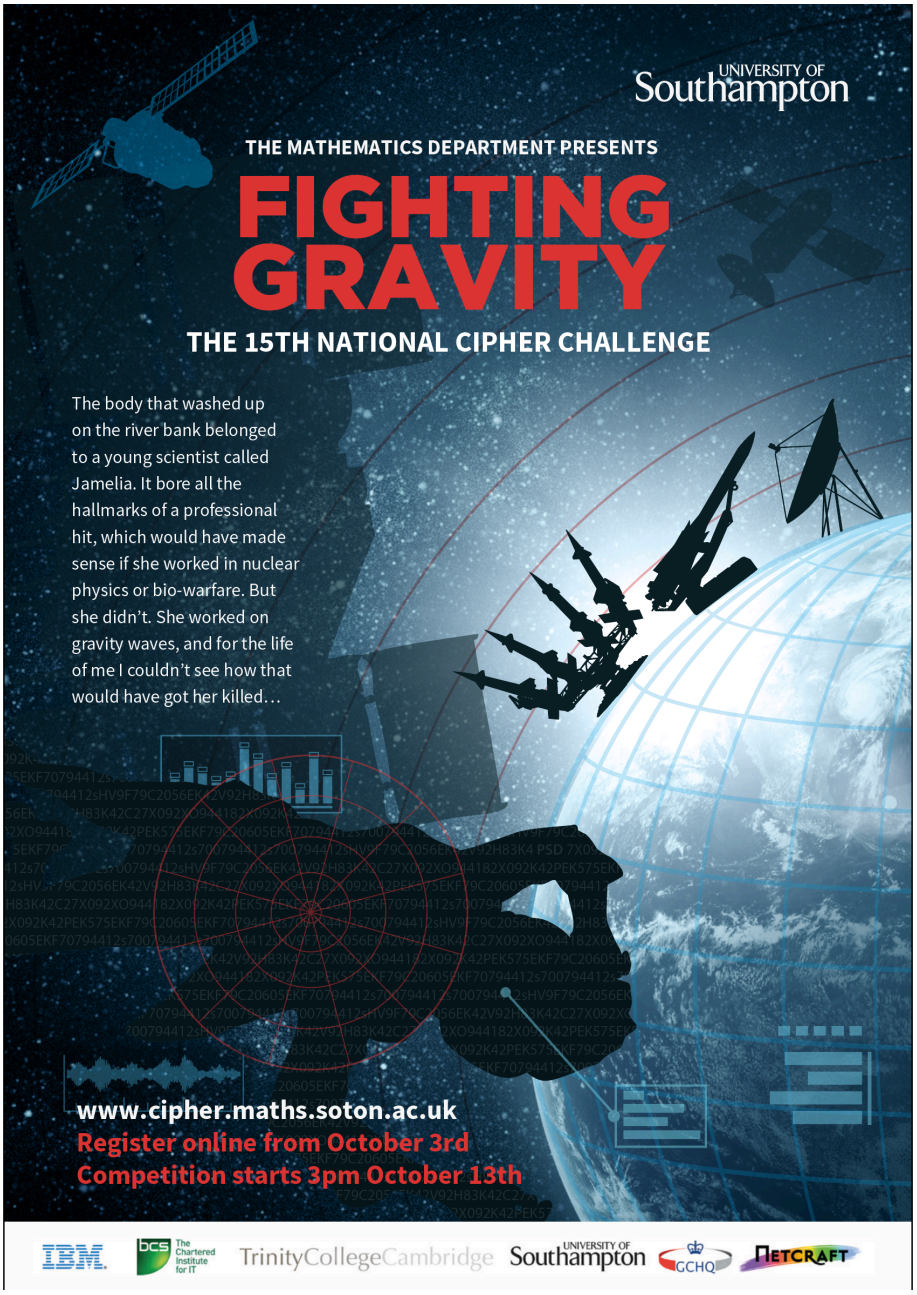
National Cipher Challenge 2016
How the winners cracked 8B



On the 15th December 2016 we published the last part of Fighting Gravity, the fifteenth National Cipher Challenge. Competitors had wrestled with Caesar Shift ciphers, affine shifts, keyword substitutions, Vigenere ciphers and transpositions of several sorts. They were about to be faced with a Hill cipher, based on matrix algebra in mod 26 arithmetic. And that was just part A of the Challenge. Part B, the part that really counted for the final competition standings was an entirely new custom cipher designed by Harry to trip them all up.

