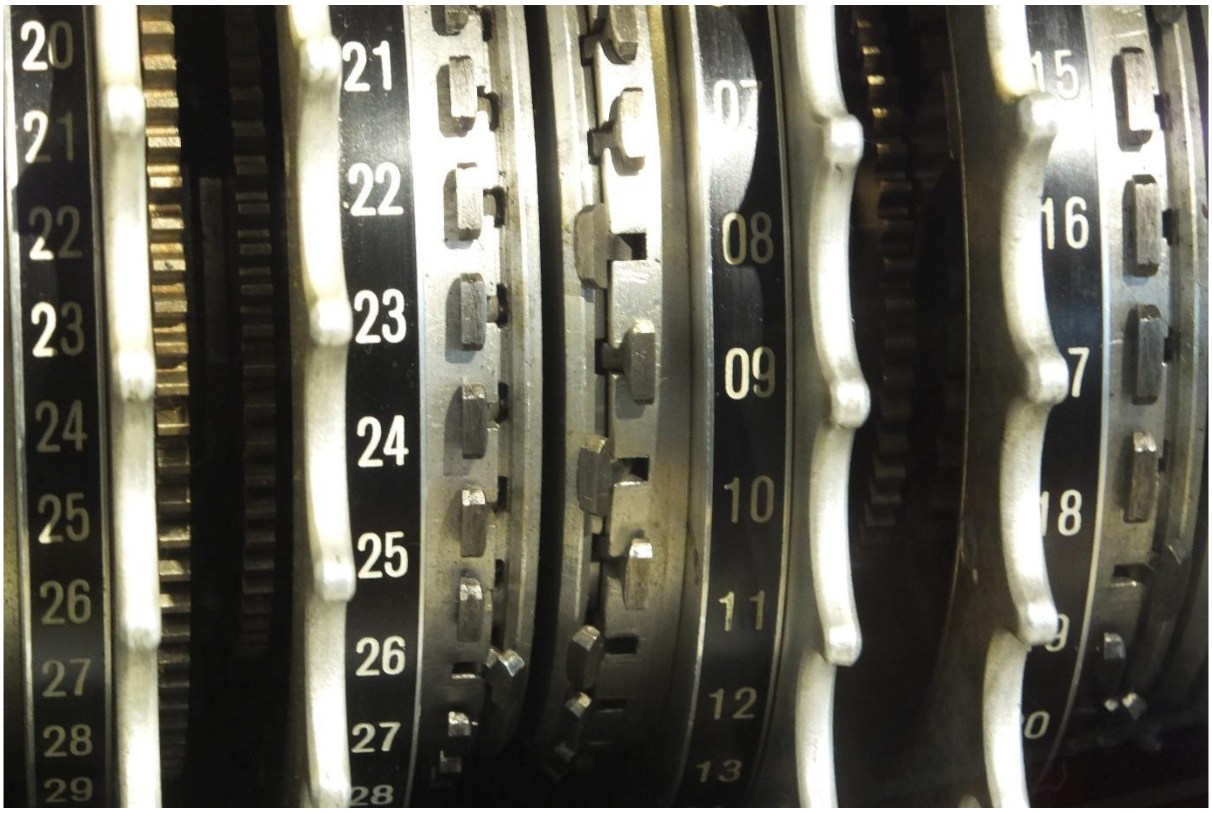
The Lorenz Cipher Machine

How it was broken at

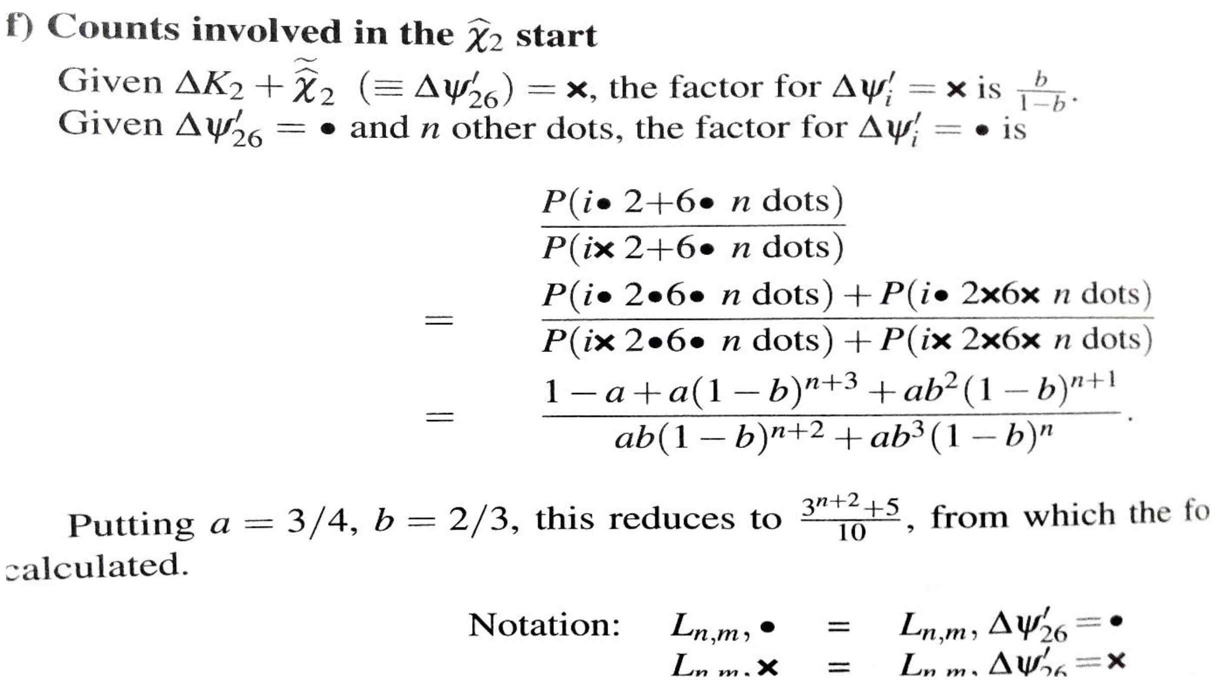
Bletchley Park



The writer of this paper was curious as to how the Lorenz machine was defeated so completely during World War 2. He searched for an explanation of how the Lorenz machine was *actually broken*, that he could understand.

