Part XI
Cipher machines

Unit 177
Enigma

The *Enigma* is a cipher machine that was first invented for commercial use, but was used by Germany (and others) in World War II. No book on mechanical cryptography would be complete without a description of the machine and some of its variations, but for the history around the Enigma, we refer you to Kahn's books cited in the references.

The significant innovation in the Enigma is the use of *rotors* (also called *wheels*) to generate permutations of the alphabet in a very-long-period polyalphabetic substitution. This innovation spread beyond Germany and was incorporated into cipher machines in America, Russia, and elsewhere; in Units 179 and 180 we will look at two such machines. A *rotor* is a wheel with a pin for each letter of the alphabet on one side. On the other side are the same number of connections. Inside, the rotor has hidden wiring that connects each pin on one side to one connector on the other. The result is a permutation of the alphabet, whose letters are sent through the rotor with an electric current. When a rotor is rotated to a new position, the permutation changes. By putting several rotors in sequence, and by rotating them separately, the alphabetic permutation can be different for every character in the plaintext.

Figure 177.1: An Enigma with four rotors. Photo by Greg Goebel

The Enigma was originally configured to use three removable and replaceable rotors, and later the German navy added a fourth. There also was a removable and replaceable *reflector*, so that the signal for a letter of the plaintext left the keyboard, passed through the three rotors, the reflector, and the inverse of the three rotors, to emerge on a wire that lit an indicator representing the ciphertext letter (see Figure 177.2). The machine is laid out such that the signal enters the rightmost rotor first, which happens to be the one that we list last in the key. The rotor listed first is leftmost and closest to the reflector. The reflector by its very nature must have a self-reciprocal (involutory) (degree-2) permutation. The overall permutation of the machine is therefore also self-reciprocal. In the following equation in operator notation, we use U for the reflector ("Umkehrwalze" in German) and W for the rotors ("Walzen" in German), to distinguish them both from the symbol R that we use for the Caesar rotation. The current positions of the rotors are denoted with r. From the equation, we can see that the overall permutation is a conjugation of the reflector's permutation; therefore the overall permutation is also self-reciprocal and has degree 2.

$$c = E\,p = W_3(r_3)^{-1}\,W_2(r_2)^{-1}\,W_1(r_1)^{-1}\,U\,W_1(r_1)\,W_2(r_2)\,W_3(r_3)\;p = X^{-1}\,U\,X\;p \qquad (177.1)$$

Because of this property, a letter is never enciphered to itself; this weakness makes it easier to place cribs and helped cryptanalysts break the Enigma during the war. See Tables 177.1 and 177.2 for the permutations of the reflectors and rotors that are relevant to Germany and its military. For those models using four rotors, the reflector must be one of the thin ones, and the first rotor must be β (beta) or γ (gamma). In such machines, the first rotor is placed in a position specified by the key and does not rotate. Reflectors are always placed in the same position.

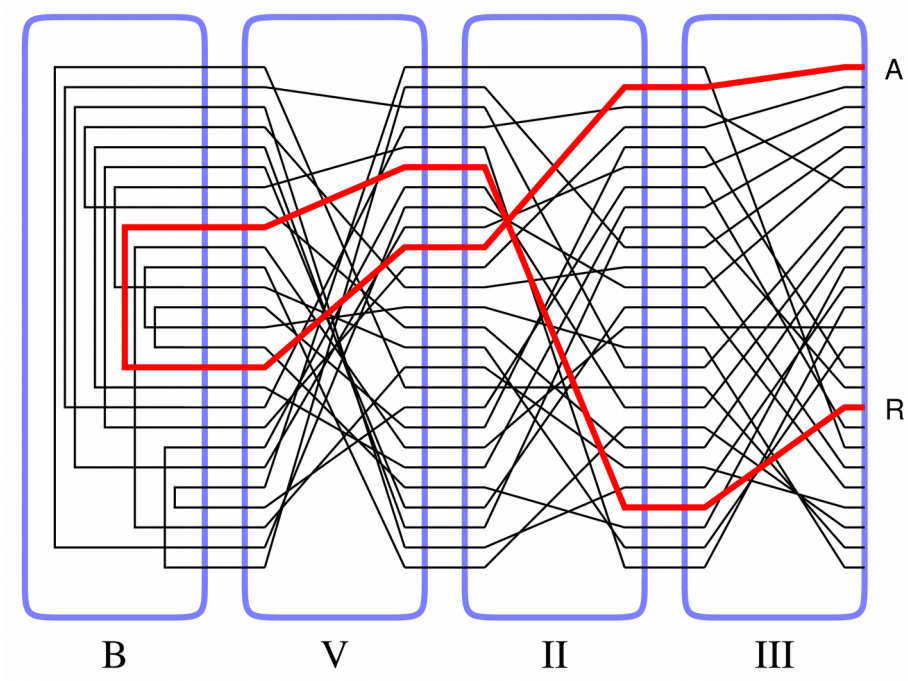


Figure 177.2: Diagram illustrating how the signal passes through the rotors and reflector of a three-rotor Enigma machine. The three rotors are all in position 0. *The path changes when one or more rotors advance.*

The β and γ rotors are thinner than the others, so that one of them with a thin reflector occupied the same width as one of the normal-sized reflectors. Furthermore, there are configurations of thin reflector with thin rotor that mimic the thick reflectors so that the four-rotor machine is backward-compatible with the three-rotor machine. In terms of alphabetic permutations, for this to occur, we need the thick reflector to be the conjugate of a thin reflector:

$$W_{\text{thin}}(0)^{-1}\, U_{\text{thin}}\, W_{\text{thin}}(0) \;=\; U_{\text{thick}} \tag{177.2}$$

The choices and positions of the thin components that satisfy Equation 177.2 are B thin with β in position 0 and C thin with γ in position 0, where by "position 0" we mean the position in which their effective alphabetic permutations are those that appear in Table 177.2.