The hardest part of designing the challenge is to keep it fresh and to try to stretch the incredibly able competitors who come back year after year to learn more about ciphers and themselves. Sometimes the final

challenge is about processing a lot of data to try to find a key that is hidden like a needle in a haystack. This time Harry tried something a little different.

> *"Challenge 8B came as quite a shock, I was expecting a cipher which had a known algorithm and its strength coming from the vast key space. Enigma, 3x3 Hill and playfair cipher were some I was preparing for. However, 8Bs complexity did not come from a large key space but from that the algorithm was unknown."*
>
> Alex Barter, Gold Medallist.

So how did they do it? Hard work and genius I guess, so maybe we should try to learn from them. Below they have given us a little glimpse into how they tackled the cipher. Maybe this will help you for next year's competition. We hope so. Enjoy!

Alex Barter from The Cotswold School really captures the spirit of experimentation that is the heart of the process. A lot of things he tried didn't work on their own, but did contribute to his understanding of what was going on. Even something as simple as counting the characters really helped:

> "The Challenge looked a bit like this but about 600 times longer…
> 10200 20020 12002 11120 00210 02010 21012 10021 10201 12011 20002
>
> To start with I noticed that it only has 3 characters in; 0, 1 and 2 (quite obvious). I immediately thought it may have something to do ternary (base 3). So I removed the spaces and counted the characters which totalled:
> 32545
> which has the factors…
> 1, 5, 23, 115, 283, 1415, 6509, 32545

So I checked how many unique combinations in the text there were in blocks of all the factors. For blocks of 5 there were 135 combinations, if there was a 26 letter alphabet encoded in ternary I would expect 26 or less (since some letters may not have been contained in the text). So it probably wasn't ternary.

I was then thinking it was some special cipher that only used 3 characters, a cipher I had not come across before. However after searching the internet for a few minutes I did not find anything.

My next thought was that it could be a form of morse code, where the 0 = . (dot), 1 = - (dash) and 2 = x (separator between morse characters). However, I examined the text and found that the longest separation of 2 consecutive 2s was 19; there is no morse character with 19 symbols so it couldn't have been morse. I did test it for 0 = - (dash) and 1 = . (dot) too just in case.

I was a bit stuck so I read though 8A to see if there were any clues, but I could not find any.

I then checked the title (as last year the title was the first sentence of the message) for clues,
"You can't make an omelette without breaking eggs", I thought that could be a clue as when I looked it up online for its meaning, one website said "In order to achieve something, it is inevitable and necessary that something should be destroyed". This gave me that idea that maybe the 2s had been specially put in there to throw us off. I tried converting the 2's to 1's, and 2's to 0's and completely removed the 2's altogether and converted the result from binary into numbers, however for all 3 attempts some numbers were larger than 26 so couldn't have fitted the 26 letter alphabet. I did also note that removing all the 2's resulted in the text still being a multiple of 5 – which could be significant.

I then split the text into blocks between the 2 and wrote them in rows, e.g.
10200200201200211120002100201021012
10
00
00
01
00
111
000
100
010
101

I noticed that each 5 rows (or every multiple of 5 rows in some cases) they were the same length. So I thought that each column of every 5 rows was a letter written in binary (I tried both ways of writing it from top to bottom and bottom to top). So I converted each column to an integer, with the binary number written from top to bottom. To my surprise they were all below 26 which could have meant I successfully converted the numbers to characters. I then put the text into my identify program which uses custom-pre-generated average statistics created from a database of plaintexts of previous challenges and other random texts for many different ciphers. It identified it as a simple substitution cipher, with a very good score.

I then put in my simple substitution solver which uses simulated annealing to find the key and after 2 cycles bam. English!”

Code breaking isn’t always about individual genius, team work was essential in breaking the Enigma cipher even when it depended on deep individual insight. In recognition of this we always award team prizes as well. This year the Gold Medal winning team of Liam Zhou and Benjamin

Dayan cracked Challenge 8B on the first day with a mix of analytic and coding skill. Though it sounds like they might need to work on tidying up and documenting their code:

> *“We've done the challenge for a number of years and have enjoyed it a lot and learnt a lot of cool stuff. Liam and I do have a bunch of code we wrote and applied for the challenges. It's a bit messy, I have python code and Liam has C#. My code is also a bit mixed together.”*
>
> *Benjamin Dayan, Westminster School*

Their key idea was that the lack of 2’s in the cipher text must be significant:

