Introduction

Secret writing has been used for thousands of years; probably for as long as we have had secrets to keep. These may be messages in war, messages between corporations, or just personal secret messages. Even languages may be thought of as a kind of code.

When we send secret messages there are three types of participants. The sender, the intended recipient or receiver, and possibly the interceptor or ‘bad guy’. The act of disguising your message is known as encryption, and the original message is called the plaintext. The encrypted plaintext is known as the ciphertext (or cryptogram), and the act of turning the ciphertext back into the original plaintext is decryption.

Steganography is the act of physically hiding your message; in ancient times you might write your message on the shell of a boiled egg using special ink that would soak through the porous outer shell. When the egg was cracked and peeled, your secret message would be found written on the egg white. In the 20th century, agents involved in espionage would use the microdot, shrinking their entire message smaller than could be seen by the naked eye.

Codes on the other hand involve turning words or phrases into other words. For example, a command such as ‘attack at midnight’ might simply become ‘eagle’. The problem with codes is that they involve carrying a code book, essentially a dictionary, to translate what you want to say into code. If this book falls into enemy hands then your code is revealed and is now useless, and replacing each code book is not something that can be done frequently. Also, a code may not have the flexibility to deal with all messages you might conceivably wish to send.

In comparison, ciphers work on the level of the individual letters of your message. Ciphers may replace letters in a message with other letters, or numbers, or symbols. For example, if ‘a’ becomes 0, ‘b’ becomes 1, ‘c’ becomes 2 and so on, then a word like ‘secret’ becomes ‘18 4 2 17 4 19’. (Note, counting from ‘a’ as zero is a necessary convention that we will continue to use later on). This provides a much greater flexibility in our messages, and it will be ciphers that we will be dealing with during this course.

Ciphers comes in two parts: The first part is the algorithm; this is simply the method of encryption. The second part is the key. The idea is these work together like a lock-and-key and, for the code breaker, having one without the other is just half the problem. The key may change frequently meaning the greater the number of keys the more difficult the cipher becomes break by brute force (an exhaustive check of all possible keys).

Secret writing has always been a constant struggle between the code maker and the code breaker. Cryptography is the study of making secret messages, whereas cryptanalysis is the study of breaking those secret messages. Cryptology is the collective name for both cryptography and cryptanalysis; the word cryptology coming from the Greek words kryptos meaning hidden and logos meaning word.
1 Monoalphabetic Ciphers

1.1 The Caesar Shift

We begin with a simple cipher as described by Julius Caesar in the Gallic Wars. In it he describes taking two alphabets, from a to z, one above the other. However, the second alphabet is shifted three places to the left. The top alphabet represents letters in the plaintext, while the the alphabet underneath shows what these letters become in the ciphertext. Note, throughout this course I will represent the plaintext in lowercase letters, and the ciphertext in uppercase letters.

| plaintext | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| ciphertext | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |

So, a plaintext message like

veni, vidi, vici

becomes

YHQL, YLGL, YLFL

in the ciphertext.

Clearly, the shift does not have to be 3, (it can be 4, 5, etc). The idea of shifting the alphabet is the algorithm, and the size of the shift is the key. This is not a very secure cipher since there are only 26 keys (including plain English when you shift by 26). This cipher can be quickly broken by a simple exhaustive check of all 26 possible keys. (The original strength of this cipher may have come from Caesar’s enemies not realising he was using a cipher at all).

A set is a collection of objects (numbers, letters, symbols, or other objects). Let \( \mathcal{A} \) denote the set of letters alphabet a, b, c, ..., z. Let \( \mathcal{P} \) denote the alphabet of the plaintext, usually \( \mathcal{P} = \mathcal{A} \). Let \( \mathcal{C} \) denote the ciphertext alphabet, this set may or may not be the same as \( \mathcal{A} \).

A monoalphabetic cipher is a rule which associates each letter of \( \mathcal{P} \) with a unique letter in \( \mathcal{C} \). Furthermore, different letters in \( \mathcal{P} \) are sent to different objects in \( \mathcal{C} \). This means the size of \( \mathcal{C} \) is equal to or greater than the size of \( \mathcal{P} \), and each letter in \( \mathcal{P} \) is paired with an image in \( \mathcal{C} \). This will allow us to define an inverse rule from \( \mathcal{C} \) to \( \mathcal{P} \) that will decrypt our messages from the ciphertext back to the original plaintext.

Below is a diagram using the first five letters of the alphabet. The first is a valid example of a monoalphabetic cipher. The second is not valid because one letter in the plaintext alphabet is sent to two different letters in the ciphertext alphabet. Finally, the third is not valid because two letters in the plaintext alphabet are sent to the same letter in the ciphertext alphabet.
1.2 Modular Arithmetic

In the Caesar Shift we saw how a shift of 3 caused the cipher letters A, B, C to wrap around at the end. Now, instead of the letters let’s use the numbers 0 to 25, where a becomes 0, b becomes 1, and so on until z becomes 25.

Then a shift of 3 is equivalent to adding 3 to each number, where 23 + 3 = 0, 24 + 3 = 1 and 25 + 3 = 2.

This is called *modular arithmetic*. It works like a clock, where (for a 12 hour clock) if you start at 11 o’clock and add three hours it’s 2 o’clock. However there is nothing special about the number twelve, in fact we could imagine a clock of any size and for ciphers we will mostly be imagining a clock of size 26. For this reason modular arithmetic is sometimes called *clock arithmetic*.

**Definition.** Given integers $a$ and $b$ and positive integer $m$, we say that $a$ and $b$ are *congruent modulo m* if $m$ strictly divides $(a - b)$, i.e. leaves no remainder, and write $m|(a - b)$ or $a \equiv b \mod m$. The number $m$ is called the *modulus* in this situation.

**Example.** If we set $a = 28$, $b = 13$ and $m = 5$, then $5|(28 - 13) = 15$ or $28 \equiv 13 \mod 5$.

The following properties hold:

(i) $a \equiv b \mod m$ if and only if there exists an integer $q$ such that $a = qm + b$; eg, $28 = 3(5) + 13$;

(ii) If $a \equiv b \mod m$ then $b \equiv a \mod m$; eg, $13 = -3(5) + 28$;

(iii) The numbers congruent to $b$ modulo $m$ are . . . , $-2m + b, -m + b, b, m + b, 2m + b, \ldots$; eg, numbers congruent to 13 mod 5 are . . . , $-7, -2, 3, 8, 13, 18, 23, \ldots$;

(iv) If $a = qm + r$, with $0 \leq r < m$, then $r$ is the remainder of $a$ on division by $m$ and $a \equiv r \mod m$; eg, $28 \equiv 3 \mod 5$;

(v) $a \equiv b \mod m$ if and only if $a$ and $b$ have the same remainder on division by $m$; eg, 28 and 13 both leave remainder 3 after division by 5;
(vi) Any integer is congruent modulo $m$ to exactly one of $0, 1, \ldots, m - 1$, as these are all the possible remainders on division by $m$; if $m = 5$ then all integers are congruent to one of $0, 1, 2, 3, 4$ modulo $5$.

(vii) If $a \equiv b \mod m$ and $b \equiv c \mod m$, then $a \equiv c \mod m$ because $a, b$ and $c$ all have the same remainder on division by $m$. In this case we may write $a \equiv b \equiv c \mod m$; eg, $28 \equiv 13 \equiv 3 \mod 5$;

(viii) If $a \equiv b \mod m$ and $c \equiv d \mod m$, then $a + c \equiv b + d \mod m$ and $ac \equiv bd \mod m$; eg, If $28 \equiv 3 \mod 5$ and $-19 \equiv 1 \mod 5$ then $28 - 19 \equiv 3 + 1 \equiv 4 \mod 5$ and $(28)(-19) \equiv (3)(1) \equiv 3 \mod 5$.