We would like to point out that the rotation of a rotor is equivalent to sandwiching its operator between a Caesar rotation R of the alphabet and its inverse (look back at Unit 20):

$$W(r) \;=\; R(r)^{-1}\, W(0)\, R(r) \;=\; R(-r)\, W(0)\, R(r)$$

Then we can rewrite the encryption operator in Equation 177.1 as

$$\begin{aligned}
E \;&=\; W_3(r_3)^{-1}\, W_2(r_2)^{-1}\, W_1(r_1)^{-1}\, U\, W_1(r_1)\, W_2(r_2)\, W_3(r_3) \\
&=\; R(-r_3)\, W_3(0)^{-1}\, R(r_3)\, R(-r_2)\, W_2(0)^{-1}\, R(r_2)\, R(-r_1)\, W_1(0)^{-1}\, R(r_1) \\
&\quad\; U\, R(-r_1)\, W_1(0)\, R(r_1)\, R(-r_2)\, W_2(0)\, R(r_2)\, R(-r_3)\, W_3(0)\, R(r_3) \\
&=\; R(-r_3)\, W_3(0)^{-1}\, R(r_3-r_2)\, W_2(0)^{-1}\, R(r_2-r_1)\, W_1(0)^{-1}\, R(r_1) \\
&\quad\; U\, R(-r_1)\, W_1(0)\, R(r_1-r_2)\, W_2(0)\, R(r_2-r_3)\, W_3(0)\, R(r_3)
\end{aligned}$$

In other words, we can think of the rotors in the Enigma as remaining stationary, with changing Caesar shifts inserted between and aside them. Including the ring settings would have the same effect as changing the rotor positions in the above equation.

There is a further complication regarding the rotors. At first, the letter labels on their edges were printed directly onto them, but later they were printed on rings that could be fixed onto the rotors in different positions. This does not affect the way the alphabetic permutations are applied, but it does introduce offsets into the key (see below for how the key is specified) and does affect how the rotors advance between the encipherments of the plaintext letters (also see below regarding rotor advancement).

We must now discuss the rotation of the rotors in the Enigma machine. When the operator presses a key on the keyboard, the advancement of the rotors occurs before the plaintext letter is enciphered. The rightmost rotor (which is listed last in the key) advances one step every time a key is pressed. There are one or two notches on the rings for each rotor (except β and γ) that control the advancement of the rotor to its left. See Table 177.3 for the locations of the notches. Since beta and gamma can only be used as the rotor next to the reflector, they do not have these notches. If a rotor is in a notched position, then when it advances it causes the rotor to its left also to advance. The Enigma machine has a bug (which programmers now call a "feature") known as *double-stepping*: When the middle rotor reaches its notched position, then on the next key-press, it advances again along with the rotor to its left. For example, suppose that the rotors in use are I, II, and III. Rotor III is notched at position V, and rotor II at position E. So the middle rotor advances after the rightmost rotor reaches V,

as in ..., AAU, AAV, ABW, ABX, ... When rotor II reaches E, the double-stepping occurs, as in ..., ADU, ADV, AEW, BFX, BFY, ...

Later models had a *plugboard* ("Steckerbrett") that allowed the transposition of up to ten pairs of letters before they entered the rotors. After leaving the rotors, the plugboard applied the same transpositions again. This had the effect of adding another alphabetic permutation (S) to Equation 177.1:

$$c \;=\; S\, W_3(r_3)^{-1}\, W_2(r_2)^{-1}\, W_1(r_1)^{-1}\, U\, W_1(r_1)\, W_2(r_2)\, W_3(r_3)\, S\, p$$

We did not write "S^{-1}" on the left because S is its own inverse.

The encryption key of the Enigma consists of these things:

- the choice of reflector
- the choice of the three or four rotors and the order in which they are placed
- the ring setting for each rotor (one number [1-26] for each)
- the plugboard connections (a set of up to ten pairs of letters, with no repetition)
- the starting positions of the rotors (a sequence of three or four letters)

Typically, all but the last of these were configured each morning as the internal daily settings. The machine would be locked and only the starting positions of the rotors could be changed during the day. They served as the message key and were, ideally, different for each message during the course of any given day.