The book *Breaking Teleprinter Codes at Bletchley Park* contains a great deal of information about the process, but it is intended for an academic audience and contains many statistical formulae that are beyond this writer's level of understanding.

Example:



f) Counts involved in the $\widehat{\chi}_2$ start

Given $\Delta K_2 + \tilde{\widehat{\chi}}_2 \ (\equiv \Delta\psi'_{26}) = \times$, the factor for $\Delta\psi'_i = \times$ is $\frac{b}{1-b}$.
Given $\Delta\psi'_{26} = \bullet$ and n other dots, the factor for $\Delta\psi'_i = \bullet$ is

$$\begin{aligned} & \frac{P(i\bullet\ 2{+}6\bullet\ n \text{ dots})}{P(i\times\ 2{+}6\bullet\ n \text{ dots})} \\ = & \frac{P(i\bullet\ 2\bullet 6\bullet\ n \text{ dots}) + P(i\bullet\ 2\times 6\times\ n \text{ dots})}{P(i\times\ 2\bullet 6\bullet\ n \text{ dots}) + P(i\times\ 2\times 6\times\ n \text{ dots})} \\ = & \frac{1-a+a(1-b)^{n+3}+ab^2(1-b)^{n+1}}{ab(1-b)^{n+2}+ab^3(1-b)^n}. \end{aligned}$$

Putting $a = 3/4$, $b = 2/3$, this reduces to $\frac{3^{n+2}+5}{10}$, from which the fo
calculated.

Notation: $L_{n,m},\bullet \quad = \quad L_{n,m},\ \Delta\psi'_{26}=\bullet$
$L_{n,m},\times \quad = \quad L_{n,m},\ \Delta\psi'_{26}=\times$

What follows is the writer's understanding of the process, gleaned from the Internet and the above book. Apologies if you find it either too simple or too complex. If you find any errors, please let us know at: www.tnmoc.org

It doesn't go as far as studying the analysis performed by Colossus, but it certainly covers the basic details. Nor does it cover the broad historical background of that part of Bletchley Park – it covers the technical aspects of breaking Lorenz from a non-statistical angle as far as that is possible. We assume you are reading this because you know about the Lorenz machine and want to know more. If you don't know how the machine works then this paper will help you understand it.

The cryptographers at Bletchley Park named the strange radio cipher stream that they were struggling to understand: ***Tunny***. Fish names were used for the ever-increasing network of radio links that were being set up by Germany across Europe and beyond. For example, Bream, Jellyfish and many others.

When the war was over three of the top codebreakers, Jack Good, Donald Michie and Geoffrey Timms wrote *General Report on Tunny, with Emphasis on Statistical Methods*. This was classified by the UK government until June 2000. It forms the basis for most of the current information about the activities of that section of Bletchley Park during the war. It is a long, detailed and fascinating account of every aspect of codebreaking the Lorenz cipher during World War 2. It has been expanded upon and commented by experts in cryptography in an excellent book called: *Breaking Teleprinter Ciphers at Bletchley Park*. See *References* at the end of this paper.