1.3 Additive Ciphers

The Caesar Shift is also known as an additive cipher. As we know, there are 26 possible keys. If $0 \leq p \leq 25$ is the value of your plaintext letter, $0 \leq b \leq 25$ is the value of your key, then the value $c$ of your ciphertext letter is given by;

$$c = p + b \mod 26$$

Example. If we encipher the message ‘I like the park’ using a key $b = 16$ we get;

<table>
<thead>
<tr>
<th>plaintext: i l i k e t h e p a r k</th>
</tr>
</thead>
<tbody>
<tr>
<td>in numbers: 8 11 8 10 4 19 7 4 15 0 17 10</td>
</tr>
<tr>
<td>$p + b \mod 26$: 24 1 24 0 20 9 23 20 5 16 7 0</td>
</tr>
<tr>
<td>ciphertext: Y B Y A U J X U F Q H A</td>
</tr>
</tbody>
</table>

To decrypt a cipher letter $c$ we need to find the decryption algorithm, or decryption key, to obtain the original plaintext letter $p$. If our message was sent using an additive cipher we simply subtract the key-value $b$; or equivalently we need to find the additive inverse $0 \leq b' \leq 25$ such that $b + b' = 0$.

Proposition 1.3.1 If $0 \leq b \leq 25$ is the encryption key for an additive cipher, then the decryption key $b' = 26 - b$.

Proof. If $c$ is our cipher letter, then

$$c + b' = c + (26 - b) \equiv (p + b) + (26 - b) \equiv p + 26 \equiv p + 0 \equiv p \mod 26.$$ 

Hence, the additive inverse of $b$ modulo 26 is $(26 - b)$.

(Note, the term theorem is reserved for important results, the term proposition is used for smaller results. Proofs are traditionally set out this way, beginning with ‘Proof’, and terminating with a small box to show the proof is finished).
1.4 Common Divisors

Given two non-zero integers $x$ and $y$, we call $z$ a common divisor (or common factor) if $z$ is an integer that exactly divides $x$ and $y$ with no remainder, i.e. $z|\!\!|x$ and $z|\!\!|y$. The largest such divisor is called the greatest common divisor (highest common factor).

**Example.** The divisors of $x = 18$ are $\pm1, \pm2, \pm3, \pm6, \pm9, \pm18$. The divisors of $y = 30$ are $\pm1, \pm2, \pm3, \pm5, \pm6, \pm10, \pm15, \pm30$. The common divisors of $x$ and $y$ are $\pm1, \pm2, \pm3, \pm6$ and the greatest common divisor is 6.

We call two non-zero integers $x$ and $y$ coprime if they share no common divisors other than $\pm1$, i.e. their greatest common divisor is 1, or equivalently there exists integers $s$ and $t$ such that $sx + ty = 1$.

**Example.** The integers 21 and 26 are coprime because they share no common divisors, or equivalently $(5)(21) + (-4)(26) = 1$.

The second definition of coprime is an equally valid, however the proof is not completely trivial so we will leave that for a future course in number theory. We will, however, be using this definition repeatedly throughout this course.

1.5 Multiplicative Ciphers

Instead of adding a key-value to our plaintext, we could multiply by a key-value $a$, such that $c = ap \mod 26$.

For example; if $a = 4$ then

<table>
<thead>
<tr>
<th>plaintext:</th>
<th>i</th>
<th>l</th>
<th>i</th>
<th>k</th>
<th>e</th>
<th>t</th>
<th>h</th>
<th>e</th>
<th>p</th>
<th>a</th>
<th>r</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>in numbers:</td>
<td>8</td>
<td>11</td>
<td>8</td>
<td>10</td>
<td>4</td>
<td>19</td>
<td>7</td>
<td>4</td>
<td>15</td>
<td>0</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>$ap \mod 26$:</td>
<td>6</td>
<td>18</td>
<td>6</td>
<td>14</td>
<td>16</td>
<td>24</td>
<td>2</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td>16</td>
<td>14</td>
</tr>
<tr>
<td>ciphertext:</td>
<td>G</td>
<td>S</td>
<td>G</td>
<td>O</td>
<td>Q</td>
<td>Y</td>
<td>C</td>
<td>Q</td>
<td>I</td>
<td>A</td>
<td>Q</td>
<td>O</td>
</tr>
</tbody>
</table>

However, we cannot pick any value for $a$. In the example above, by choosing $a = 4$, the letters ‘e’ and ‘r’ in the plaintext are both encrypted as the letter ‘Q’. This violates the definition of a monoalphabetic cipher and would make decryption very hard, or impossible, to do. Which values for $a$ work?

In fact we must choose a value that is coprime to 26, i.e. shares no common factors with 26. In general, if the alphabet has size $m$, then $a$ must be coprime to $m$.

Here, the value $a = 4$ fails because it shares a common divisor with 26, namely the common divisor 2. This will cause more than one letter to be sent to the same value. For example, $4 \times 0 = 4 \times 13 \equiv 0 \mod 26$. 
Proposition 1.5.1 Let integers $a$ and $m$ be coprime, such that $sa + tm = 1$ for some integers $s, t$. Then there exists $1 \leq a' \leq m$ such that $aa' = 1 \mod m$. If $a$ is the encryption key for a multiplicative cipher, then the decryption key is $a' \equiv s \mod m$.

Proof. If $sa + tm = 1$, then $tm \equiv 0 \mod m$, and

$$sa + tm \equiv sa \equiv 1 \mod m.$$ 

Hence the multiplicative inverse of $a$ is $a' \equiv s \mod m$.

If $p$ is our plaintext letter, and $c = ap$ is our cipher letter, then;

$$a'c \equiv sc \equiv sap \equiv (1)p \equiv p \mod m.$$ 

as required.

The value $a'$ is the multiplicative inverse of $a$ modulo $m$. This will always exist if the key-value $a$ and the alphabet size $m$ are coprime.

If $a$ is the key-value of a multiplicative cipher, then the enciphered letters are $0, a, 2a, 3a, \ldots, (m - 1)a \mod m$. If $a$ is coprime to $m$ then by multiplying by $a' = s$ we get the values $0, 1, 2, 3, \ldots, (m - 1) \mod m$. Since the second list has all different values, the first list must have all different values, and we have a valid monoalphabetic cipher.

There are only 12 valid key-values coprime to $m = 26$, namely 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25.

Example. If $a = 21$ then the message ‘cab’ becomes ‘QAV’. To decrypt we need to find the multiplicative inverse of $a$. Earlier, we worked out that the multiplicative inverse was 5. So, to decrypt, we multiply by 5 (modulo 26).

1.6 Affine Cipher

An affine cipher is a multiplicative cipher followed by an additive cipher. If $1 \leq a \leq 25$ is coprime to 26, and $0 \leq b \leq 25$ then;

$$c = ap + b \mod 26.$$ 

For each of the 12 valid values for $a$ we have 26 choices for $b$. So altogether there are $12 \times 26 = 312$ possible affine ciphers, for an alphabet of size 26.
1.7 General Substitution Ciphers

Remember, a monoalphabetic cipher is a one-to-one rule that pairs each letter in the plaintext alphabet with a letter in the ciphertext alphabet. However, not all monoalphabetic ciphers need to be defined by a formula, like those formulas we have seen for the additive, multiplicative and affine ciphers. Instead, we may define a cipher letter-by-letter, by defining where each letter in the plaintext alphabet is sent in the ciphertext alphabet.