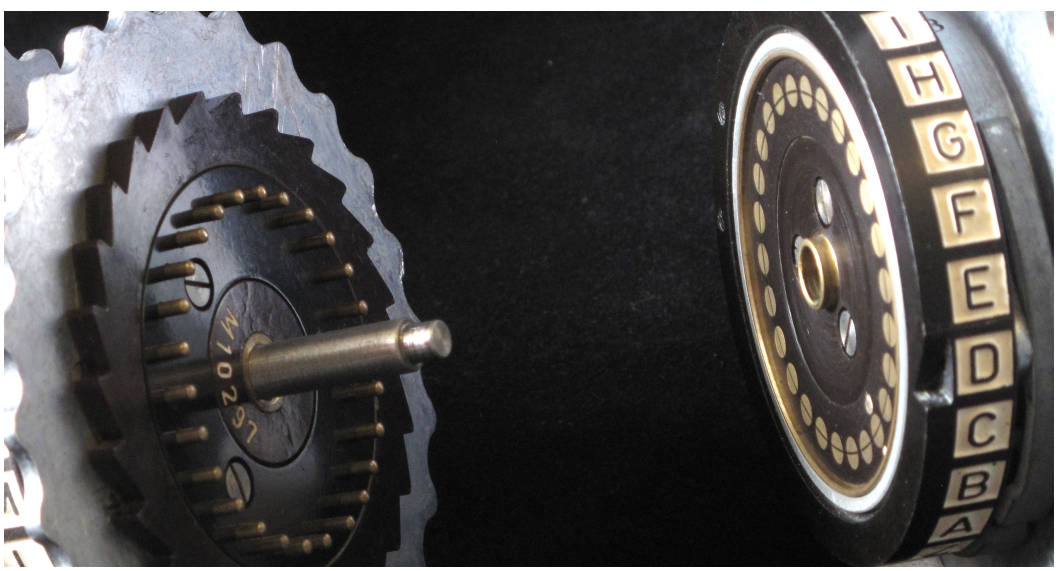


Figure 177.3: The two sides of rotors used in the Enigma. One side has pins, the other pads. The advancement notch is visible in the ring on the rotor to the right. Photo by Ted Coles.

Let’s try an example of encryption with the Enigma machine. Take this plaintext:

THISXMESSAGEXISXENCRYPTEDXWITHXTHEXENIGMA

and use the key B, II, I, VI, 5, 7, 9, GELATINOUS, GOO. On the first press of a key, the rightmost rotor (VI) advances to P and the keyboard emits a signal on the line representing T. The signal passes through the plugboard to emerge on the line for I, since (I T) is one of the transpositions specified in the key. Next, the signal enters rotor VI, but since the ring setting is 9 and the rotor is in position P, it enters the P pin (8−8+15 = 15 modulo 26, and we must take 1 less than the ring setting because we consider A to be zero). Rotor VI then emits its signal on the pad for R (see Table 177.2), which becomes K after we add the 8 and subtract the 15. The middle rotor (I) has its ring at 7 and is in position O, so we subtract 6 and add 14 to K to find that the signal enters on the S pin. The signal leaves rotor I on the S pad, which becomes K after we add the 6 and subtract the 14. The leftmost rotor (II) has ring setting 5 and is at position G, so we subtract 4 and add 6 to see that the signal enters on the M pin and leaves on the W pad. Adding the 4 and subtracting the 6 gives a signal entering reflector B on the U pin. The signal leaves the reflector on its C pin. Subtract 4 and add 6 to see that it enters the E pad of rotor II, so it leaves via the Z pin. Add back the 4 and subtract the 6, then subtract 6 and add 14, to find that the signal enters rotor I on the F pad and leaves on the D pin. Add back the 6 and subtract the 14, and subtract 8 and add 15, to find that it enters rotor VI on the C pad and leaves on the X pin. Add back the 8 and subtract 15 back into it to get a signal on the Q wire. The plugboard does not affect Q, so theis is the ciphertext letter. This painful process continues until the entire text is enciphered. The complete ciphertext is

QXRYIASMEJFZDGZMHYNBBRLFNRDEQDVPNMHAEJAYH

If you work through the details, you will notice that a double-stepping occurs when the D of ENCRYPTED and the following X are enciphered. The positions of the rotors go from GPM to GQN to HRO.

reflector	input ABCDEFGHIJKLMNOPQRSTUVWXYZ
A	EJMZALYXVBWFCRQUONTSPIKHGD
B	YRUHQSLDPXNGOKMIEBFZCWVJAT
C	FVPJIAOYEDRZXWGCTKUQSBNMHL
B thin	ENKQAUYWJICOPBLMDXZVFTHRGS
C thin	RDOBJNTKVEHMLFCWZAXGYIPSUQ

Table 177.1: Alphabetic permutations of the Enigma's reflectors. Later versions of the machine have a D *rotor whose wiring is changed as part of the key.*

rotor	input ABCDEFGHIJKLMNOPQRSTUVWXYZ
I	EKMFLGDQVZNTOWYHXUSPAIBRCJ
II	AJDKSIRUXBLHWTMCQGZNPYFVOE
III	BDFHJLCPRTXVZNYEIWGAKMUSQO
IV	ESOVPZJAYQUIRHXLNFTGKDCMWB
V	VZBRGITYUPSDNHLXAWMJQOFECK
VI	JPGVOUMFYQBENHZRDKASXLICTW
VII	NZJHGRCXMYSWBOUFAIVLPEKQDT
VIII	FKQHTLXOCBJSPDZRAMEWNIUYGV
β	LEYJVCNIXWPBQMDRTAKZGFUHOS
γ	FSOKANUERHMBTIYCWLQPZXVGJD

Table 177.2: Alphabetic permutations of the Enigma's rotors in position 0*. In the forward direction, input is on a pin and output on a pad.*

rotor	location of notch(es)
I	Q
II	E
III	V
IV	J
V	Z
VI	M, Z
VII	M, Z
VIII	M, Z

Table 177.3: Location of advancement notches on rotors in the Enigma.