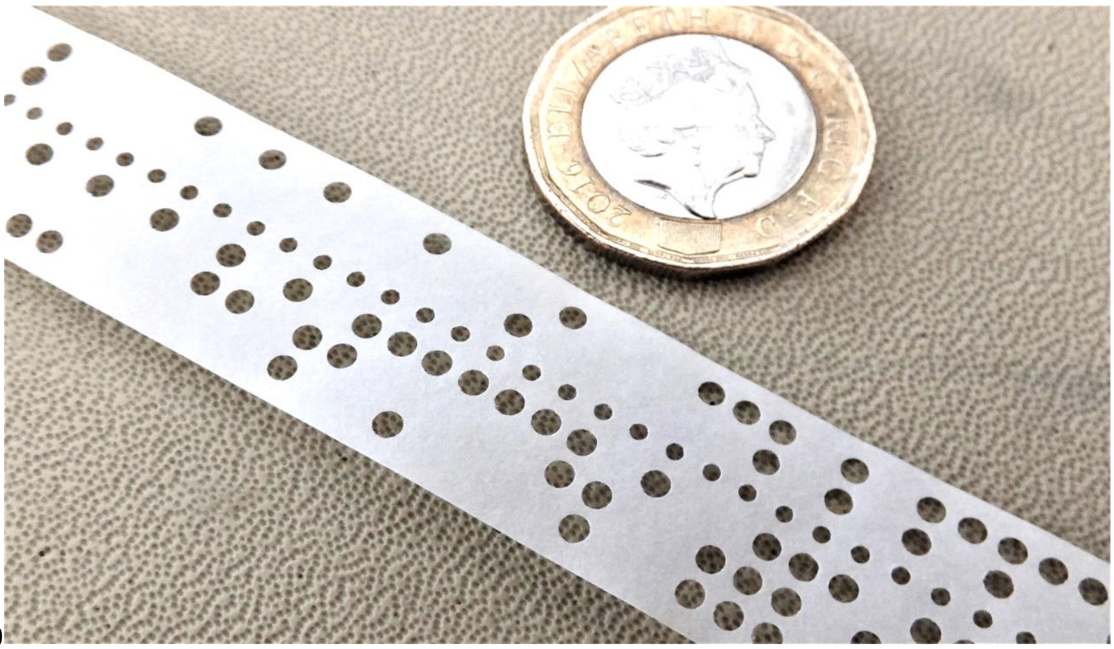
Basic Operation of Lorenz machine

Teleprinters were used in many countries around the world from the 1920's. Messages were sent via either land lines or radio. During World War 2 the Germans used mainly radio for their links, this allowed portability. Lorenz, by the way, has nothing to do with Enigma, nothing.

Each letter of a message contained 5 *data bits*. These bits could be any combination of the two binary states 1 or 0 (a hole in the tape or no hole). If we calculate how many possibilities there are from 5 bits, we get 32. This is not enough for a 26-letter alphabet AND ten digits AND punctuation. So two of the 32 letters are special: *letter shift* and *figure shift*. This almost doubles the number of symbols that can be generated and allows roughly 60 letters to be represented.

The code used was the International Telegraph Alphabet No2 which was pretty standard around the world at that time. Although the radio signal was identified as teleprinter traffic, attempts at making any sense of the transmissions resulted in failure. It looked like a random stream of letters with no bias towards any particular letter (as in *letter frequency analysis* where common letters, like E and T in English, are more common than say, Q or Z. Look up *Caesar Shift* for a method of cryptography which was broken using letter frequency analysis).

Bletchley Park used 5 channel paper tape (shown below, which was common in the Teleprinter world) to record and process the messages. The UK one pound coin shows the scale of the tape:



0

An example of some of the ITA2 coding is shown below (O = hole in tape)

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**	sp	**K**
	O	O	O		O	O	O				O							O		O		O		O	O	O	O		1
	O	O		O				O		O	O	O	O				O	O	O			O	O	O					2
Tape	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
	O			O			O		O	O		O		O	O		O	O		O		O	O		O	O		O	3
	O		O	O	O		O	O			O	O		O	O	O			O				O		O				4
	O		O					O	O				O	O		O	O	O			O		O	O	O	O	O		5

The Lorenz machine uses 12 code wheels, each wheel has a number of cams (switches – 501 in total) around its periphery. The first 5 wheels (K1 to 5) are arranged so that one bit of each incoming letter has its own wheel (bit 1=K1, bit 2=K2 and so on).

These wheels can change the value of their data bit depending upon the way their cams are set. Cam set = change the data bit: a 0 becomes a 1, and a 1 becomes a 0. Cam not set = no change to the data bit: 0 stays at 0 and 1 stays at 1. These changes are always made using modulo 2 arithmetic. Modulo 2 arithmetic crops up in nearly all aspects of cryptography, otherwise known as Exclusive OR or XOR for short. It is a way of apparently scrambling data and then unscrambling it again later. There follows a demonstration of Modulo 2 arithmetic. "+" is often used to mean XOR

0+0=0
1+0=1
0+1=1
1+1=0

A simple way to remember these rules is: *"If they're the same, the answer is 0"*
"If they're different, it's a 1"

Take the letter A for example. In ITA2 teleprinter code it is 11000 (in binary). "0" means no hole in tape, "1" means hole.

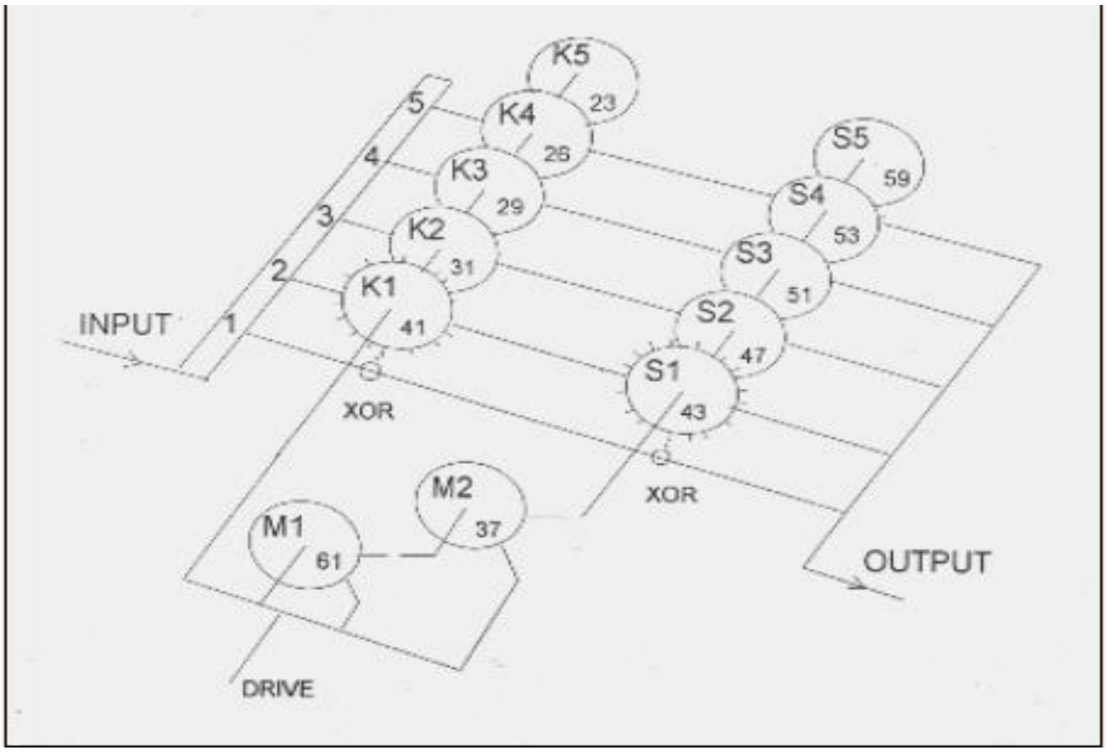
Now XOR it with part of the cipher key, say 10001. (This is what the K wheels do)

```
     11000  letter A in data
 XOR 10001  cipher (any value for simplicity)
 =   01001  result which is letter L, this is cipher text
```