Example. Let’s use the first five letters a, b, c, d, e. Then we may define a cipher as:

\[
\begin{align*}
a &\rightarrow C; \\
b &\rightarrow A; \\
c &\rightarrow B; \\
d &\rightarrow E; \\
e &\rightarrow D.
\end{align*}
\]

We may represent this with a diagram:

So a plaintext message such as ‘bead’ becomes ‘ADCE’ in the ciphertext.

To decrypt this message we simply reverse the direction of the arrows:

We may encrypt a message twice, performing one encryption after another, making a composite of ciphers. We can represent this by juxtaposing the two diagrams. We may represent the the result on one diagram by following the total path from left to right for each letter. The result is a new monoalphabetic cipher, however the new cipher is no more secure than the original.
How many monoalphabetic ciphers exist? To calculate this answer consider your choices for each step. Building our cipher from scratch, we have 26 choices for the first letter of the cipher. We then have 25 choices left for the second letter, and 24 choices for the third letter, and so on. Continue this way until we reach the final letter of the cipher, for which we will have one choice left that we have to take.

\[
\text{plaintext} \begin{array}{cccccccccccccccccccccccc}
a & b & c & d & e & f & g & h & i & j & k & l & m & n & o & p & q & r & s & t & u & v & w & x & y & z \\
\end{array}
\]

To find out the total number of keys, i.e. the total number of combinations of A to Z, we multiply choices together. Multiplying the integers 1 to \( n \) inclusive is called \( n \) factorial and is written \( n! \) so

\[
n! = 1 \times 2 \times \cdots \times (n - 1) \times n.
\]

Hence, the total number of key is 26! and is approximately equal to \( 4 \times 10^{26} \) - a very large number indeed. And too many keys to check by brute-force alone.

### 1.8 Breaking the Cipher: Frequency Analysis

So far we have looked at the roles of the sender and receiver. Now, how will an an interceptor break a monoalphabetic cipher without the key?

Given a long piece of text, one can determine the frequency of each letter. In general, the frequencies of the letters in a given text do not change. Below is the expected frequency of each letter in English:

| letter | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| frequency (%) | 8.2 | 1.5 | 2.8 | 4.2 | 12.7 | 2.2 | 2.0 | 6.1 | 7.0 | 0.1 | 0.8 | 4.0 | 2.4 | 6.7 | 7.5 | 1.9 | 0.1 | 6.0 | 6.3 | 9.0 | 2.8 | 1.0 | 2.4 | 2.0 | 0.1 | 0.1 |
The letters ‘e’, ‘t’ and ‘a’ are the most common, with ‘j’, ‘q’, ‘x’ and ‘z’ the least common. Naturally, frequencies would be different in different languages. Also, frequencies are slightly different depending on the type of message it is, i.e. personal, military or other.

In a monoalphabetic cipher, if the plaintext letter ‘e’ becomes a different letter, say ‘w’, then ‘w’ will now be the most common letter in the ciphertext. The underlying frequencies of the letters remain unchanged, and this is a clue to help us break the cipher. This is called frequency analysis.

And there are other tricks we can use: Look for common words like ‘the’ or ‘and’. Look for letters at the ends of words that might be ‘s’. Look for double letters that might be ‘oo’ or ‘tt’; or certain pairs of letters, called bigrams, that might be ‘th’ or ‘qu’. Below are the ten most common bigrams and trigrams.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>th</td>
<td>er</td>
<td>on</td>
<td>an</td>
<td>re</td>
<td>he</td>
<td>in</td>
<td>ed</td>
<td>nd</td>
<td>ha</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>the</td>
<td>and</td>
<td>tha</td>
<td>ent</td>
<td>ion</td>
<td>tio</td>
<td>for</td>
<td>nde</td>
<td>has</td>
<td>nce</td>
<td></td>
</tr>
</tbody>
</table>

Example. Let’s use frequency analysis to break:

VCIL NCI VWKIU WDI DSXLT AI HDIWQELB WU E FWYI NCI TTYQ WPSLI
WLT AZ IZI EL KWEL EU UIIQELB USAI BDIIL PIWJ NS DIUN XFSI
VCWN VSPX PT LSN E BEKI NS VWLTID VCIDI AZ SPT YSAFWLES LU TVIPP
WHUIL YI AWQI U NCI CIWDN BSDV JSLTID EUPI SJ HIWXNZ JW DI NCII VIPP

The ciphertext is 199 letter long, and the individual frequencies are given by:

| letter | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| frequency (%) | 3 | 3 | 4 | 6 | 5 | 2 | 0 | 2 | 1 | 8 | 2 | 2 | 8 | 0 | 6 | 0 | 5 | 2 | 0 | 7 | 4 | 5 | 9 | 2 | 2 | 2 |
The most common letter is ‘I’ which suggests it is the letter ‘e’ in the plaintext. The most common trigram is ‘NCI’, so maybe this is the word ‘the’. Continuing in this way we will get the final message:

```
when the waves are round me breaking as i pace the deck alone
and my eye in vain is seeking some green leaf to rest upon
what would not i give to wander where my old companions dwell
absence makes the heart grow fonder isle of beauty fare thee well
```

1.9 Transposition Ciphers

So far we have looked at ciphers that substitutes a letter in the plaintext alphabet for a letter, number or symbol in the ciphertext alphabet. Collectively these are known as substitution ciphers. However, there is another type of cipher, known as transposition ciphers, this is when you mix-up the position of the letters of the message, to form an anagram.

For example, I am going to send the message “this is a railfence cipher”. I write the message as two rows:

```
  t h i s i s a r a i l
  f e n c e c i p h e r
```

We now scramble up the order of the letters by taking the columns of the messages to make “tfhein-sciescairpahielr”. To decipher the message we need to reverse the procedure, in other words, we write the cipher in columns of length 2. The original message is then written in two rows.

In general, we can rearrange the positions of the letters any way we want. For a message of length \(n\), number the positions from 1 to \(n\). We may then represent the transposition using a diagram. For example, we may rearrange the word “eggs” as follows

```
1  1
2  2
3  3
4  4
```

to make the cipher “gsge”. It’s scrambled eggs!

Notice we may use the same ideas that we used for substitution ciphers for transposition ciphers. The only difference is substitution ciphers act on the plaintext letter’s value (an alphabet of size \(m\), usually the letters a to z), while transposition ciphers act on the plaintext letter’s position (from 1 to \(n\)).
1.10 Commuting Ciphers

We have seen that enciphering a message twice using two general substitution ciphers will make a new general substitution cipher - but not necessarily a more secure one! But does order matter? If I change which cipher I apply first, and which cipher I apply second, will the results be different? Earlier we saw this example:

```
You see the results are different. This is true in general.
```

On the other hand, if the order of the two ciphers do not matter (both orders give the same result) we say the ciphers commute. For example, two additive ciphers or two multiplicative ciphers will commute.

One of the major problems in cryptography is how to send the key. Traditionally both the sender and the receiver needs to know the key so they may encipher and decipher the message. This means the key needed to be sent secretly and securely. One solution to this problems is to use two ciphers that commute.

Imagine Alice wants to send a message to Bob. Alice and Bob each have their own cipher, called $A$ and $B$ respectively, and these two ciphers commute. First, Alice applies her cipher to the plaintext $p$, written $A(p)$, and sends the message to Bob. Bob does not know how to decrypt this message, so encrypts it again using his cipher, written $B(A(p))$ and sends it back to Alice. Alice now has a message that has been enciphered twice! But since the ciphers commute we have $B(A(p)) = A(B(p))$. If Alice applies her decryption key to this message it will remove her cipher, leaving only Bob’s encryption $B(p)$. Finally, if Bob applies his decryption key he will be able to read the original message from Alice.
Every time the message was sent it was enciphered, using either Alice’s key, Bob’s key, or both. Yet, the whole exchange took place without the two individuals exchanging keys! If you can find two ciphers that commute, they may be used in this way as a solution to the key distribution problem.

