\,\overline{Z\ K}\,XYWS$ . . . etc etc . . .

does not count

In the example in the book, the most common pairs are GZ and ZK, occurring 35 times each.

You have to be careful about counting:
The occurrence of "ZK" in positions 9 and 10 is not a block, so we do not count it. On the other hand, the second occurrence counts since it is a block.

Our approach: The decryption matrix is $\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1}$, we'll write this as $\begin{pmatrix} x & y \\ z & w \end{pmatrix}$ and solve for $\begin{pmatrix} x & y \\ z & w \end{pmatrix}$ based on some guesses. Guess ①:

$$\begin{pmatrix} x & y \\ z & w \end{pmatrix} \underset{\overset{\uparrow}{GZ}}{\begin{pmatrix} 6 \\ 25 \end{pmatrix}} = \underset{\overset{\uparrow}{TH}}{\begin{pmatrix} 19 \\ 7 \end{pmatrix}} \mod 26.$$

So
$$6x + 25y \equiv 19 \mod 26$$
$$6z + 25w \equiv 7 \mod 26.$$

Guess ②:
$$\begin{pmatrix} x & y \\ z & w \end{pmatrix} \underset{\overset{\uparrow}{ZK}}{\begin{pmatrix} 25 \\ 10 \end{pmatrix}} = \underset{\overset{\uparrow}{HE}}{\begin{pmatrix} 7 \\ 4 \end{pmatrix}} \mod 26,$$ so 2 more eqns.

I.e. we guess the most common "bigrams" in our ciphertext correspond to the most common "bigrams" in English.

Overall:

$$\left.\begin{aligned} 6a + 25b &\equiv 19 \mod 26 \\ 25a + 10b &\equiv 7 \mod 26 \end{aligned}\right] \text{ solve for } a \text{ and } b$$

$$\left.\begin{aligned} 6c + 25d &\equiv 7 \mod 26 \\ 25c + 10d &\equiv 4 \mod 26. \end{aligned}\right] \text{ solve for } c \text{ and } d.$$

We can solve this, though it is trick at times because not all coefficients that arise are relatively prime to 26. We get

$$\begin{pmatrix} 17 & 5 \\ 18 & 23 \end{pmatrix}$$ for the decryption matrix.

We solve the first system:

$$10(6a+25b) = \overset{10\times}{19} \bmod 26$$
$$25(25a+10b) = 7\times 25 \bmod 26$$

after we reduce mod 26, this is:

$$8a + 16b = 7 \bmod 26$$
$$-\quad a + 16b = 18 \bmod 26$$

$$7a = 15 \bmod 26.$$

Then $7^{-1} = 15 \bmod 26$, so

$$a = 15^2 = 17 \bmod 26.$$

So $25\cdot 17 + 10b = 7 \bmod 26$

$$\Rightarrow 10b = 24 \bmod 26$$

No unique solution since $\gcd(26, 10) = 1$. We have:

$10 \cdot 5 = 24 \bmod 26$

$10 \cdot 18 = 24 \bmod 26$, so $b = 5$ or $18 \bmod 26$.

So try the second congruence. Set $a = 17$, then

~~$25\cdot 17 + 10b = 7 \bmod 26$~~

$$6\cdot 25 + 25b = 19 \bmod 26$$

$\Rightarrow 25b = 21 \bmod 26$, and $25^{-1} = 25. \bmod 26.$

$$\Rightarrow b = 25\cdot 21 \bmod 26$$
$$= 5 \checkmark.$$

So we have a, b. Similarly for c, d.

Crypto

§ 5.3 Running key ciphers.

Here is an improvement of the Vignère cipher.

A running key cipher is a Vignère cipher where the length of the key is equal to the length of the plaintext.

Example:

Plaintext : we will infiltrate their treehouse at dawn

key stream : Th isis thesuperse cretp asswordth at they.

ciphertext : pleat dbuja fivrl ivyib gtjwa rfxll amwha

This sort of method is not susceptible to the Kasiski test. Repetitions in the ciphertext tell us nothing of the key.