Reading and references

Mark Stamp and Richard M. Low, *Applied Cryptanalysis: Breaking Ciphers in the Real World*, Hoboken: Wiley, 2007, section 2.2.

Wikipedia has many pages on the Enigma. Start with these:
https://en.wikipedia.org/wiki/Enigma_machine
https://en.wikipedia.org/wiki/Enigma_rotor_details

Philip Marks and Frode Weierud, "Recovering the Wiring of Enigma's *Umkehrwalze* A," *Cryptologia* 24:1 (2000) 55-66, https://doi.org/10.1080/0161-110091888781

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, pages 420-423 and elsewhere, https://ia600606.us.archive.org/30/items/B-001-001-264/B-001-001-264.pdf, https://archive.org/details/B-001-001-264

David Kahn, *Seizing the Enigma: The Race to Break the German U-Boat Codes, 1939-1943*, New York: Houghton Mifflin, 1991.

Programming tasks

1. Simulate the Enigma in Python. Allow for the possibilities that either three or four rotors are used. Implement all reflectors and rotors that are presented in this unit.

Exercises

1. Decipher this ciphertext. The key is B, I, V, III, 5, 9, 15, `PS`, `AY`, `IJ`, `ER`, `VN`, `MAX`. An `X` has been placed between words in the plaintext, and `XX` between sentences.

```
LZJWKMOPGWGKESNUONOICEWIAULJIQTLRUZKAANKPWWGVQHRCWVZZERYGGZBBLCY
BSZWPLWGSVUWFRKGLHPRJDHJYOSTDPKRGAMJEZCNFMQGBAHYBALYMUJCPOOJGEDN
NKHBHMAFMUGLQDCYPESQZDIYOQWQSTZSUMDCMQSSXKHGYKMLTZKFDXYZCFXKWJZA
EPSVIZVUIQAVWGRJAW
```

2. Decipher this ciphertext. The key is C, II, IV, VI, 13, 25, 7, `DECRYPTION`, `ABC`. An `X` has been placed between words in the plaintext, and `XX` between sentences. This exercise will test your implementation of double-stepping.

```
TFELTJWHTSFJBHTRBBOWLHNRLMNDWHGZZIGNYHWKKWJQTNCKQQTNWNJRSWDJPGDM
ILQOIYPPYJUZFWCYZKPJZSWISAISUJECOQAHCFVUDJTQQRYEHVUGEIYHRMDBXNNB
IFOZDIARLAOXTFYMIWAPRACSFXZDXADRKJNJUVVVNZMLYZZOYAUCTMHIVVPYCXVI
DTQDEZGVEGMMTCXZUIGRUCRCDWDXEYHMSVVVKKLJWJWAWYONPYBUKWKBIESHKYYF
RGQAMDIXEBXWQSXLPG
```

3. Verify Equation 177.2 for U_{thick} = B, U_{thin} = B thin, $W_{\text{thin}} = \beta$ in position 0 and for U_{thick} = C, U_{thin} = C thin, $W_{\text{thin}} = \gamma$ in position 0.

Unit 178
Attacking Enigma with the index of coincidence

Analysts during World War II used every bit of their computing power to break the Enigma. In the modern world, we have so much computing power that we waste most of it. A statistical attack like the one we present here uses a lot of computation. We are going to use the index of coincidence (IoC) (see Unit 12) to help us find the configuration of an Enigma machine that was used to encrypt a ciphertext. In this unit we will restrict ourselves to a three-rotor machine, but the method can be generalized to four-rotor versions.

The attack has these three stages, which we will discuss in more depth below.

1. Try all choices of reflector and rotors and (provisional) initial positions and keep the configuration and starting positions with the highest IoC.
2. Keep the reflector and rotors from stage 1, but vary the ring settings and initial positions and keep those with the highest IoC.
3. Try plugboard wirings one pair of letters at a time and maximize the IoC.

In the first stage, we try all possible combinations of reflector and rotors. Remember that a rotor cannot be used twice in the same configuration, since each machine was outfitted with one of each. We also try all possible starting positions for the rotors (the message key). In this stage, we fix the ring settings to be at 1=A for all rotors (we could take any arbitrary ring setting, but it should not matter much). There are three reflectors, eight rotors, and twenty-six possible starting positions for each rotor, so we have to try $3 \cdot 8 \cdot 7 \cdot 6 \cdot 26^3 = 17{,}716{,}608$ configurations, and therefore we recommend a compiled programming language such as C, rather than a slow interpreted one like Python. The configuration whose resulting plaintext has the highest IoC is kept for stage 2. The winning margin may not be large, so we should take some care; if there is no clear winner, perhaps the method has failed and we need to try a more exhaustive search (see below).

We consider the initial positions of the rotors found in stage 1 to be provisional because the true ring setting can affect whether the rotor acts more like it starts in one position or more like in a neighboring position. So in this stage we will give some leeway to the rotor positions, and allow it to have a starting position one place earlier in the alphabet. The ring setting on the leftmost rotor is irrelevant for a ciphertext-only attack because it only affects the locations of the stepping notches, which aren't used for that rotor. This ring setting is important for cipher clerks, because it affects the message key. But for us, we can fix the ring position to 1 and not allow it to vary. As we increase the ring setting of a rotor, the starting position increases with it. If, for example, in stage 1 we found the best starting position for the middle rotor to be G, then we should try these possibilities for that rotor (remember we give it some leeway):

ring setting 1	starting position F
ring setting 1	starting position G

ring setting 2	starting position G
ring setting 2	starting position H
ring setting 3	starting position H
ring setting 3	starting position I
⋮	⋮

We need to also give leeway to the starting position of the leftmost rotor, even though we do not vary its ring setting. So there are $2\cdot(2\cdot 26)^2 = 5{,}405$ possibilities to try. The one that gives a plaintext with the highest IoC is kept. In this stage, there should be a clear winner.