Remember, each letter in the plaintext has two properties, value and position. Substitution ciphers act on value, while transposition ciphers act on position. But these two components do not interact, meaning if we use a substitution cipher and then rearrange the letters, we will get the same result as if we had rearranged the letters first, then applied the substitution cipher. In other words, substitution ciphers and transposition ciphers commute and can be used in the exchange described above.

For example, if Alice uses the general substitution cipher from section 1.7, then the word ‘bead’ becomes ‘ADCE’. Bob receives the message and uses the transposition cipher from in section 1.9, so ‘ADCE’ becomes ‘CEDA’. Alice receives this message and applies her inverse substitution cipher, so ‘CEDA’ becomes ‘ADEC’. Finally, Bob receives the message and applies his inverse transposition cipher, so ‘ADEC’ becomes ‘bead’.

Although no key is exchanged in this method, it is still only as strong as it’s weakest link. With the first and third pass only encrypted once, standard techniques may be applied to break the cipher. In other cases all three passes may be used to determine the original message. For example, if we used additive ciphers to encrypt our message then if the someone intercepts the three passes \(c_1 = p + a\), \(c_2 = c_1 + b\) and \(c_3 = c_2 - a\) then that person may use \(c_1 - c_2 + c_3\) to recover the original message.
2 Polyalphabetic Ciphers

Monoalphabetic ciphers were used for thousands of years, but were vulnerable to statistical attacks, until a new idea in encryption began to be adopted in order to disguise letter frequencies. A polyalphabetic cipher is a rule that uses a different monoalphabetic cipher to encipher a plaintext letter $p$ depending on its position in the plaintext. This means the same letter appearing in two different positions in the plaintext may be enciphered in two different ways, so a double letter in the plaintext may not necessarily be a double letter in the ciphertext.

2.1 Vigenère Cipher

The following cipher is named after the 16th century French diplomat Blaise de Vigenère. Imagine we want to encrypt the message: ‘shaken not stirred’.

We start by constructing a Vigenère Square:

Notice, each line of the Vigenère Square is a Caesar Shift, starting with a shift of zero, and ending with a shift of 25.

Next we need a **keyword**. Let’s use the word **BOND**. We write that repeatedly above our message:

Keyword: **B O N D B O N D B O N D B O N D**

Plaintext: **s h a k e n n o t s t i r r e d**

To encrypt the first letter: go to row ‘B’ in the square - the first letter of the keyword; next find column ‘s’ - the first letter of the plaintext; and where they meet in the middle is the first letter of the ciphertext ‘T’. For the next letter do the same again: go to row ‘O’ and column ‘h’, and where they meet is the second letter of the ciphertext ‘V’. Continuing in this way we get:
Keyword: BOND BOND BOND BOND BOND
Plaintext: shaken not stirred
Ciphertext: TVNNFBAUGGLSFRRG

Notice, the same letter in the plaintext may now be enciphered as different letters in the ciphertext depending on its position. Also notice, two letters that are the same in the ciphertext may come from two different letters in the plaintext.

If the keyword has length $l$, let $k_1, k_2, \ldots, k_l$ be the letters of the keyword. If $p_i$ is the plaintext letter in the $i$th position, then the ciphertext letter in the same position, $c_i$, is given by

$$c_i = p_i + k_i \mod l \mod 26.$$

Given the keyword, we may decrypt our ciphertext by simply reversing this algorithm.

The effect of the Vigenère cipher is to disguise the frequencies. The longer the keyword the more effect it has on frequency.

2.2 Breaking the Vigenère Cipher

If the keyword used in the Vigenère cipher has length $l$, then every $l$th letter will a result of the same monoalphabetic cipher. If we can determine the length of the keyword then, given a long enough message, we will be able to use standard cryptanalysis tools for monoalphabetic ciphers on every $l$th letter to decrypt the whole message.

If monoalphabetic ciphers are broken by looking for common letters, polyalphabetic ciphers are broken by looking for common words.

Example. Below is ciphertext, 188 letters long, believed to have been enciphered using a Vigenère cipher:

```
FHVDWSEEMFQCZXVQRZAOBOCGOXPYIPQTZKQUPYMFZADMIRMFKMFFHVAVWJTVMBBHTMBFUIGTDDEEKVPIGTCYAKJZMIJMRQVZOSZEIMODZBKMSVDSZTLIZYXZCWEEBVD
```

We begin our cryptanalysis by looking for any string of letters that appears more than once in the ciphertext. If we look closely, we can see the string ‘FHV’ appearing three times in the ciphertext. Since the keyword is repeated, we proceed by assuming this is a common trigram that just happens to appear under the same three letters of the keyword.

Notice, the distances between the occurrences of ‘FHV’ is 50 letters, followed by another 100 letters.

The Kasiski Test: If a string of letters appear repeatedly in a polyalphabetic cipher, then the length of the keyword, $l$, may be a common divisor of the distances between occurrences.

So in our example above, the keyword may have length 50, 25, 10, 5 or 2.
2.3 Probability and The Friedman Test

We start with some definitions. The sample space $S$ of an experiment or random trial is the set of all possible outcomes. An event $E$ is any subset of the sample space; a collection of outcomes within the sample space.

If all outcomes in the sample are equally likely, the probability of $E$, written $P(E)$, is given by

$$P(E) = \frac{\text{Number of outcomes in } E}{\text{Total number of outcomes in } S}$$

**Example.** Let’s pick a card from a pack of 52 cards. Each outcome is equally likely with a probability of $1/52$. Then;

$$P(\text{picking an ace}) = \frac{4}{52}$$
$$P(\text{picking a spade}) = \frac{13}{52}$$
$$P(\text{picking a red card}) = \frac{26}{52}$$

Given an event $E$, the probability of not $E$ is

$$P(\text{not } E) = 1 - P(E).$$

**Example.**

$$P(\text{not picking an ace}) = 1 - P(\text{picking an ace})$$
$$= 1 - \frac{4}{52}$$
$$= \frac{12}{13}$$

For any two events $E$ and $F$ of a sample space $S$,

$$P(E \text{ or } F) = P(E) + P(F) - P(E \text{ and } F).$$

If $P(E \text{ and } F) = 0$ we say $E$ and $F$ are *disjoint events*.

**Example.**

$$P(\text{picking an ace or a spade}) = P(\text{picking an ace}) + P(\text{picking a spade}) - P(\text{picking ace of spades})$$
$$= \frac{4}{52} + \frac{13}{52} - \frac{1}{52}$$
$$= \frac{16}{52}$$
$$= \frac{4}{13}$$

We subtract $P(\text{picking ace of spades})$ to prevent counting it twice.
Since a card being a spade and a red card are disjoint events, \( P(\text{picking a spade and a red card}) = 0 \) and;

\[
P(\text{picking a spade or a red card}) = P(\text{picking a spade}) + P(\text{picking a red card}) \\
= \frac{13}{52} + \frac{26}{52} \\
= \frac{39}{52} \\
= \frac{3}{4}
\]

If \( P(E \text{ and } F) = P(E)P(F) \) we say \( E \) and \( F \) are independent events.