Now to reverse the process we XOR L with the cipher key 10001

```
     01001  letter L in cipher stream
 XOR 10001  same cipher code as before
  =  11000  we get the original letter A back
```

Lorenz Machine Logical Layout



The input letter (from keyboard or paper tape reader) is first processed by the K wheels. The current position of say, K1 will have a cam which will be either set or not (K1 has 41 cams). If the cam is set, the input bit is XOR'd with the cam position. (If the cam is not set then the bit is unchanged, otherwise it is inverted: 0 becomes 1 or 0 becomes 1. This happens on all five K wheels in parallel.

The modified output from the K wheels then reaches the S wheels where the bits are again changed/not changed depending on the S wheels cam settings. In the unlikely case of ALL the cams on all the wheels being off (not effective), the output would be an unchanged copy of the input.

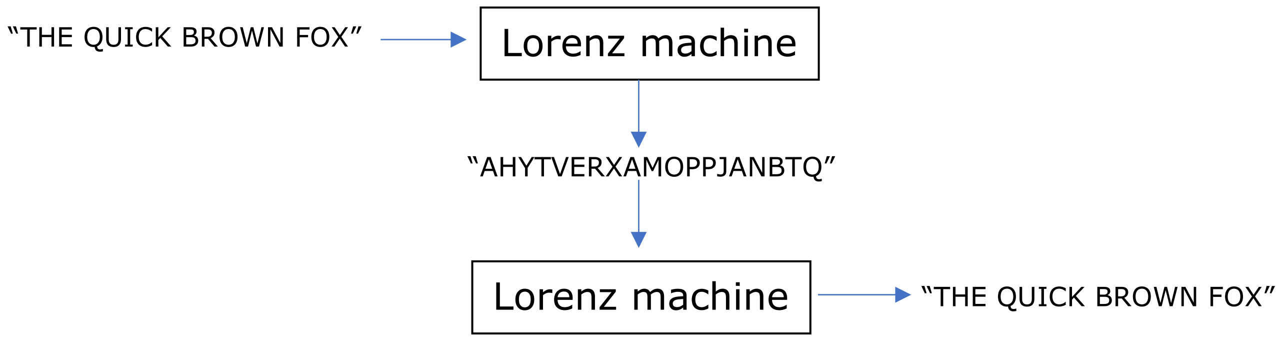
The K wheels *move on one step* for every letter received but the S wheels *only move if commanded to do so by the Motor wheels*, M1 and M2. The motor wheels have their cams set such that they will move roughly 50% of the time, and stay still roughly 50% of the time. ("Time" meaning in step with the letters coming in.) The two motor wheels were intended to make the cipher more unpredictable and difficult to break. The Germans believed that Lorenz was unbreakable. They were wrong and cryptography took several important steps forward at Bletchley Park.

The output from the S wheels is either the final *enciphered* or the *deciphered* output. This is because a Lorenz machine will encrypt an input stream or it will decrypt an input stream. The machine just applies the settings on the wheels to the input, it doesn't know if its input is cipher or plain text, and it doesn't care, it just goes through the motions. Follow the progress of say, the letter "A":

K/S Wheel number	Letter "A" →		K wheels →		Result →		S wheels →	Output (letter "Z")
1	1	+	1	=	0	+	1	**1**
2	1	+	0	=	1	+	1	**0**
3	0	+	1	=	1	+	1	**0**
4	0	+	0	=	0	+	0	**0**
5	0	+	1	=	1	+	0	**1**

If the S wheels *don't* move (as a result of the motor wheels not commanding them to do so) then the S wheels contribute the same 5 bits that they did the time before. The designers of Lorenz thought that this would make the cipher more secure; in fact it simplified the job of the analysts, as we shall see later. (Some later studies of this process have suggested that the machine would have been much more secure if the S wheels always moved on, and the the motor wheels were dispensed with.)

If two Lorenz machines are connected together, and the wheels *are set identically on both machines*, then plain text entered on one machine will be enciphered, transmitted to the other machine, where it will be deciphered. If "THE QUICK BROWN FOX" is entered to one machine, "THE QUICK BROWN FOX" will be output by the other. But, the letters passing from the first machine to the second will be gobbledegook, this is the *Cipher Text*. It was this Cipher Text that was received by radio and then sent to Bletchley Park for analysis. It is the deciphering of that data that is the subject of this paper.