The action of the plugboard causes more apparent randomness for the cryptanalyst. But for each pair of letters swapped by the plugboard that is correctly identified, the randomness can be reduced. Hence, every time we account for a pair of letters cabled on the plugboard, the IoC of our plaintext will increase. To find the first one, we try $26\cdot 25/2 = 325$ pairs (divide by 2 because AB is the same as BA, etc.). We keep the pair that gives the greatest increase in IoC and include it in all future decryptions. To find the second pair, we try $24\cdot 23/2 = 276$ pairs (we don't include the letters of the first pair in the search). When we reach a point where adding another pair to the plugboard reduces, rather than increases, the IoC of the plaintext, then we stop. At this point, the attack is finished.

We must, of course, look at an example. Consider this ciphertext that was encrypted with a three-rotor enigma. It is long enough that we are confident of success.

```
NNBOFZRDRLRSYRRVJPSXPSZMWHUHESXPFKKHYRNUFYMYXQJPMOLTPMRTKKCVJQNT
HAPVQBFHJWVRXWDOMQNWRJOJXIPBBHZVLPSGZJCTVUMMUUEFHEKNYHWIXVCJQKSJ
IEZLSUXZLRLKJBOYCZHNYUCKWPAIYCMUTKUFHMCOXWLEMSYWKZYBOFZCPTQZKNZQ
DDGTPHRMFREBNMQUFMRELRGAFXVRWKAOEVKBQIMGMTIBKBGLRRAYBOIRXEQLFOZA
QTDFDNUAGDHVATDTBSOXAFTHZAHWKAWACQKLRKCJTNQEWODQJEUMNQXKCAYQVVBL
KMDQNIMEHKVMHENUTIPTAPNCQFPHVKEJLDEIDGXGRYJAZJILHLJUEJNYBFTUPEMP
FXUXBICDFBUDYLMJMDKTUHWKKLOKJNAXRIUDIHVSDJTSTRRNNZKIAVEYASORARRS
TKUWJNTFYCSOSQEFGZKJPFDDHPXSUJTRFSCZPJKPZVQLNNTGODWCTCKEUBJWXNNA
WFBJUQXEDYIVTYJMNLSZQQMQVKVNNORUFPZPXTHZQIDHTFDYYXVJMAXLWDWHMUWS
BKTFYMJPKGNGILAUPPVXNWLUDAISQPLZUZDGXRFOHUYJRRBHWIFHVBJQXBGVAFTT
IZGXXZVGOEOUUUXSUXJKMBSEPTLOBZITMNIKYQBHMVIIHAOJNILNNIKCEKERRAOY
YNPWLCYQRGUALYOZFUZVSHXDGKLSDRLFZVKPDZVMKUHMJNMZXBSWLZGIURVJIQZX
STUVXIXLDHOKNEDJLXCKMEMDWVWAMPZZZOBZYVHMDNCJHXCOJHZUHEPYWHKRJMVH
STVUPGRZEJZBTOJRCTQFYLOQFCWMDDBKHRKUXXUOVYBOQZTGJBQKDBQLENNNXBOV
OTSOVMSXGIMIGXFIPMEFFUAWUYZBQBEAXJFDWBDUAAPRYRXNYBMIH
```

In the first stage of attack, we try all possible reflectors, rotors, and starting positions. Ring settings are all set to 1. We sort from lowest to highest IoC and find that the best is when the reflector is B, the rotors are II, III, V, and the starting positions are KEZ. It's a narrow victory, but we will run with it. If we find that we cannot solve the ciphertext, then we can always come back and try the second-place configuration.

reflector	rotors	starting positions	IoC
⋮	⋮	⋮	⋮
A	VI IV VIII	IZS	1.0603

C	VIII II V	QHT	1.0607
B	V VII VI	NIK	1.0625
B	II III V	KEZ	1.0635

We keep the choices of reflector and rotors, and now concentrate on the ring settings. For the fast rotor (on the right), we found our result above for ring setting 1 and initial position Z. So we try 2 and A, 3 and B, ..., 26 and Y. But because of the uncertainty we have in the starting positions, we try these plus or minus one step. At the same time, we try varying the ring setting and starting position in a similar fashion for the middle rotor. For the slow rotor, to be safe, we allow the starting position to vary by one step, but still keep ring setting 1. After the search, here is the end of our list, sorted by IoC:

ring settings	starting positions	IoC
⋮	⋮	⋮
1 17 7	J U F	1.2585
1 16 8	J T G	1.2659
1 17 9	J U H	1.2833
1 17 8	J U G	1.2943