Example.

\[
P(\text{picking a red ace}) = P(\text{card is red and ace}) \\
= P(\text{picking a red card})P(\text{picking an ace}) \\
= \frac{26}{52} \times \frac{4}{52} \\
= \frac{2}{52} \\
= \frac{1}{26}
\]

The probability of picking two aces without replacement;

\[
P(\text{picking two aces}) = P(\text{picking an ace on first trial})P(\text{picking an ace on second trial}) \\
= \frac{4}{52} \times \frac{3}{51} \\
= \frac{12}{2652} \\
= \frac{1}{221}
\]

Now, let \( n \) be the length of our ciphertext. Let \( n_0 \) be the number of ‘a’s in the ciphertext, \( n_1 \) be the number of ‘b’s and so on until \( n_{25} \), the number of ‘z’s in ciphertext. Picking letters at random, the probability of picking two ‘a’s is given by

\[
P(\text{picking two ‘a’s}) = P(\text{picking ‘a’ on first trial})P(\text{picking ‘a’ on second trial}) \\
= \frac{n_0}{n} \times \frac{n_0 - 1}{n - 1} \\
= \frac{n_0(n_0 - 1)}{n(n - 1)}
\]

If \( n \) is large, then \( n \approx n - 1 \) and \( P(\text{picking two ‘a’s}) \approx (n_0/n)^2 \), where \( n_0/n \) is the frequency of ‘a’ in the ciphertext.
The index of coincidence, IC, is the probability of picking any two letters the same;

\[
\text{IC} = P(\text{picking two letters the same})
\]

\[
= P(\text{two 'a's}) + P(\text{two 'b's}) + \cdots + P(\text{two 'z's})
\]

\[
= \sum_{i=0}^{25} \left( \frac{n_i}{n} \right)^2
\]

where the Greek letter Σ means the sum over all \(n_i\).

Example. Using the frequency table in section 1.8 we can calculate the index of coincidence for English plaintext

\[
\text{IC} \approx (0.082)^2 + (0.015)^2 + \cdots + (0.001)^2 = 0.065.
\]

Note, since a monoalphabetic cipher does not change the underlying frequencies, we can expect ciphertext encrypted with a monoalphabetic cipher to have the same IC value of 0.065.

On the other hand, if the frequencies of each letter in the ciphertext is 1/26 then the distribution will be perfectly uniform, and the index of coincidence IC \(\approx 26(1/26)^2 = 0.038\).

The index of coincidence is a measure of how strong your cipher is. A value close to 0.065 suggests a monoalphabetic cipher, while the closer the value is to 0.038 the closer the distribution is to being uniform, and suggests a polyalphabetic cipher.

Example. The Vigenère cipher above has length \(n = 188\) and the following frequencies

| letter | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| frequency (%) | 3.7 | 3.7 | 2.1 | 3.7 | 5.6 | 6.4 | 3.7 | 2.7 | 4.8 | 2.1 | 5.3 | 1.6 | 8.5 | 0 | 2.7 | 5.3 | 2.7 | 3.7 | 3.7 | 3.2 | 2.1 | 6.9 | 2.1 | 2.1 | 2.1 | 8.5 |

Giving an index of coincidence IC \(\approx 0.049\), suggesting it is a polyalphabetic cipher.

If the keyword in our polyalphabetic cipher has length \(l\), then writing the ciphertext in rows of length \(l\) will make an array with approximately \(n/l\) rows.

\[
\begin{array}{cccccccccccccccccccccccc}
   k_1 & k_2 & k_3 & \cdots & k_{l-1} & k_l \\
k_1+1 & k_2+1 & k_3+1 & \cdots & k_{2l-1} & k_{2l} \\
\vdots  & \vdots  & \vdots  & \ddots & \vdots  & \vdots  \\
k_{n-l-1} & k_{n-l} & \cdots & k_{n-3} & k_{n-2} & k_{n-1} & k_n.
\end{array}
\]

Each column will be the result of one monoalphabetic (Caesar) cipher. Now we can find \(l\) by using a different method to calculate the index of coincidence:
Pick two letters at random from the ciphertext. The probability the first letter is from column 1 is $1/l$.

And, for large $n$, the probability they both come from the column 1 is approximately $(1/l)^2$.

So, picking two letters at random, the probability they both come from the same column is $l \times (1/l)^2 = 1/l$. Now we can calculate the index of coincidence again as follows:

$$IC = P(\text{picking two letters the same})$$

$$= P(\text{picking two letter from same column})P(\text{those two letters are the same})$$

$$+ P(\text{picking two letter from different columns})P(\text{those two letters are the same})$$

$$\approx \left( \frac{1}{l} \right) (0.065) + \left( 1 - \frac{1}{l} \right) (0.038)$$

$$= \frac{0.027}{l} + 0.038$$

Notice, long keywords will give you a value closer to 0.038, while short keywords will give you a value closer to 0.065.

Rearranging above, we get the **Friedman Test** for the length of the keyword, namely

$$l \approx \frac{0.027}{IC - 0.038}$$

**Example.** For our Vigenère example we have already worked out $IC \approx 0.049$, so using the Friedman Test we get

$$l \approx \frac{0.027}{0.049 - 0.038} \approx 2.45.$$  

From the Kasiski test we already thought $l$ might be a divisor of 50, namely 50, 25, 10, 5 or 2. So it seems most likely to be 2 or 5.

Try writing the ciphertext in rows of 5 letters. We may now apply decryption methods, such as frequency analysis, on each column.

Some guesswork can also help us find the keyword. For example, the first three letters of the ciphertext, ‘FHV’, is repeated three times. When the ciphertext is written in rows of 5, these three instances of ‘FHV’ appear in the same columns, meaning the letters have been enciphered in the same way.

Originally this may have been a common trigram such as the word ‘the’. We may use the formula $k_i \equiv c_i - p_i \mod 26$ to piece together the keyword.

Eventually we discover the keyword was **MARIO** and decrypt the ciphertext as below:

‘the Vigenère cipher is a polyalphabetic cipher named after the sixteenth century french diplomat blaise de Vigenère and it was eventually broken three hundred years later using the methods developed by kasiski and babbage’
2.4 One Time Pad Cipher

As we have seen, when using the Vigenère cipher, the fewer the repetitions and the longer the keyword the closer the frequencies are to a uniform distribution. The one time pad cipher uses a string of random letters, as long as the message itself, as its key.

Example.

Keyword: Q F D X S I J L C K S Y N
Plaintext: a t t a c k t h e t o w n
Ciphertext: Q Y W X V S C N O F G U A

If you are the interceptor with no knowledge of the key, then it can be proven that this cipher is unbreakable. We will not prove this here, however the idea is that given a message of length $n$ we could, by carefully choosing the key, decrypt the ciphertext to make any message we want. This message could be good, bad, or nonsense, and the key would be equally valid. So all plaintext messages of equal length are equally likely - and there is no way to tell what the original message may have been.

Here the strength of the cipher does not lie in the individual monoalphabetic ciphers (which are additive) but in the randomness of the key. The key must be genuinely random; algorithmic or computer generated random keys will not do. Also, to prevent repetition, each key must be used to encrypt one message only, then discarded.

In reality, the one time pad cipher is impractical to use. When choosing which cipher to use we must balance security with practicality and ease of use. However, the one time pad cipher is used today to send messages between the President of the USA and the President of Russia.
3 Enigma

By the beginning of the 20th century it had become possible, and necessary, to mechanise encryption. In 1918 a German engineer named Arthur Scherbius patented the Enigma Machine. Originally it was sold to banks, railway companies and other organisations who needed to send secret information. By the mid-1920s the German military started to use the Enigma Machine, with some differences from the commercial version of the machine. Enigma was used by the German military throughout World War II, therefore breaking the enigma cipher became a top priority, first by the Polish, then later by the British and Americans.

3.1 Enigma Encryption

The Enigma Machine was an electro-mechanical machine, about the size of a typewriter, made with steel casing inside a wooden box. On the outside you will see two sets of letters which were the keyboard and the lampboard. You would type your message using the keyboard, the message would then be encrypted letter-by-letter, but instead of printing on paper the encrypted letters would light up on lampboard. The encrypted message would then be written down by the operator and would be transmitted by radio. The machine itself did not transmit.

