Ciphers Where Alice and Bob Need to Meet

Based on notes by William Gasarch

We will use three characters: Alice and Bob who want to communicate secretly, and Eve who wants to see what they are talking about. Alice and Bob do not want Eve to be able to decode their messages.

1. The *Plaintext* is the message you want to send. For example *Discrete Math Plus Plus is the nickname for this CMSC 389*.
2. The *Ciphertext* is the message after it is encoded. For example *EJTDS FUFNB UIQM VTQM VTJTU IFOJD LOBNF GPSDN TD490.* (See note on this below.)
3. If Alice and Bob want to exhange messages then they need to both know SOMETHING ahead of time. What they know is called a *key*. This will be clearer with examples.

Note 0.1 In the above example of coding *Discrete Math Plus Plus is the nickname for this CMSC 389* I used several conventions:

1. I wrote the message in all capitols. This is a standard convention for Eduation- the Plaintext is in normal font, the ciphertext is all capitols.
2. I wrote the message in blocks of five. This is also a standard convention. If I wrote it like *EJTDSFUF NBUI QMVT QMVT JT UIF OJDLOBNF GPS DNTD 490.* then it would be MUCH easier for Eve to decode. Using the blocks-of-five does NOT make it much harder for Alice and Bob.
3. In the above I shifted the letters by 1. This is a standard cipher discussed below. I also shifted the numbers by 1. This is not standard.

1 Shift Cipher

Alice and Bob have wanted to exchange secret messages for the last 4000 years. One of the earliest techniques for this, called the *Caesar Cipher*, operates as follows.

First imagine all letters as numbers. A is 0, B is 1, C is 2, etc, Z is 25. Map every letter to the letter that is three higher (modulo 26). So, the the last three letters shift to the first three. So

A goes to D

B goes to E

$\vdots$

V goes to Y
W goes to Z
X goes to A
Y goes to B
Z goes to C.

More generally, a *shift cipher* is a code where every letter shifts a constant amount.

Lets say that Alice shifts by $s \in \{0, 1, 2, \ldots, 25\}$. We can write this as

$$f(x) = x + s \bmod 26.$$

If Alice uses f what does Bob use to decode? He will use

$$g(x) = x - s \bmod 26.$$

Note that $f(g(x)) = x$. Also note that for ANY choice of s there is a $-s$. We do not actually use $-s$, we use $26 - s$ which accomplishes the same thing.

Are shift ciphers good?

PROS

1. The scheme is easy to describe, easy to code, and easy to decode. So Alice and Bob can operate very fast.

2. Alice and Bob only have to agree on the shift. Since the shift is in $\{1, \ldots, 25\}$, they can easily communicate to each other which shift to use.

CONS

1. The scheme is easy so Eve may spot the pattern.

2. If Eve knows that it is a shift cipher then she can just try all 25 possible shifts. (See later for a fuller explanation.)

3. Alice and Bob do have to meet privately once to agree on the shift. (Is this avoidable?)

Breaking the Cipher:

As noted above Eve could just "try" all 26 possible shifts. But what does that mean? She could, for each shift, decode and see which text "makes sense". But this would be time consuming by hand. Worse- it may be hard to automate. What does it mean to a computer to "make sense?" Is there a better way?

YES! The frequencies of each letter in English is known. (E.g., e is the most common letter). Let p_i be the expected relative frequency of the ith letter in a text. These values are known. Hence $\alpha = \sum_{i=1}^{26} p_i^2$ is known and is 0.065.

Let T be a text. Assume that it was coded with a shift of s. Let q_i be the relative freq of the ith letter in T. Then q_{i+s} is roughly p_i. Hence $I_s = \sum_{i=1}^{26} p_i q_{i+s}$ will be around 0.065. Also, if s' is NOT the shift then $I_{s'} = \sum_{i=1}^{26} p_i q_{i+s'}$ will be around 0.038.

SO, rather than try to see what shift "looks right" or "makes sense" just compute I_s for $s = 1, 2, \ldots, 25$ and whichever one yields something close to 0.065 is the shift. It is likely there will only be one such.

2 Linear Cipher

We can use a more complicated function. For example

$$f(x) = (3x + 4) \bmod 26.$$

If Alice uses f to code, what does Bob use to decode? He needs to use $g(x) = Ax + B \bmod 26$. We need to find A, B that work We need

$$f(g(x)) = x$$

$$f(Ax + B) = x$$

$$3(Ax + B) + 4 = x$$

$$3Ax + 3B + 4 = x$$

We'll set $3B+4 = 0$ and $3A = 1$. AH- can we solve those equations? In both cases we need a number whose mult inverse is 3 mod 26. For now we'll try all possibilities (there are faster ways).