The best is for ring settings 1, 17, 8 and starting positions JUG. We continue with the assumption that we have found the correct settings for the rotors and now turn (pun intended) our attention to the plugboard. There are (26 · 25) / 2 = 325 ways to connect two letters. If we decrypt the ciphertext with our favorite rotor settings and each possible choice for one wire on the plugboard, we find these IoCs:

plugboard pair	IoC
R V	1.1833
K R	1.1899
⋮	⋮
C F	1.3799
P T	1.3888
B X	1.3912

The last few are all good choices, but to follow our method, we keep only the best pair, BX, and repeat the procedure assuming that this pair is correct and look for a second pair. This time, we only need to consider (24 · 23) / 2 = 276 new pairs.

plugboard pair	IoC
R V	1.2652
I V	1.2686
⋮	⋮
C F	1.4838
A L	1.4876
P T	1.5027

So we keep the pair `PT` and continue to find the next pair to be `AL`, giving an IoC of 1.6081. We are now in the range of typical English text, so let's pause for a moment and look at the resulting plaintext:

```
ZEENTRMAGEGINMAOCMOGSIEURBOURDOGRSMAIGHYXGOBJEFTIVEIMPARTIHLAGSI
MPERSOAALSTUDYAGD...
```

OK, we're not finished yet. We can see some words peeking out at us, but it is clearly not correct. So we find a fourth pair, `CF`, with IoC 1.7200. Let's pause again and see the plaintext:

```
ZEENERMAGEGINMAOFMOGSIEURBOURDOGISMAIGHYAGOBJECTIVEIMPARTIALAGSI
MPERSOGALSTUDYAGD...
```

There is marginal improvement. Searching for a fifth pair reveals `GN`, with IoC 1.7960. Trying to find a sixth pair only *reduces* the IoC for this ciphertext, so we may be at the end. When we check, the plaintext now looks correct:

```
THEGERMANENIGMAOFMONSIEURBOURDONISMAINLYANOBJECTIVEIMPARTIALAND
IMPERSONALSTUDYAND...
```

We have found the entire key. The plaintext is from *German Problems and Personalities* by Charles Sarolea:

> The "German Enigma" of Monsieur Bourdon is mainly an objective, impartial, and impersonal study, and the author has been careful not to obtrude his own private views. It is only in the last chapter that he attempts to draw the lesson and point out the conclusion of his own inquiry. And his conclusion is an eloquent though restrained plea for a Franco-German *rapprochement*, and in favour of the only policy which will bring about that reconciliation. France, he argues, does not want a revision of the Treaty of Frankfurt. She does not want compensation or revenge. ... A French statesman, on the eve of the Treaty of Frankfurt, made the rhetorical statement that France would never surrender one stone of her fortresses nor one inch of her territory. Animated by a very different spirit, modern French statesmen do not claim back to-day one inch of lost territory. All that the French people demand is that the claims of justice shall be heard, that Alsace-Lorraine shall cease to groan under the heel of an arbitrary despot, that Alsace-Lorraine shall be governed according to her own laws, that the Alsatians shall be treated as a free people, and not as conquered subjects.

In the case where too many cables are used in the plugboard, it may be impossible to succeed in stages 1 and 2. When that happens, we need to try all configurations of reflector, rotors, ring settings, and initial positions of the rotors. For each, we try to maximize the IoC in the same way as in stage 3. Only when we find an acceptable plaintext do we stop the search. This can be a long and difficult process, so it would be best to use a fast programming language, such as C.

The index of coincidence is not the only statistic that we can maximize to break Enigma ciphertexts. See references for refinements of the attack and the use of other statistics.

Reading and references

James J. Gillogly, “Ciphertext-Only Cryptanalysis of Enigma,” *Cryptologia* 19:4 (1995) 405-413, https://doi.org/10.1080/0161-119591884060

Note that there are 325 possible two-letter plugboard wirings, not 625.

Heidi Williams, “Applying Statistical Language Recognition Techniques in the Ciphertext-Only Cryptanalysis of Enigma,” *Cryptologia* 24:1 (2000) 4-17, https://doi.org/10.1080/0161-110091888745

Olaf Ostwald and Frode Weierud, “Modern breaking of Enigma ciphertexts,” *Cryptologia* 41:5 (2017) 395-421, https://doi.org/10.1080/01611194.2016.1238423

Programming tasks

1. Implement the attack on a three-rotor Enigma in which it is possible to do stages 1 and 2.

2. Implement the search on an Enigma that has too many plugboard connections for you to succeed with stages 1 and 2.

Exercises

1. We can start off with something easy. This ciphertext was encrypted with a German army Enigma without a plugboard. The army used three rotors chosen from only I, II, III, IV, and V, and reflectors B and C. To increase the IoC, X has been placed between words and XX between sentences.

```
XBCWHIYYBMIUFWJBFMBNCYRABCJABKUHIWUTWUYTIDYMWYHZQWOVFGUQELLAZDGY
BGNYEIWXTHISOPYETEASUBYSVJRDBPEAIWSBBPCQGRRHZDNFMQCOCPISCFNIXZCJ
LCEBZFKDDBOQRHWAEGOFOKHAXVCSVWTPQHYBVTAEIMXHWMUZZLUYRKNCTJEEZXBI
KAUCSBGLDDTPVLFWQXTZRTYBVOYJVXXGWJDZXOEHYPTKTOIPSEGCLWFMOKSABZWD
ESGNSPEFGKNYFIPPJFWTRTKYOSLLGPHLWRJUBVLEFDRDCUIWGYURLWNUPTSHDPOV
MIYDWFXLOPDFAZRPQGWEBEILBDNDCDKDRLHTCIYNOMRYLVZZFNWDHVTBWDXMRHUE
SEEBPOYLJVZXVCACJYZJYSHYAUPMOOHBFYBDSTSOXUVETTSXLSSPLRHRTARRIAVO
QKCRKGWEGXQAUAVSZEDPIAJUNSARRQNJZUWQHOBSTYAILRLECLWUABQJHPROOCPE
ZUJYWIJZMVQGGCEBVHOVGHVVPCBNVVNNXXASQHMKPSZPWLRMMW
```