Enigma is a polyalphabetic cipher, where each individual monoalphabetic cipher turns the 26 letters of the alphabet into 13 pairs, also known as a product of transpositions. A letter is encrypted as its paired partner.

For example, here are the first ten letters of the two monoalphabetic ciphers needed to send the message ‘hi’.

<table>
<thead>
<tr>
<th>input:</th>
<th>a b c d e f g h i j</th>
</tr>
</thead>
<tbody>
<tr>
<td>output 1:</td>
<td>f i e h c a j d b g</td>
</tr>
<tr>
<td>output 2:</td>
<td>h g j f i d b a e c</td>
</tr>
</tbody>
</table>
Each cipher turns the 10 letters into 5 pairs, so if ‘a’ is encrypted as ‘f’ then ‘f’ is encrypted as ‘a’. And the message ‘hi’ becomes ‘de’.

A product of transpositions makes decryption easy. Notice, encrypting a message twice will return the original message. So, starting again from the beginning, we may encrypt the message ‘de’, and get ‘hi’ back. A product of transpositions is self inverse.

In other words, if $E$ is the Enigma cipher for a given position, then $E(h) = d$ and $E(d) = h = E(E(h))$. This is how enigma decryption worked; by encrypting the message for a second time using the same settings you would get the original message back.

The enigma settings changed every day. These daily settings were written down on a keysheet, which themselves were changed once a month. Next we will see how many settings the enigma had.

### 3.2 Combinations

We have already seen the idea of multiplying choices together, where $n!$ is the product of all integers from 1 to $n$.

For example, imagine we want to list all the ways to arrange the five objects A, B, C, D, E. Starting from scratch, we have five blank spaces to fill:

```
  _ _ _ _ 
```

For the first space we have 5 choices, A, B, C, D, E. Then, having picked a letter for the first space, we would have 4 choices left to fill the second space. After that, for the third space we would have 3 choices. Then 2 choices, then finally 1 choice for the last letter.

```
A B C D E
A B C E D
A B D C E
B A C D E
B A C E D
E D C B A
```

You can think of your choices as nodes in a tree diagram. Multiplying choices together we find there are $5! = 120$ ways to arrange A, B, C, D, E. And in general there are $n!$ ways to arrange $n$ objects.

**Definition.** *Permutations* are when you choose $r$ objects from $n$ and the order matters.
For example, imagine I want to choose 3 objects from 5. First, list all 5! ways to arrange A, B, C, D, E. Let the first three letters be the three objects I want, the last two letters are the objects I don’t care about. So,

\[ A \ B \ C \mid D \ E \]

The order of the first three objects matter, but the order of the last two objects does not matter, so

\[ A \ B \ C \mid D \ E = A \ B \ C \mid E \ D \]

There are 2! ways to arrange the last two objects, so we need to divide our list of 5! by 2!. The total number of ways to choose 3 objects from 5 is \( \frac{5!}{2!} = 60 \).

Note, this is still the same as multiplying our choices together. We have five choices in the first place, four choices for the second place and three choices in the third place, so \( 5 \times 4 \times 3 = 60 \) - and stop there.

In general, the number of permutations is given by

\[ n \ P \ r = \frac{n!}{(n-r)!} \]

**Definition.** *Combinations* are when you choose \( r \) objects from \( n \) and the order doesn’t matter.

So, in the example above

\[ A \ B \ C \mid D \ E = B \ A \ C \mid E \ D \]

In this case, we do not wish to distinguish between the different ways to arrange the first three objects. There are 3! ways to arrange the first three objects, so we further divide the list of \( \frac{5!}{2!} \) by 3! to get \( \frac{5!}{3!2!} = 10 \).

In general, the number of combinations is given by

\[ n \ C \ r = \frac{n!}{r!(n-r)!} \]

When you type in a letter on the Enigma Machine it completes a circuit and lights up a letter on the lampboard. The Enigma Machine has five components:

- Rotors 1, 2, 3: There are three rotors in the machine. Inside each rotor is a criss-cross of wires, which together form a composition of three general substitution ciphers. The army enigma machine had five rotors (labelled I, II, III, IV, V), from which you would choose three to put in the machine. As we have seen, there are 60 ways to choose 3 rotors from 5.

- The Reflector: The reflector made 13 pairs, and is the part responsible for the individual monoalphabetic ciphers being a product of transpositions. The reflector was not a choice, but fixed and unchanged.
The Plugboard: The plugboard is a military addition and is the main difference between this and the commercial Enigma Machine. The plugboard turned 20 letters into 10 pairs, leaving 6 letters unchanged. These pairs could be changed at any time. The number of ways to choose 20 from 26 is \( \frac{26!}{6!} \). We then form 10 pairs; the order of the pairs do not matter so we further divide this number by 10!. Finally, the order of the two letters within the pairs do not matter, i.e. AB is the same pair as BA, so we further divide by 2 for each pair. So, the total number of ways to connect 20 letters into 10 pairs is \( \frac{26!}{2^{10}10!6!} \approx 1.5 \times 10^{14} \).

Rotor 1 moved after every letter. When rotor 1 had done a full revolution it would kick rotor 2 one place. When rotor 2 had done a full revolution it would kick rotor 3 one place. So, there was a fast moving rotor, a middle rotor and a slow moving rotor.

You could rotate the rotor labeling and kickover point in relation to the internal wiring, this was called the ‘Ring Setting’. This does not increase the number of individual monoalphabetic ciphers but changes the pattern in which the machine moves between ciphers. Due to a quirk of the middle rotor called ‘double stepping’, the rotors actually had a period of \( 26 \times 25 \times 26 = 16900 \).

Finally, each individual message was encrypted with its own message key - this was the three rotor starting positions, which the operator was free to choose himself and send at the beginning of the message. Each rotor has 26 starting positions, giving \( 26 \times 26 \times 26 = 17576 \) message keys.

So altogether, the total number of monoalphabetic ciphers for the Enigma Machine was

\[
\frac{60 \times 26! \times 26^3}{2^{10}10!6!} \approx 1.59 \times 10^{20}
\]

As you can imagine, breaking the enigma code was quite a task! You may notice that this number is smaller than the number of general substitution ciphers we saw earlier. The number of combinations is only a measure of how difficult a cipher is to break by exhaustive key-checking (brute force!). However, what really made enigma so difficult to break were the moving rotors which creates the polyalphabetic cipher which protects it from frequency analysis.
3.3 Breaking Enigma

The Polish and British codebreakers eventually came up with a number of ways to determine enigma settings, these exploited various procedural mistakes and human errors made by the enigma operators. However, one such method was due to an inherent flaw in the design of the Enigma Machine itself.

Each monoalphabetic cipher consists of thirteen pairs, as a consequence no letter can be paired with itself. The cryptanalysts could then take some common phrase, slide it underneath the ciphertext and find where that phrase might fit in the ciphertext; two letters were not allowed to match. These phrases were known as cribs.

For example, there is only one place the phrase ‘wetterbericht’ (‘weather report’) may fit in the ciphertext below. Moving the crib to the left or right will result in two letters coinciding.

\[
\begin{align*}
j & \ x & \ a & \ t & q & b & g & g & y & w & c & r & y & b & g & d & t \\
w & e & t & t & e & r & b & e & r & i & c & h & t
\end{align*}
\]

Let’s assume this portion of the ciphertext does correspond to our crib phrase. Then we can draw a diagram describing the relations between letters, where two letters are joined by a labelled edge if those two letters form a pair at that position. This diagram is called a menu.