> “We found there weren't very many 2s, which was odd. We tried weird combinations of ascii, sometimes including the 2s or removing them or replacing them with stuff. Then we found that splitting by the 2s yielded such an ordered structure, so we thought that must be it.
>
> So we split the 0s, 1s and 2s by the 2s. You then get a whole bunch of chunks, where there are sets of 5 chunks which all have the same length. So like 01, 10, 11, 11, 00 is one set of 5 chunks. I split up the chunks into these sets of 5. I then took the first digits of the five two digits, so here 0 1 1 1 0 is the first five digits. Next I took the second digit off the set of 5, here 1 0 1 1 0 . Afterwards you go to the next set of 5 and extract their digits. And so on.
>
> Finally I alphabetized the results, as there only 26 five digit binary combinations, despite 32 possible values. I ran it through my mono solver, and that was it.”

Some competitors ask if it is cheating to use a computer to break the challenge, but doing that wouldn't have helped on its own. To crack this

one you needed to think very hard about what was going on. This wasn't a standard cipher like a lot of the ones we use in the Challenge. It was based a little on the bifid cipher, a very well known paper and pencil algorithm, but twisted by the use of 2 as a null separator to remove the regular rhythm of a bifid cipher that is its great weakness. It wasn't possible to plug the cipher text into an elementary cipher cracker and ask it to break the text some real thought had to go into it first.

James Hogge, this year's Silver Medallist, describes his process well:

> "My first thought was that I needed an alphabet that I could work with rather than 1s 2s and 0s. After factorising the length of the ciphertext and seeing that the only small factor was 5, I tried converting 5 letter groups to decimal (assuming that they were ternary numbers) but there were over 140 unique combinations so it couldn't have been a straight map to the alphabet. Another thought I had was that it could be some form of a Gronsfeld cipher where the key had some larger and some smaller numbers. To see if this was true, I checked whether the floor of each number when divided by 26 was periodic. This was not true.
>
> At this point, I started playing around with 2 being used as a special character because I had noticed that it never occurred in groups larger than one so the next thing I checked was the number of 2s in the message. This was also a multiple of 5. This seemed like a good lead because once the twos are removed, you're left with a binary message and the smallest number of bits you need to represent the entire alphabet is 5.
>
> After researching more classical ciphers on Wikipedia I thought that it could be some form of a null cipher. There was a two at the very end of the message so I thought that perhaps it could be that every number preceding a 2 was part of the actual message and the rest was actually gibberish. When this part of the message was put together and split into 5 digit groups,

there were only 22 unique groups. Less than 26 though so it could have been a possibility. There was the mention of eggs in the title so I tried using the baconian alphabet (excluding J and V) as well as the two common 25 letter alphabets that seem to occur in classical cryptography (excluding J or excluding Q) and finally the whole alphabet. I tried 2x2, 3x3 Hill ciphers, monoalphabetic substitution and Bifid (where appropriate) however nothing gave results.

After that I tried splitting the ciphertext on the 2s. This was more promising because I noticed the pattern where the lengths of the groups always occurred in groups of 5. First idea that came to mind was what happens if I take the first digit of each section and say that that is one character from the ciphertext then the second digit of each section etc. This then gave me the 26 unique 5 letter groups. I arbitrarily assigned them to letters and ran it through Hill and monoalphabetic substitution programs and the monoalphabetic program gave readable English."

Team Amgine from Cedar's Upper School took a similar approach to Benjamin and Liam using a range of tools including spreadsheets, Python scripts and their own brains. They described their thought processes as follows:

"At first, seeing that the text was made up of the digits 0, 1 and 2, we suspected that the message was composed of the letters of the alphabet encoded in 3-trit ternary, but, after trying to convert back to characters, we discovered this resulted in less than 26 different characters. We also briefly considered an incomplete Trifid cipher (encouraged by the Bifid for 7B), but rapidly dismissed this idea. Looking more closely at the text, we saw that twos never appeared next to each other. This implied that the twos were separating the binary digits into blocks of some description.