2. Break this ciphertext that was encrypted with a three-rotor Enigma. The plaintext is in English. To increase the IoC, X has been placed between words and XX between sentences. There are five cables in use on the plugboard. The rotors were chosen from all except beta (β) and gamma (γ). The reflector is B or C. There may come a point where you have to try two forks in the road.

```
CCFZIAPYHFQWOWJULWBHWFGIXPYDRGEDDSBXCGVXLQCPWMXEMYRWJKFUQPFGDESI
PRJIIWBISATECJCFRNAXWCDKQPDDRSPMHACXVJYZLZKABALXTSWISWVCLJNIZOAK
LVLNGHCHFPMLBGGCPKROXBWDUFCRHBNLNJHJJEGVFQNSAOQUZESPDGMHJVCRMAUX
KQRMWQQFBOKDNOANLHTGGHEGVDJGERNNQMFBVPNSWTYPRLPADUCYPNGWBCCQRDRK
MFAZTXICVIAGNBXKZTCWATGRERОIDTDENCZVEUTJIIGGXBEVPBAUEMEKPUWNRLNF
MIJXCAYKVKCKEHKMFKJADUSMCYBTVNLHWFKNPXEHQOGUHMJDRIOBQEWRSMCGSVCO
VYPJLMJCNCEVXBAPKTERFWAKDPVVCZBGXYLUHKTEIJCYRQZZLCPCNFLFHGMSDAUK
```

```
UIWJIVNBRHKRUMNGTBUMOGKJFZTNZMOUKFKCJVKAXNRPVRLDQUMLVMIMFVKGTPRK
TPPZXQUHXKOFDQFZPRBHWOFBUYJDPDQDHLGZKEHRSVRTEGJBWPMRTTZYZHAUAGTR
EOUPMTGLHOHMAKIAUQPUOOIWIGBMGECRQZQVWVBBCLGDDNSAMYMUAAERFVNPVINT
YGSKTYXTCPTBURVRGIXEKIQQMWLWYDOMAAKBUNPEZYMPLMPZVAKKAAFBGGMYQRRA
PDUNFPROCZBBUDJVHGYYHXOAWOZCVLFUVTELHKFGWBKZPTCTXJFLUKVDHXVKOWTK
BNEUAEYMHLGEPILRCVPWEGLIOBIUUZWMEJPPIDJTCRXJVBQUZRHQTBYDEVEFCOLH
JWVGBBSJDEDLGKQSIVVLSTZWBNHLKCNUKSUSFEWIHEDGYTFWWJKHNZKCEMYOZGKV
QUIPGGPSDZALLMDMJUYWIGUSFQUQVPQYZCTMUQTYBAZGTUJFFWQTHJGHUIAGRGSN
ZNBYNLNTZPJJGMDFTWBHUQTYFGVMVPYBNTAODQFSQPYLLDZGYHAGZJIWKTDEORYP
AIHTVDOIZYXSTUWKBLIGCCQTZJUQVEPIEUTTVPLKZJZNYOSTUXYEJJFKZFLOUAWN
NWGXCMYCCRDAGONYQBYBGOJMQPPGIYTRFCVXSZVOTFKNGGCPVLVSPQRBFUJFZKPE
ZHYKTPVWVFBUBIFLQJMLTHBTKE
```

3. Decrypt this ciphertext that was encrypted with a three-rotor enigma. There are ten cables in use on the plugboard. The reflector is C and the rotors are II, I, V. The plaintext is in English; believe it or not, the word “enigma” does not appear in it. Remember that the first ring setting is irrelevant.

```
UOPIIKUDEZGWQFXUVYSTROHVNAYLAKHPJVRIVEPNPTTMHXGJYFWJQPAWMZVQFPYB
FZYMYJKOFROAABKIXSFBOYFWTHVNYMWEJKYWVCHCJMFHORYLNOLLIUYNZETGGZOQ
UUANZXKQWLNYNZEXDHWAKXPXLMDULPTANBGAYTWOBLYVGVFSNQDEQQGVZAMEGADB
JPNYRFVUPEJTZKMKUUJUHOHFJNTBLACTDJFRQTQFPTGWSXMVSRIAKVOJGOQTRZGX
ACYYRZQMSRBOMNRDZGDKYQZJMLOOWGIIBIUMHYBMBXSFSLYQGBWXYUDAPWDGYARW
JGMTDXNDYBYRNZFWTAPLHBZPZADJDBZODAYGTKQCZIZQLKYUFXRDXZJEVYEUCZCF
MUGYMTRCQXNMJGHWYDYLERSXUPNCZICUZHSAPPBTZMTDUWNOKEKRRESQJIKKLQVP
VAUJHXTQJAIOVBOSPENPMWHFERFAWJDLISNIFFGTBBFYCBFJSXOFOZYMWDMPEBIY
VRTDTHKYSXDYGRALISAJBWFQXKVYWTYFMORTYDVPKYXFXWYEOSEWGPIZOQNBPYQH
DVNDNQYRJCAVXZKZIXZEXDSGBEVAVQGTGPSJUUXNTISCYPRMNZWCXVWLRUTBEXYG
UNLBKASRUPIFXKWXCPWQTVIMRHBCMNRSOVGWZZQBFPEVRWCVRGSRMAJBQXLVGEGL
HQRULPWLXYOHPKZMLIKPCZASULNRWJRFMYVVTFBXBPICXHZCOGZQEXPSSOLUIIXD
ZM
```