For example, above we have

\[
\begin{align*}
\text{position:} & \quad 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \ 13 \\
\text{ciphertext:} & \quad a \ t \ q \ b \ g \ g \ y \ w \ c \ r \ y \ b \ g \\
\text{crib:} & \quad w \ e \ t \ t \ e \ r \ b \ e \ r \ i \ c \ h \ t
\end{align*}
\]

From the crib we know that (te) is a pair in the second position, (tq) is a pair in the third position, (tb) is a pair in the fourth position, and (tg) is a pair in the thirteenth position. Continuing in this way we get the menu below;
We now continue by considering the Enigma Machine with and without its plugboard. The overall result in the $i$th position, $E_i$, is equal to the result of the plugboard, $P$; followed by the combined result of the three rotors and reflector in the $i$th position, $R_i$; followed by the result of the plugboard again; i.e.

$$E_i = PR_iP. $$

For example, from the crib we can see that, in position 2, ‘t’ becomes ‘e’. This is the combined effect of the plugboard, the rotors in position 2, followed by the plugboard again:

Remember, because the Enigma Machine acts as a product of transpositions, this diagram also works in the opposite direction.

Alan Turing realised that, for a given rotor setting, it is possible to deduce certain plugboard settings. For example, let’s assume (ta) is a pair on the plugboard. So after the first use of the plugboard, input ‘t’ becomes ‘a’.

This is then followed by the rotors in the second position. Starting from some initial position, below are the complete outputs of 13 successive monoalphabetic ciphers from an enigma machine without its plugboard, i.e. $R_i$, $1 \leq i \leq 13$. Notice, these ciphers are still products of transpositions.

<table>
<thead>
<tr>
<th>input:</th>
<th>a b c d e f g h i j k l m n o p q r s t u v w x y z</th>
</tr>
</thead>
<tbody>
<tr>
<td>output 1:</td>
<td>j f q x h b s e k a i y z t v u c w g n p o r d l m</td>
</tr>
<tr>
<td>output 2:</td>
<td>p n s k u z o w v l d j r b g a t m c q e i h y x f</td>
</tr>
<tr>
<td>output 3:</td>
<td>k d p b i q t m e o a n h l j c f t g r x z y u w v</td>
</tr>
<tr>
<td>output 4:</td>
<td>x o d c h z l e p y u g q w b i m v t s k r n a j f</td>
</tr>
<tr>
<td>output 5:</td>
<td>o f y e d b z x l w q i n m a s k v p u t r j h c g</td>
</tr>
<tr>
<td>output 6:</td>
<td>v w p t m x k u o l g j e z i r q y d k a b f s n</td>
</tr>
<tr>
<td>output 7:</td>
<td>d r t a g u e m z k j x i v y w s b q c f n p l o i</td>
</tr>
<tr>
<td>output 8:</td>
<td>v z e j c q u n l d y i r h w x f n t s g a o p k b</td>
</tr>
<tr>
<td>output 9:</td>
<td>h x i z g p e a c y o v s t k f w u m n r l q b j d</td>
</tr>
<tr>
<td>output 10:</td>
<td>h v x m z i k a f s g w d q u r n p j y o b l c t e</td>
</tr>
<tr>
<td>output 11:</td>
<td>o w m p y l t z k x i g c u a d r q v f n s b j e h</td>
</tr>
<tr>
<td>output 12:</td>
<td>r u z 1 l j y i t f e m d k x q r o a p h b w v n f c</td>
</tr>
<tr>
<td>output 13:</td>
<td>t l s p o h x f q k j b w r e d i n c a y z m g u v</td>
</tr>
</tbody>
</table>
This works under the assumption that the second and third rotors do not move. The probability that the second rotor will move for a crib of length \( n \) is \( n/26 \), so cribs were kept to 10-14 letters. From this table we see the input ‘a’ in the second position becomes ‘p’.

![Diagram of Enigma machine](image)

This leaves the second use of the plugboard. However, since we know the final output is ‘e’, we can deduce that (pe) is the other plugboard setting.

In the same way, by considering what happens in the third, fourth and thirteenth positions we get three further plugboard settings, namely (kq), (xb) and (tg).

But here we have a contradiction, (ta) and (tg) cannot both be pairs on the plugboard. So our original assumption that (ta) is paired on the plugboard is wrong. If we can show all possible pairs for a given letter like ‘t’ lead to a contradiction, then our rotor setting must be incorrect, and we should try another.

Turing’s brainwave was to realise that, due to the way the enigma works as a self-inverse product of transpositions, if we had assumed (tg) was a plugboard setting it would equally imply (ta), and so this assumption is equally false. The settings (pe), (kq) and (xb) may also be eliminated for the same reason. So, once we find a contradiction, we can eliminate all other plugboard settings derived from our assumption as ‘fruit of a poison tree’.

Turing used this principle at Bletchley Park in his ‘Bombe Machine’. The Bombe acted as simultaneous Enigma Machines connected in series, logical implications from one ‘enigma machine’ could then feed into the others, which were set at different positions with their input and output determined by the menu.

![Bombe Machine](image)

In the example above, the rotors of four of these ‘enigma machines’ would be set to position 2, 3, 4 and 13. An electrical current applied to the initial plugboard assumption of (ta) would result in a current passing through (pe), (kq), (xb) and (tg). This feedback was detected allowing these plugboard settings to be rejected. The chain of logical conclusions were deduced almost instantaneously by the electrical circuits.
After making an initial plugboard assumption, either;

- the current flows through all plugs for a given letter. The rotor position must be incorrect and we should try another;
- the current flows through one plug for a given letter, or all but one plug, and the machine stops;
- the current flows through some other number of plugs, meaning further checking is required;
- we get a false stop due to coincidence.

To reduce the problems of false stops and results that required further checking, the cryptographers used menus that contained cycles, this made inconclusive situations less likely and reduced the number of false stops. In our example above, using the menu we see a cycle connecting the letters t, e, and g. The correct rotor set-up will then make the following loop possible:

Applying successive Enigma functions we get the relation

\[ t = E_{13}E_5E_2(t). \]

Because the plugboard is a product of transpositions it is self-inverse. This means if we write \( E_i = PR_iP \) consecutive plugboards will cancel each other out, and this relation can be rewritten as

\[ P(t) = R_{13}R_5R_2(P(t)). \]

This calculation still requires an assumption about the plugboard, but it is a much tougher condition to satisfy and may be used to find contradictions as before. We may produce similar conditions from other cycles from the menu. The probability of satisfying \( k \) such cycles at random, causing a false stop, is \((1/26)^k\).

When the machine stops the rotor and plugboard settings are recorded. The remaining settings can be deduced by hand. Gordon Welchman later added a ‘diagonal board’ to the Bombe, increasing the machine’s connectivity and speed. Now the Bombe Machine could check the logical implications of all 17576 rotor positions in 20 minutes.

With all this in place, the bombe machines became a highly successful attack on the Enigma throughout the Second World War.
4 Public Key Cryptography

A cipher system is symmetric, or one-key, if it is easy to deduce the decryption key from the encryption key. In practice, for symmetric systems, these two keys are often identical. However, if it is practically impossible to deduce the decryption key from the encryption key, then the system is called asymmetric or public key.

A breakthrough was made in 1977 when Rivest, Shamir and Adleman devised such a system, now known as RSA. Let’s describe the general routine, with an example using the smallest possible numbers:

4.1 RSA

Choose two prime numbers $q_1$, $q_2$. Set $m = q_1 q_2$ and $A = (q_1 - 1)(q_2 - 1)$. Now choose a number $E$ which is less than and coprime to $A$. To encrypt the plaintext $p$ as ciphertext $c$ we use the formula

$$c \equiv p^E \mod m$$

The numbers $E$ and $m$ are published and form our public key to encipher the message.