$3 \times 1 = 3$
$3 \times 2 = 6$
$3 \times 3 = 9$
$3 \times 4 = 12$
$3 \times 5 = 15$
$3 \times 6 = 18$
$3 \times 7 = 21$
$3 \times 8 = 24$
$3 \times 9 = 27 \equiv 1$
AH- so 9 is the number we seek.

$$3B + 4 = 1$$

Mult both sides by 9 and reduce mod 27.

$$B + 4 \times 9 = 9$$

$$B + 36 = 9$$

$$B + 10 = 9$$

$$B = -1 = 25$$

And now for A.

$$3A = 1$$

$$A = 9$$

.

So Bob uses $g(x) = 9x + 25$.

Does EVERY linear cipher have an inverse? Lets try $f(x) = 2x$. We need an inverse of 2 mod 26. There isn't one— $2x$ is always even mod 26.

It turns out that so long as the coeff of x is relatively prime to 26 then an inverse exists. So the coeff can be any of $\{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\}$. The constant term can be anything.

Are these codes good?

PROS

1. The scheme is easy to describe, easy to code, and easy to decode (once you know the trick). So Alice and Bob can operate very fast, though not as fast as with the shift cipher.

2. Alice and Bob only have to agree on the multiplier and the shift. This amounts to knowing one number from $\{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\}$ and another from $\{0, 1, \ldots, 25\}$. We represent the numbers in base 2. Each number is 5 bits long, so two numbers take 10 bits. This is small.

CONS

1. The scheme is easy so Eve may spot the pattern, though it's not as easy as the Shift Cipher.

2. If Eve knows that it is a linear cipher then she can just try all $12 \times 26 = 312$ possible linear ciphers. Notice that this is harder than for a shift cipher.

3. Alice and Bob do have to meet privately to agree on the parameters. (Is this avoidable?)

3 Quadratic Cipher

One can look at quadratic ciphers, for example:

$$f(x) = (2x^2 + 5x + 9) \bmod 26.$$

These are called quadratic ciphers. They have similar PROS and CONS to linear ciphers. However there is one more serious CON: given a quadratic polynomial it is hard to determine if it has an inverse on $\{0, \ldots, 26\}$. Note that there are more quadratics than linear function so a PRO is that its harder to crack than a linear cipher.

4 Polynomial Cipher

One can look at any polynomial, for example:

$$f(x) = (2x^8 + 5x^2 + 9) \bmod 26.$$

(Frankly I do not know if this would work.)

The higher the degree the harder for Alice and Bob to tell if it has an inverse AND the harder for Eve to try to decode it.

5 Any Permutation

Alice and Bob pick a *random* permutation of $\{A, \ldots, Z\}$.

Is this a good code?

PROS

1. It seems as though Eve has to try 26! possibilities.

CONS

1. The key Alice and Bob use is a list of the letters of the alphabet in some order. In base 2 this is $26 \times 5 = 130$ bits (though it can be done in somewhat less).

2. Alice and Bob do have to meet in secret to estabish the key. (Is this avoidable?)

3. Eve has to go through ALL 26! possibilities to crack this code! hence this code is UNBREAKABLE!!!!!! (Note- this is NOT TRUE as we will see soon. This is typical of the entire history of crypto which can be summarized as:

   CODE MAKER: I have an unbreakable code!

   CODE BREAKER: I just broke it.

CODE MAKER: Whoops.

Throughout history codes thought unbreakable were broken because the way to break them was unrelated to how they were derived.

6 Frequency Analysis

Given a text that came from a random permuation how can Eve crack it? NOT by going through all 26! possiblities. That would take too long. But note that e is the most common letter in the English Language. th is the most common two-letter pair. People have compiled tables of such information and these can be used to crack a coded text if its long enough.

Also, if Eve knows you are cracking (say) military codes she may use that to her advantage— slightly different patterns may hold.

In old times (say 2000 years ago) it was still fine to use a random cipher since the computation power to do a freq analysis wasn't there yet. But now it is. Hence in the real world today nobody uses any of the ciphers mentioned above.

All of the ciphers discussed so far are mono-sub ciphers, meaning that they map the alphabet letter by letter. Any such cipher can be broken by a Freq Analysis.

7 Matrix Codes

Here is one way to defeat the freq analysis.

Let A be the following matrix.

$$\mathbf{A} = \begin{pmatrix} 8 & 9 \\ 11 & 7 \end{pmatrix}$$

We can map *pairs of numbers* with this matrix as follows. The pair (x, y) will map to the pair you get by applying the matrix and reducing modulo 26, which is

$$((8x + 9y) \bmod 26, (11x + 7y) \bmod 26).$$

From start to finish: take a text, convert the letters to numbers, (assume it has an even number of letters), break the sequence of numbers into blocks of 2 numbers each, and apply the matrix to each pair to get an encoded pair.

Notice that this can be extended to 3×3 matrices or more generally $k \times k$ matrices.

If a 2 by 2 matrix is used then a freq analysis based on pairs may still be possible. Same for 3 by 3. I suspect that at 10 by 10 this code cannot be cracked with freq analysis.