But only if BOTH machines are set exactly the same and the 12 wheels start from the same starting positions

Complexity of the key

The key is determined by the settings of the cams on the 12 wheels. These wheels have the following number of cams on each:

K	41, K2 31, K3 29, K4 26, K5 23
S	43, S2 47, S3 51, S4 53, S5 59
M1	61
M2	37

These numbers a *relatively prime* which means that if you went through the whole sequence of possible numbers, there wouldn't be any repeats until you had stepped:

41x31x29x26x23 = 22,041,682 steps for the K wheels

43x47x51x53x59 = 322,303,017 .. .. .. S ..

61x37 = 2,257 steps for the motor wheels

Another way of looking at the complexity of the key is that there are a total of 501 cams on the wheels. This theoretically allows 2 to the power of 501 settings, which is an exceedingly big number – more than the number of atoms in the universe.

Which means that just guessing at the settings is not going to be feasible. Don't forget that computers as we know them didn't exist then.

The first version of the Lorenz machine was the SZ-40. Later versions added hardware changes that made breaking even more difficult. These are called *limitations* but this paper only covers the basic machine (which is difficult enough). We will mention them towards the end but only in passing.

Initially the wheel settings were only changed once a month. Once the wheels had been broken, only the starting positions had to be found and the traffic could be read until the end of the month. But as time went by the settings were changed more often, eventually every day. This pushed the cryptographers very hard.

Key Distribution

The settings and start positions of the 12 wheels were distributed to the various German military units in the form of a *Code Book*. Setting a Lorenz machine from scratch was quite a long job, and it had to be done accurately. Lorenz key information was of the highest grade and security was paramount. Several successful attempts at getting the codebooks for *Enigma* were recorded but there were almost no cases of the Lorenz codebooks being compromised by Bletchley Park.

Because the process of setting and changing the wheel patterns was lengthy, changes were initially only made once a month. As the war progressed changes were made more frequently until changes were made every day towards the end.

Below is a sheet for the settings of an actual cipher ("0"=cam not set, "+"=cam set, will change cipher, number in brackets is number of cams on wheel), last number is number of +'s:

```
S1 0+0+0 0+0++ 0+0+0 +0+00 ++000 ++++0 0+0+0 +0++0 ++0                  [43] 22
S2 00+0+ 0++0+ 0+++0 0++0+ 0+00+ +0000 +0+0+ ++00+ 0++0+ 0+             [47] 24
S3 0+00+ ++000 +0+00 0++00 +0+0+ +0+0+ ++00+ 0+00+ 0+0++ +0+0+ +        [51] 26
S4 +0++0 0++0+ 00+0+ 0+00+ +0000 +++00 +0+++ +0+0+ 0+0+0 +00+0 ++0      [53] 27
S5 +0+00 +0+0+ 0+00+ 000++ 000+0 +++0+ +0+0+ 0+00+ ++00+ 0+++0 0+0++ 0+0 [59] 29

M1 +0+00 00+++ 00+++ +0+++ +00++ ++0++ ++0++ ++000 ++++0 +++0+ ++00+ +++0+ + [61] 41
M2 ++000 +0++0 ++++0 +++00 ++000 +0+0+ 000++ +0                          [37] 20

K1 0++00 ++00+ +00++ 00++0 0++++ 00++0 0+00+ +00++ 0                     [41] 21
K2 000+0 +00+0 ++000 0+++0 0+0++ ++0++ 0                                 [31] 15
```

```
K3 +++00 +++00 +++00 +0000 +++0+ 00+0                                        [29] 14
K4 00+0+ +++00 +00++ 0000+ +++00 +                                           [26] 13
K5 ++000 +++0+ 00+++ 0+000 ++0                                               [23] 12
```

Note that the number of cams set (+) is always as near as possible 50% of the total cams on that wheel. Except the motor wheels, they are different. This is to try to ensure that the resulting cipher stream contains 50% 1's and 50% 0's and looks like a random stream of information, to deter cryptanalysis. The rules for the movement of the motor wheels are such that the S wheels move, on average, 50% the time. (And therefore stand still 50% the time.)

The first few sequences of 5 bit codes generated by the K wheels above, starting at their "home" position would be:

```
00101 10101 10110 01000 00010   11111 10111 00111 01000 10001 . . . . . . .
```

The distribution of the keys is also sometimes called *The Key Distribution Problem* and it was always a problem faced by any cryptographic system which needs the same key at both ends of a link. Today we have *Public Key Cryptography* which removes this problem. (Google: *Public Key Cryptography*)

Depths
From June 1941 mistakes by the German operators resulted in a few short messages being received which used the same key - depths. But these were few. They gave the cryptographers a clue about the cipher being used but nothing else. As you will see in the next section, by adding two messages together with exclusive OR, the key is cancelled out. But what remains is a single stream of letters, each of which is the addition (mod 2) of two original letters. There is no way of separating them without further information.

First Significant Break
The first significant break of Tunny was on 30th August 1941 when a German operator (effectively) sent two different messages on the same machine setting (the same key). Both messages started with the twelve letters: HQIBPEXEZMUG. It turned out that these referenced the starting positions of the 12 wheels, but nobody knew that at the time. That message became known as simply ZMUG.

This is known as a *Depth*. Let's call the two messages Plain1 (P1) and Plain2 (P2). The key is the same for both messages, call this K.

The first cipher text (Z1) consists of P1 XORed with K.

The second cipher text (Z2) consists of P2 XORed with K.

If the plain text messages had been identical then the cryptanalysts would be none the wiser, but they were different although some stretches were the same, but shifted due to abbreviations being used. This was a major blunder by the machine operator.

We will use "+" instead of "XOR". Note that XOR is both addition and subtraction in one. It forms the *difference* between two values.