For example, let $q_1 = 2$, $q_2 = 5$. Then $m = 2 \times 5 = 10$, and $A = 1 \times 4 = 4$, and let’s choose $E = 3$. The numbers we work with must be less than the value of $m$, so in this example I will send the message

message: b a d e g g
in numbers: 1 0 3 4 6 6

We proceed by raising the values of our letters to the power $E$ and finding the result modulo $m$,

plaintext: b a d e g g
in numbers: 1 0 3 4 6 6
raise to power $E$: 1 0 27 64 216 216
modulo $m$: 1 0 7 4 6 6

To decipher, a similar process is used, except that an integer $D$ is used in place of $E$ such that

$$p \equiv c^D \mod m. \quad \text{In other words, we need to find a number } D \text{ such that }$$

$$p \equiv c^D \equiv (p^E)^D \equiv p^{ED} \mod m.$$ 

**Theorem 4.1.1 (Fermat’s Little Theorem)** If $q$ is prime and $x$ an integer coprime to $q$, then $x^{q-1} \equiv 1 \mod q$.

**Proof.** If $x$ and $q$ are coprime then there exists integers $s$, $t$ such that $sx + tq = 1$. Equivalently, $sx \equiv 1 \mod q$. This means $x$ has multiplicative inverse $s$. So if $mx \equiv nx \mod q$ we may multiply both sides by $s$ and see $m \equiv n \mod q$. 

28
Therefore no two items in the list \(x, 2x, 3x, \ldots, (q - 1)x\) can be congruent modulo \(q\). This list is simply \(1, 2, 3, \ldots, q - 1\) modulo \(q\), although possibly in a different order. If both lists are the same, multiplying all terms in each list gives us:

\[
x^{q-1}(q-1)! \equiv (q-1)! \mod q.
\]

Since \(q\) is prime, it is coprime to all integers \(1, 2, 3, \ldots, q-1\), therefore we may cancel the term \((q-1)!\) on both sides as before. This leaves \(x^{q-1} \equiv 1 \mod q\) as required.

A corollary is an easy consequence of a theorem or proposition. Here we will be using the following corollary to Fermat’s Little Theorem.

**Corollary 4.1.2** Let \(q_1, q_2\) be primes, \(m = q_1q_2\) and \(A = (q_1 - 1)(q_2 - 1)\). Choose an integer \(E\) such that \(E\) is coprime to \(A\). Then there exists an integer \(D\) such that \(ED \equiv 1 \mod A\) and \(x^{ED} \equiv x \mod m\), for all integers \(x\).

**Proof.** First we'll show there exists \(D\) such that \(ED \equiv 1 \mod A\). If \(E\) and \(A\) are coprime then \(sE + tA = 1\), and we may simply use the integer \(D = s\).

To prove \(x^{ED} \equiv x \mod m\) we will instead prove an equivalent identity: If \(ED \equiv 1 \mod A\), then \(ED = kA + 1\) for some integer \(k\). And \(x^{ED} = x^{kA+1} = (x^A)^kx\). So we will prove that

\[
(x^A)^kx \equiv x \mod m.
\]

If \(q_1\) divides \(x\) then \((x^A)^kx \equiv x \equiv 0 \mod q_1\). If \(q_1\) does not divide \(x\) then Fermat’s Little Theorem tells us that

\[
x^A = (x^{q_1-1})^{q_2-1} \equiv 1^{q_2-1} \equiv 1 \mod q_1
\]

Or equivalently,

\[
(x^A)^kx \equiv 1^{k}x \equiv x \mod q_1
\]

Either way the congruence is true mod \(q_1\). Similarly it is true \(q_2\). Therefore, since \(q_1, q_2\) are both prime, the congruence must also be true mod \(m = q_1q_2\).

So to decrypt our message, all we need to do is find a number \(D\) such that \((E \times D) - 1\) is a multiple of \(A\). \(D\) is used to decipher the message and is not published. \(D\) is our private key.

Remember, such a number will always exist since, if \(E\) and \(A\) are coprime, \(sE + tA = 1\) and we may use \(D = s\). For example, if \(A = 4\) and \(E = 3\) as above, then \((7)(3) + (-5)(4) = 1\) and we may use \(D = 7\) as our decryption key.

| ciphertext: 1 0 7 4 6 6 | raise to power D: 1 0 823543 16384 279936 279936 | modulo m: 1 0 3 4 6 6 | plaintext: b a d e g g | 29 |
NB. I could have used $D = 3$ here, but did not want you to think the encrypt key $E$ and the decrypt key $D$ had to be the same.

Encrypting one letter at a time leaves you vulnerable to a simple frequency attack. This problem is overcome by grouping the letters into blocks.

For example, let $q_1 = 5$, $q_2 = 23$ (so $m = 115$), $E = 47$ and $D = 15$. Numerically, our message ‘bad egg’ has values ‘103466’. We may split this into three blocks of two letters ‘10 34 66’ and it is these groups that are acted on by the cipher.

We may use the properties of modular arithmetic to make these calculations more manageable. By observation notice that $34^2 = 1156 \equiv 6 \mod 115$. Furthermore, $6^{11} \equiv 1 \mod 115$. So a calculation like $34^{47} \mod 115$ becomes

$$34^{47} = (34^2)^{22} \times 34^2 \times 34 \equiv (6^{11})^2 \times 6 \times 34 \equiv 1^2 \times 6 \times 34 \equiv 204 \equiv 89 \mod 115.$$  

When we encipher the whole message we get:

- blocks: 10 34 66
- ciphertext: 80 89 111

We may now perform a similar operation on $89^{15} \mod 115$ to decipher the message

$$89^{15} = (89^3)^5 \equiv 19^5 \equiv 34 \mod 115.$$  

So the ciphertext becomes;

- ciphertext: 80 89 111
- plaintext: 10 34 66
- message: ba de gg

Since $q_1$ and $q_2$ are small here, $m$ may be quickly factorised, and the private key $D$ can be found from the public key. In reality the primes used are much larger, making factorisation a very difficult task. So, the size of the key is protecting you from brute-force attack, while blocks protects you from frequency analysis.

### 4.2 Signatures

In this system, it is intended that a message is put into cipher using $E$ and deciphered using $D$. But in fact, the value of $E$ and $D$ are interchangeable. That is, it is also possible to put a message into cipher with $D$ and decipher with $E$. This has a very important function in authenticating messages.

Consider two correspondents, John and Mary. Each has their own (different) $E$ and $m$ values which are public and known to everyone. Each has their own $D$ value which is known only to themselves.
Mary’s prime numbers were 127 and 191, so her public key is $E_M = 8023$, $m_M = 24257$. And Mary’s private key is $D_M = 18187$. John’s public key is $E_J = 8993$, $m_J = 11413$ (try to work out his prime numbers or private key!)

John is sending a message to Mary and it is important that Mary can know without any doubt that the message does indeed come from John. This (very much simplified) is one way it can be done.

John wants to send the message ‘meet you at six john’. Written in blocks this becomes:

me et yo ua ts ix jo hn

or, in numbers:

1204 0419 2414 2000 1918 0823 0914 0713

John attaches his signature, john or 0914 0713, again - but enciphers it using his (private) $D$ value (and $m$) so it now reads 2981 6024:

1204 0419 2414 2000 1918 0823 0914 0713 2981 6024

John now enciphers the message completely using Mary’s publicly known $E$ and $m$ values:

5141 12103 10542 3905 15868 21143 7264 15966 0124 15283

On receiving it, Mary uses her private $D$ (and $m$) values to produce:

1204 0419 2414 2000 1918 0823 0914 0713 2981 6024

She extracts the signature on the end and, knowing that the message is supposed to be from John, she uses his $E$ value on it to get:

1204 0419 2414 2000 1918 0823 0914 0713 0914 0713

or, in words;

me et yo ua ts ix jo hn  jo hn

and the only one who could have made that possible is John.