Think about the PROS and CONS.

8 Vigenere Cipher

Here is another way to defeat freq analysis is the *Vigenere Cipher*. Here is an example:

- Every letter that is in a place $\equiv 0 \bmod 5$ is coded by a shift-4.
- Every letter that is in a place $\equiv 1 \bmod 5$ is coded by a shift-15.
- Every letter that is in a place $\equiv 2 \bmod 5$ is coded by a shift-7.
- Every letter that is in a place $\equiv 3 \bmod 5$ is coded by a shift-13.
- Every letter that is in a place $\equiv 3 \bmod 5$ is coded by a shift-1.

Alice could communicate the key to Bob as the sequence (4, 15, 7, 13, 1). Or she could just say DOGMA since D is the fourth letter of the alphabet, O is the 15th, etc. We call DOGMA the Key, and 5 the keylength. They could be any word (or sequence of numbers in $\{0, \ldots, 25\}$) and any length.

How to crack it? We give three methods. Both require long texts. Both involve finding the key length and then doing a freq analysis on the appropriate subtexts (e.g., in the above example you would do a freq analysis on every 5th letter).

Index Test: The *Index of Coincidence* of a text is the prob that two randomly chosen letters are the same. It is known that for an English text this is roughly 0.68, and for a random text this is roughly 0.38. Both of these facts are still true if the text is encoded by a mono-substituition cipher. Both of these facts are still true if you take all of the letters in the place that is $\equiv L' \bmod L$ for any L, L'.

Given a text of length n, let n_i be the number of times the ith letter occurs. The index of coincidence is

$$\sum_{i=1}^{26} \frac{n_i(n_i-1)}{n(n-1)}.$$

Given a text that you think was coded by a Vig cipher you can GUESS that the key length is 2,3,4,... and for each guess form the text that is every 2nd, 3rd, ... letter and see if the IC is close to 0.68 or .038

Kasiski Examination: We first find the key length. Imagine that in the text you see ABDUQ several times. It is likely that the DISTANCE between them is the key length of a multiple of it. So find the differences between the repeated text and then find a common factor for all of them. This gives you a SMALL set of key lengths to look at. Then use the Index Test on those candidates.

9 Vigenere Plus Cipher

In the Vig cipher we used a SHIFT on every Lth letter. We could instead use a Linear, Quadratic, Polynomial, Matrix on every Lth letter.

To my knowledge no such code was ever used. This is just an accident of history. Let Y be a year (I do not know what year it is). Before year Y, using a Vig Plus Cipher would have been two cumbersome for Alice and Bob. After year Y, there were better techniques.

10 An Uncrackable Code: the One-Time Pad

Definition 10.1 If a and b are bits (0 or 1) then $\oplus$ (also written XOR and called "exclusive or") is defined as follows:

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

The following facts are easy to verify.

Fact 10.2 *Let a, b, c be bits.*

1. $(a \oplus b) \oplus c = a \oplus (b \oplus c)$.
2. *For all bits a,* $a \oplus a = 0$.
3. $a \oplus b \oplus b = a \oplus (b \oplus b) = a \oplus 0 = a$.

We now describe the one-time pad.

1. Alice and Bob have to meet (or communicate over a secure channel that Eve cannot listen to) and agree on a randomly generated sequence of bits – a VERY long sequence. Say it's
$$r_1 r_2 \cdots r_N.$$
This is called *the key.* They then part.

2. If (later) Alice wants to send
$$a_1 a_2 a_3 \cdots a_m$$
she sends
$$(r_1 \oplus a_1)(r_2 \oplus a_2) \cdots (r_m \oplus a_m).$$
When Bob gets this string, which he sees as
$$s_1 \cdots s_m$$

he can decode it by taking

$$\begin{aligned}(r_1 \oplus s_1)(r_2 \oplus s_2)\cdots(r_m \oplus s_m) &= (r_1 \oplus (r_1 \oplus a_1))(r_2 \oplus (r_2 \oplus a_2))\cdots(r_m \oplus (r_m \oplus a_m)) \\ &= ((r_1 \oplus r_1) \oplus a_1)((r_2 \oplus r_2) \oplus a_2))\cdots((r_m \oplus r_m) \oplus a_m)) \\ &= a_1 a_2 \cdots a_m\end{aligned}$$

3. If either Alice or Bob wants to send another message they will start with r_{m+1}.

PROS: This is impossible to crack! Since the original key was random, if Eve sees the message

$$s_1 s_2 \cdots s_m$$

it will look random to her.

CONS: N is LARGE! They have to meet and exchange A LOT of information. In fact, if they plan to later communicate N bits they need to have a key of length N.

PROBLEM: Can Alice and Bob use a shorter key?

PROBLEM: Can Alice and Bob agree on a secret key (e.g., $r_1 r_2 \cdots r_N$) without having to meet?