Now we know from modulo two arithmetic that if we XOR Z1 with Z2 the key will be eliminated. It will "cancel out". K + K = 0

This is because P1 + K + P2 + K = P1 + P2 (the same thing @ with itself = null).

You can see that if the two texts had been the same then: P1 + P2 = 0, which means "there is no difference between these two texts".

So now we have the two texts, deciphered, but mixed together. A beautiful gift from the Germans. But how to unravel them?

Fortunately, P1 and P2 both started the same but then differed after a few letters.

The first text started (after the 12 letter identification) "SPRUCHNUMMER" and the second text "SPRUCHNR". This gave John Tiltman (a cryptanalyst) a starting point. Being very good at crosswords and knowing German he teased apart the following letters. Remember each letter was the XOR of two letters and there were many possible combinations and possibilities. As he worked along the text clues were revealed from the shorter text as to what the next letter could be in the longer text.

For example, you could make a letter D by adding J and E or N and S or M and Y and so on. In fact, there are 28 other additions that will result in D, hence this could be a long job. If one letter made sense in P1 but its partner not in P2, then it was wrong and another must be tried. This process took Tiltman ten days. Bear this "suck it and see" process in mind when we examine *Turingery*.

Interestingly if you XOR the new (correct) plaintext to one of the original cipher texts, you get the key! This is because if A+B=C then B+C=A and A+C=B. In other words, if you know two variables, you can usually find the third.

It was Bill Tutte who worked out the structure of the machine from this first depth. He drew out the pattern of the obtained key on a grid of squared paper. He was looking for possible repeat patterns in the structure of the data. After many failed attempts he spotted that the number 41 on his grid (say 41 wide) produced a noticeable pattern. He went on to deduce that the machine had 12 wheels and the length of each wheel, including how the two motor wheels operated. This was said to be the greatest intellectual achievement of the second World War.

Although depths sometimes allow the current key to be obtained, they don't tell you anything about the settings of the cams on the twelve wheels. The plain texts *can* sometimes be teased out, but this is a long and tricky process.

Depth Example
Let's make up a depth to demonstrate the problem of separating the two possible texts:

Plain text 1: HELLO FATHER
Plain text 2: HELLO MOTHER

Our key will be: 10101 10101 10101 10101 10101 10101 10101 10101 10101 10101 10101 10101

Not a very good choice but we are trying to understand the principle, not generate a secure system. Also, we won't use all 12 wheels. Just a straight XOR with each letter of the plain text to keep it simple, the principle won't be changed:

```
Plain1:      H     E     L     L     O   space   F     A     T     H     E     R
   ..      00101 10000 01001 01001 00011 00100 10110 11000 00001 00101 10000 01010
Key:       10101 10101 10101 10101 10101 10101 10101 10101 10101 10101 10101 10101
Cipher1:   10000 00101 11100 11100 10110 10001 00011 01101 10100 10000 00101 11111

Plain2:      H     E     L     L     O   space   M     O     T     H     E     R
   ..      00101 10000 01001 01001 00011 00100 00111 11101 00001 00101 10000 01010
Key:       10101 10101 10101 10101 10101 10101 10101 10101 10101 10101 10101 10101
Cipher2:   10000 00101 11100 11100 10110 10001 10010 01000 10100 10000 00101 11111
```

So here we have a *depth*, two messages sent on the same key. We determined earlier that if you add the two cipher texts together modulo 2, *as long as the key was the same*, then the key will "drop out", be cancelled leaving just the two messages merged together. Adding the two cipher texts (XOR):

```
Cipher1:   10000 00101 11100 11100 10110 10001 00011 01101 10100 10000 00101 11111
Cipher2:   10000 00101 11100 11100 10110 10001 10010 01000 10100 10000 00101 11111
Result:    00000 00000 00000 00000 00000 00000 10001 00101 00000 00000 00000 00000
```

From this we can see that the two plain texts were the same for the first 6 letters (HELLOsp) , then they differ for two letters, then they are the same again. Without further help it would be impossible to work out what either message contained. However, if we somehow did manage to work out what Plain1 said, by adding it (XOR) to the cipher text, an interesting thing happens:

```
Plain1:     00101 10000 01001 01001 00011 00100 10110 11000 00001 00101 10000 01010
Cipher1:    10000 00101 11100 11100 10110 10001 00011 01101 10100 10000 00101 11111
Result:     10101 10101 10101 10101 10101 10101 10101 10101 10101 10101 10101 10101
```

We have recovered the key! This is the powerful result of recognising a depth and then working on it.

But there must be something else to seed the untangling of the two texts. This happened in the ZMUG depth because the two messages started the same but then the second one started using abbreviations thus shortening the second text. Is was only because the first part of the message had a stereotypical beginning: SPRUCHNUMMER in the first message and SPRUCHNR in the second (meaning MESSAGE NUMBER) that the cryptographers guessed at a few letters following and teased apart the two texts. Quite an achievement.

This technique of prising apart two texts from a depth was known as *anagramming*. It also happened that the same text was sent on two (sometimes more) keys. If one of the keys was already known, the other could be recovered by a similar method as shown above.

In July 1942 Alan Turing was interested in breaking Tunny and developed a method which became known as *Turingery*. It relied on having at least 500 letters of a partial depth, some stretch of cipher text that had been repeated and some of the key had been revealed.

The Germans had arranged that there would be, as nearly as possible, an equal number of 1's and 0's in the cipher stream. It thus appeared as a random stream of teleprinter letters with no bias for any particular letter. But, when the ***delta*** of this cipher text was calculated, which means that each letter was XORed with the next letter, then that letter was XORed with the next letter and so on, it revealed ***changes*** in the cipher stream. And this exposed a non-randomness which was meaningful.