We then examined the possibility that the twos had been added as filler, serving no purpose other than to confuse the cryptanalysist. However, simply removing them and then converting from 5-bit binary (as was used in the 2014 8B cipher) still yielded more than 26 different values.
Our confidence that the message was encrypted using 5-bit binary was boosted by the use of a small spreadsheet to find the relative frequencies of the different digits, which showed that the frequencies of '1' and '0' were indeed similar to those of English text (using 26 characters) encoded in 5-bit binary.
On closer inspection, we noticed that the blocks of binary digits (obtained by removing twos) were grouped in sets of 5 blocks of the same length, but there was a large variation between the lengths of different sets of 5 blocks. We immediately wondered whether these groups of 5 blocks corresponded to the five digits of 5-bit binary. However, we knew that this would result in an abnormally long message (>5000 characters), so we frantically racked our brains and the internet for other possibilities - until we remembered that part A had also been unusually long.
The simplest way of converting the binary back to text, using the blocks we had seemed to be to take every set of 5 blocks of digits that were of equal length, stack these blocks on top of each other within the set and read down in columns to give us our 5-bit numbers, repeating this for every set of 5 blocks, which would give us a list of binary numbers between 0 and 31. We wrote a program to do this, converting the binary to base-10, then using this to produce characters, and the result was a very long string of strange characters- but only 26 characters (of 32 possible) were present. We had effectively cracked the first stage of encryption but we were not all the way there yet. We altered our program to decode the binary rearranging the digits using every permutation possible within the sets of 5 blocks- every possible way to stack the blocks on top of each other in columns of 5, in case this was another stage of encryption. This did not produce anything like English, but

the last few decrypts all contained only the numbers 0-25 which could represent the alphabetic characters.
We tried converting the numbers to the 26 different alphabetic characters then fed one of these results into our assistance interface- a python 3 user interface that uses most of the many python programs that we have written over four years of cipher challenges, and attempts to identify the type of cipher using various tests. On entering the alphabetic text into the interface, the program told us that the text had an index of coincidence of about 1.7- this means that the text has a distribution of letter frequencies very different to what would be expected if each character had an equal chance of occurring. This strongly suggested that the text was a substitution cipher- letters in English have very different frequencies, and index of coincidence is not affected by Monoalphabetic Substitution.
The interface correctly identified the cipher, and from it we initiated our substitution breaker program In less than a minute, we had an answer, which we then submitted, and began to hurriedly add spaces and read the message to check that our program had not got a pair of very infrequent letters wrong. It had not – Success!

Sometimes inspiration strikes in odd ways. One of our Bronze Medallists, Elizaveta Sheremetyev, found it in the Christmas card I posted on the forums a few hours before the competition:

"*Harry's Christmas card, that was posted way too early to be a Christmas message and strangely enough almost exactly 2 hours before Challenge 8 was released, seemed like a hint. It had sequences of 1,3,7,9,13,8,2 stars and I didn't see how these corresponded to words or letters in any way. The Christmas card seemed to have a wave pattern, which brought me to the idea of printing out the blocks of binary, that I got from splitting the cipher text at the 2's, vertically.*
I saw that the blocks of binary were in fives of the same length, there are 32 different numbers that you could represent with 5-

> *bits which covers the number of letters in the alphabet and is not much bigger that it. It seemed likely that the next step would be to read the code downwards."*

The Christmas card wasn't actually intended as a clue, but in thinking around the problem Elizaveta found a different way of looking at the cipher that led directly to the solution. I am reminded of the quote from Robert Harris's novel Enigma:

> *"It was hard going, but Jericho didn't mind. He was taking action, that was the point. It was the same as code-breaking. However hopeless the situation, the rule was always to do something. No cryptogram, Alan Turing used to say, was ever solved by simply staring at it."*

Having reduced the encryption to a standard substitution cipher Elizaveta fed it into a programme she had written to crack it. This was a simulated annealing programme, which is a fancy way of saying that it makes intelligent guesses by trying random keywords and testing the solution they give for how realistic it is, keeping the best solutions and improving them a bit and trying again.

> I had a simulated annealing algorithm that generated a random shuffled key to decode the cipher text with, for the next 1000 iterations it worked from that key randomly swapping 2 letters. If within 1000 iterations it hadn't found a solution it would start with a new randomly shuffled key.
> To judge if a key is better than the previous one I used a file with quadgram frequencies (http://practicalcryptography.com characterisation/quadgrams/). I made a dictionary where each quadgram corresponded to a logarithm
> of that quadgram's frequency count divided by the total number of appearing quadgrams. To judge the
> "quality" of the decoded text I summed the logarithms of the quadgrams appearing in that text to get a

number, the bigger the number - the closer the text is to English.

I hope this gives you some idea how to go about analysing a strange new cipher and to exploit its weaknesses. You can download our codebreaking handbook from the National Cipher Challenge website to get more hints and tips on how to get started. We will be updating it (and the website) with more information ready for the new competition in October. Please do hang around!