Idea : If the key is very long, in order to remember it it typically has to be some sort of meaningful text. If this is the case, then the key (if in english) must've been about 12% e's. So, if you subtract "e" from your ciphertext, you should get ~12% correct letters.

Example continued:

We found ciphertext:

iphertext	PLEAT	DBUJA	FIVRL	IVYIB	
ub tract	EEEEE	EEEEE	EEEEE	EEEEE ...	
	LHAWP	ZXQⒻW	BEⓇNH	ⒺRUⒺX ...	etc.

And indeed, we have a decent success rate.

Better attack: If the keystream is English, try subtracting common English words. Presumably if you try subtracting "the" or "that" from every possible position, ~~some~~ positions will make more sense than others:

Example ct'd Try subtracting "THAT" from every 4-letter block of ciphertext.

block	PLEA	LEAT	EATD	
"THAT"	THAT	THAT	THAT	.... etc.
resut	WEEH	SXAA	LTTK	

Continuing, we get a long list of four-letter blocks, most of which are implausible in English. E.g. blocks like "MBVY" "FBBN". Some blocks are reasonable, though, like "SEAT". Note that even implausible blocks could occur, like "MBVY" if the plaintext contained "COMB VYING" for some reason. So it's a judgement call.

Choosing to assume that "SEAT" is a correct block, you can move on and try subtracting "THE".
If you do this, you get

ciphertext	LL AMWHAL	
keystream guess	TH ATTHE $\textcircled{Y}$	could guess this "Y" next.
plaintext	SE ATDAWN	

now you're well on your way!

Continuing in this manner eventually will work, though it's not something that can reasonably be accomplished by hand.

§5.4 One-time pads.

A <u>one-time pad</u> is a running key cipher where the keystream is a random string of letters.
This cipher <u>cannot be broken</u>! The randomness of the key means that every ~~possible~~ plaintext is equally likely to correspond to a given ciphertext. The only information to be gleaned from the ciphertext is the length of the message - ~~but~~ even the message can be padded.
So the one-time pad is said to have "perfect security"

Unfortunately in order to maintain this perfect security, the key can only be used once.

E.g. Suppose we used a random keystream of $K = YYIVFQPUBV\ldots$ to encrypt both

$$\underbrace{\text{"WE WILL INFILTRATE..."}}_{P1} \text{ and } \underbrace{\text{"BEGIN PREPARATION..."}}_{P2}.$$

and we get the resulting ciphertexts C1 and C2.

Now suppose an eavesdropper gets ahold of both C1 and C2. By subtracting them term-by-term, we get:

$$C1 - C2 = (P1 + K) - (P2 + K) = P1 - P2.$$

ie, think of the first letter. The first letter of C1 is the result of Y+W, the first letter of C2 is Y+B. So the first letter of C1-C2 is

$$(Y+W) - (Y+B) = W-B.$$

The result of this subtraction is therefore the first plaintext P1 encrypted using P2 as a running key! (Almost - there is a sign issue, but this is obviously ~~not~~ a big deal). We can crack this.

Thus, one-time-pads really can only be used once.

- A more serious problem: How to generate a string of random letters or numbers?

This is an entire area of research. The issue is that computers are (mostly) deterministic, so if you know the initial state of a program then you can predict future states (so-called pseudo-random number generators)

- An even more serious problem:
  We've basically been discussing a cryptosystem that completely relies on being able to secretly send one-time keys to your intended recipient. I Guess we need a cryptosystem to do that, now.

- Most serious problem with everything we've discussed so far in the course: Everything we've covered so far is susceptible to a "known plaintext" attack. I.e. if the eavesdropper has the ciphertext <u>and</u> a portion of the plaintext (say the first 20 characters), then deciphering the message is easy. In practice, we <u>need</u> a system resistant to this!

  Eg. We probably want to send computer files that all begin with some known preamble, like
  <html> <DOCTYPE.. etc