Take the output from one wheel for example K1 (chi1):

```
Cipher stream from K1:  0   1   0   0   1   1   1   0   0   0
                         \  /\  /\  /\  /\  /\  /\  /\  /\  /
Delta of K1:              1   1   0   1   0   0   1   0   0     ("1" = change, "0" = no change)
```

The cipher text is made up of three components:

> The plain text, the changes imposed by the K wheels and the changes imposed by the S wheels. These three parts are all XORed together in the Lorenz machine to produce the output cipher text.

The *delta* of the data stream reveals something about the randomness of the cipher text. The German operators of the teleprinters connected to the Lorenz machines repeated certain letters. To go from "letters" to "figures" needed a press of the FIG SHIFT key. This generated the code to interpret following key-presses as figures, the digits 0-9, full stop etc. But the radio links were not entirely reliable and if such a shift character was missed then the following characters would be in the wrong shift and cause mayhem. So the operators always pressed FIG and LETTER shift twice (sometimes three times). Other important characters were also repeated by the operators. This proved to be a big mistake.

The K wheels (chi) move every time but the S wheels (psi) only move about half the time. Turing's insight was as follows: whenever the S wheels ***don't*** move, and the input letter was a repeated character (like a shift), then we are adding the same thing twice. If you XOR anything with the same thing the result is zero, or null and it will have no effect and won't change the input character. In other words, when this happens, the cipher data at that point is the actual K wheel setting along with the plain text bit! As we know the key (from the partial depth) we can determine the actual value of the K wheel bit at that position. This process became known as de-chi-ing.

Although this sounds very encouraging there is a snag. How do we know when the S wheels are moving or not? We don't. But we can guess. And we will be right about 50% of the time.

Consider the actions of K1 and S1. K1 has 41 cams around it, S1 has 43. If we look at the cipher stream in the position that K1 works on, we know that every 41 letters K1 will have rotated all the way round and will be on its next revolution. Which means that the effect of K1 will repeat every 41 letters. S1 will have another effect, and this won't be of the period 43 because S1 doesn't always move. But the pattern of S1 *will* repeat after it has moved 43 times. Given at least 500 letters of recovered key, Turing's method could sometimes give the complete wheel patterns.

His method is as follows: the delta of a stretch of recovered key is written on squared paper in a grid, say 41 squares wide by 43 deep. It then takes a letter from the delta'ed cipher stream and supposes that it is one of the points when the S wheels *didn't* move. The corresponding bit is then written on the grid in pencil as 0 or 1. The same point in the cipher stream 41 places on, or back is then examined and plotted. Half the time these guesses will be wrong, and this will lead to clashes where a wheel position is both 0 and 1, clearly wrong. It is this that makes the process long and tedious with much pencilling in and rubbing out. But it is a technique that was immensely insightful and valuable.

We are also assisted by the design of the Lorenz machine. The rules about setting the tabs on each wheel allow a maximum of four 0's or four 1's in consecutive positions with as far as possible, an equal number of 1's and 0's. So, we know that if we have just found four consecutive 1's, then a 0 must follow.

And this is where the writers understanding begins to thin out. This process is continued, building up a table of might-be/might-not-be's for each position of (say) K1 until a feasible 41 steps have been set with no (or few) contradictions. The other K wheels are similarly attacked.

Although the output cipher stream was effectively a random stream of data, once the wheels had been discovered (broken) a great deal of non-randomness is revealed. The German teleprinter operators often pressed letter shift/figure shift twice to be sure that the shift would actually *take*. As in English, double letters occur in German and a very uneven distribution of letter frequencies helped to untangle the messages.

Turingery was slow and involved a lot of backtracking, but it usually worked. And it paved the way to ***Rectangling*** which (I believe) is based on the same basic theory with a huge amount of additional statistical working. Although the design of the Lorenz machine tried to achieve an equal number of 0's and 1's in the cipher stream, techniques were developed which looked for small but statistical bulges in the number of 1's or 0's. And of course, wheel breaking without depths followed, with Robinson, Colossus and several other machines.

Cribs
One technique often used was to guess part of a message. Although the designers of Lorenz told their users never to use stereotypical beginnings for their messages, they were often ignored. A message might start with the army unit it was from, the date and other things that could be guessed. When this happened the cipher text was "slid" along a combination of the guessed words and the suspected key. On Hitler's birthday, more than one message ended with the two words "Heil Hitler"; this was also successfully guessed by the cryptographers. As the war progressed the Germans started putting a handful of random letters at the start of their messages to combat this method of attack. We called this added material *quatsch*.

Wheel Setting
Once the key has been determined, by whatever method, the wheels must now be set. This means putting them into the "correct" starting position. There are a very large number of combinations of the 12 wheels starting positions:

$41 \times 31 \times 29 \times 26 \times 23$ x $43 \times 47 \times 51 \times 53 \times 59$ x $37 \times 61 = 1.603 \times 10^{19}$ i.e.16 billion billion
------------K----------   ------------S-----------   Motor

There are so many possible combinations that "try it and see" is impractical.

It was discovered that if you took the *delta* of the key and compared it to the *delta* of the cipher text in a sliding motion, at certain points there was a slight statistical bump. In other words, if you could slide the key against the cipher text, one letter at a time, at the "correct" place there would be a correlation of the two. And then if you XORed the two together, aligned at that point, the correct plain text might be revealed.

Bill Tutte found that it was possible to reduce the size of the search by only looking at two wheels instead of five. If only K1 and K2 were used then only 1271 comparisons were needed instead of millions yet the statistical bump was still present.

The K wheels and the S wheels had to be set separately, also true for the motor wheels.

Max Newman suggested building a machine that could do this – run two paper tapes along-side each other. Electronic circuits would do the *delta* of the data and electronic counters would enable statistical bumps to be spotted by the operator. Photo-electric cells would read the holes, the first time this had been done. Electronic circuits (using valves) would process the data. After each revolution one tape would step on one place so that eventually all the letters punched on one tape would be compared to all the letters on the other tape.

A machine was built by a collaboration between the Post Office and the government Research Station at Hanslope Park. The machine was named Heath Robinson by its operators because it resembled the hilarious contraptions drawn by the cartoonist of the same name.

It worked, after a fashion. Robinson was not too reliable but it enabled the start positions of the wheels to be set and became one of the first electronic machines involved in mathematical data processing. Several Robinson machines existed at the end of the war. A Post Office engineer called Tommy Flowers had been involved in the design of Robinson and he realised that there was a much better way to compare data streams. He designed the machine we know as Colossus.

It is the writer's belief that the whole of the later process of breaking Tunny, using Colossus, went back to Turingery and Turing's brilliant insight that the *delta* of the cipher stream contained vital information which could lead to possible decryption.

Lorenz Simulator
The engineers at The National Museum of Computing (Tunny gallery) have built a simulator which demonstrates the action of the first five wheels, the K wheels. You can enter letters from a keyboard and see the process of encryption step by step. An animated video accompanies the simulator.

We also have a true Lorenz machine which has kindly been lent to us by the Norwegian army. You can see it on display at the museum. www.tnmoc.org

----

Machines
The following machines were designed and built during the war:

Robinson –	For comparing a cipher message paper tape with a possible key tape
Miles, Mrs Miles –	Read one or more paper tapes and performed certain actions on the data, then punched an output tape
Dragon –	"Dragged" cribs through the K wheels looking for ***cribs***
Proteus –	Consulted a dictionary looking for ***cribs***
Aquarius –	Looked for "go backs" where the operator had experienced problems and pulled the plain text tape back a number of places
Tunny –	Simulated the Lorenz machine
Decoding –	Produced printed output once wheels had been broken and set the last stage of deciphering a message
Colossus –	The most complex, multi-purpose, wheel breaking and setting

Terms

A brief description is given here of the terms used and explained in this paper:

Cam - The mechanical cams mounted around each wheel. They could be moved (by the operator setting up the machine). There were a total of 501 cams. See photo on front cover, the cams are visible

Cipher Text - The encrypted version of the message, once it has "been through" the Lorenz machine. The output of the Lorenz machine

Cribs – A method of guessing what part (usually the beginning) of a message might say

Depth - Two messages (or part messages) sent on the same key. This is strictly forbidden as it gives away part (or all) of the cipher key

Key - The cryptographic coding that is used to encipher the message. In the SZ- 40 series of Lorenz machines this was 501 bits long

Modulo Two Arithmetic - The addition of binary values which is also known as Exclusive OR or XOR:

Exclusive OR or XOR:	0 @ 0 = 0
	0 @ 1 = 1
	1 @ 0 = 1
	1 @ 1 = 0

Plain Text - The message before encipherment

Relatively prime – there are no common factors between these numbers

Teleprinter – A keyboard-typewriter machine that could transmit/receive text over either land telegraph lines (telephone lines) or radio. The teleprinters used by the Germans printed on a continuous length of tape. This was torn at suitable places and stuck down to form the message. Most of the links were (fortunately) radio. We wouldn't have been able to intercept land lines

Turingery - Alan Turing found a way of discovering the wheel settings by a pencil and paper method. His colleagues named it Turingery

Wheels - The twelve "wheels" that hold the mechanical switches on which are set the *key* being used. The wheels are named 5x"S" (or psi) , 2x"Mu" (or Motor), 5x"K" (or chi)

Wheel Breaking - The process of discovering the settings of the cams on the twelve wheels

Wheel Setting - Discovering at which position each wheel sits at the start of the message

XOR – See *Modulo Two Arithmetic* above

Limitations
These were modifications to the basic machine, designed to make breaking more difficult:

SZ-40	1940	Basic machine as described in this paper
SZ-40A	1942	The *previous* letter of wheel K2 modified cipher
SZ-40B	1942	The fifth bit of the plain language input two letters back is added to the second position of K2. This rendered depths impossible to decipher because the cipher stream was now modified by the text being input
SZ-40C		Never used, war ended before it was implemented

Tribute from America
Albert W. Small, a cryptanalyst from the US Army Signal Corps who was seconded to Bletchley Park and worked on Tunny, said in his December 1944 report back to Arlington Hall that:

Daily solutions of Fish messages at GC&CS reflect a background of British mathematical genius, superb engineering ability, and solid common sense. Each of these has been a necessary factor. Each could have been overemphasised or underemphasised to the detriment of the solutions; a remarkable fact is that the fusion of the elements has been apparently in perfect proportion. The result is an outstanding contribution to cryptanalytic science.

References: Breaking Teleprinter Ciphers at Bletchley Park.
Wiley ISBN 978-0-470-46589-9.

This book is the holy grail of the events at Bletchley Park during the war as far as the Lorenz machine is concerned. It is based on the original report (General Report on Tunny, with Emphasis on Statistical Methods), written after the war by some of the leading cryptologists.

Charles Coultas
The National Museum of Computing - March 2020