4. Decrypt this ciphertext that was encrypted with a three-rotor enigma. There are ten cables in use on the plugboard. The plaintext is in English. This could take a long time.

```
HQWSEQGSTBGBHDBGVNYJMZHFZNIFOCTADLAHXYJXCHMQHWMEAXVZGHBMLDJGELOG
DUVVJDHWUHIYAEVCWHQSCLYHVOTKYXMOLYRIDAVVJXYGEBJUYCBKREATBYVWRBQI
EJRARBSXELJJFVJOMSJUDTKIKGQQTUINRAZLGXRTEFSDGDMNFYVAGARQNXLMAQVH
TFNVHCGJXVQQIRGKCGUWHVFZXSQRHKNJCLTKDXZZOJFWXUNLJEAPDSUXNOLDWNNR
MQMOZOOVLKRMGZVTMMKNHZIZXUIQKJPDRQSCBUFRKKDPPPIRPZBDFUOLNOPBOMGV
RFIZXWONCNFKYXPGBDOPKBQDIPLBFSDCLVMAVDFGQPMKYPCSDVKUQBHZPQRHBUZG
LMNTXVGZFFQWPOUFMCUCHLDMGLIHYIBKBDRHBNZDVZYTJJRIGYMDBEHGKCUZPTQW
IDJTZVMWXODOKUWBBGKRWPXHEWJOLEWECGCPDKATIYTJCENQKKTOLCJDIPIZCZPA
CEVIDGSSXLIQMGHLOLT
```

If you enjoy breaking Enigma messages, there are some challenges (with ciphertexts in German) at this site: https://www.ciphermachinesandcryptology.com/en/challenge.htm But note that in Enigma messages, the Germans put `X` between sentences, `J` around proper names, used `Q` for `CH`, tripled letters in designations such as U-boat `U235` becoming `UUUZWEIDREIFUNF`, and, worst of all, wrote in German.

Unit 179
SIGABA

SIGABA, also known as *Electric Cipher Machine Mark 2* (*ECM II*), *M-134-C, ASAM 1*, or *CSP-889*, is a rotor machine used by the United States during and after World War II. In this machine, the signal for a plaintext letter passes through five rotors, similar to those of the Enigma. But there is no reflector, and the stepping of the rotors is controlled by a complicated process. There are ten rotors from which to choose the *cipher wheels*; the other five serve as *control wheels*. In addition, there are five smaller, ten-contact rotors, called *index rotors*, which help in determining the irregular stepping of the cipher wheels. See Figure 179.2 for an overall functional view of the machine.

The ten large rotors are numbered 1 to 10 (or 11 to 20, or 21 to 30, etc.). In the codebooks, rotor 10 is referred to by “0”. Each of these has twenty-six contacts on each side and is labeled on the rim with the letters of the English alphabet, like those of the Enigma machine. In code books, they are referred to by the second digit only. The alphabetic permutations of these rotors are listed in Table 179.1. These rotors can be inserted into a slot in either forward or reverse (“R”) orientation, i.e., a rotor can be turned over so that the signal travels in the reverse direction through it. This does not invert its alphabetic permutation, but does an inversion and a reflection (B↔Z, C↔Y, etc.). We leave it as an exercise to work out the alphabetic permutations of the reversed rotors.

Figure 179.1: SIGABA. Photo by United States Air Force.

The small rotors are labeled on their rims by two-digit numbers. The first digit identifies the rotor, and the second is its current setting. These serve as index rotors to help scramble the signals that control the stepping of the cipher wheels. They are removable and replaceable in any order, but during WW II they were kept in numerical order (1, 2, 3, 4, 5). These rotors are not reversible and do not rotate while encryption or decryption is occurring. Their wirings are given in Table 179.2.

The overall function of the machine during encryption is as follows. The operator presses a letter or space key on the keyboard. The machine translates `Z` to `X` and space to `Z`. A signal on the wire designating the plaintext letter runs through the five cipher wheels from first to fifth. The signal emerges from the fifth wheel on the wire designating the ciphertext letter. Then four signals are sent into the first control wheel, on lines `F`, `G`, `H`, and `I`. They pass through the control wheels in reverse direction, some logic circuits whose actions are listed in Table 179.4, through the index rotors, then more logic (Table 179.5), and out to control the stepping of the cipher wheels. Any cipher wheel that receives a signal steps; if the rotor is inserted in the forward direction, then it steps to the alphabetically preceding letter; if it is inserted in the reverse direction, it steps to the following letter. After the cipher wheels have advanced, the center control wheel advances one step. If it passes from `O` to `N` in the forward orientation, or from `O` to `P` in the reverse orientation, then the control wheel in the fourth slot advances one step. If it, too, passes from `O` to an adjacent letter, then the control wheel in the second slot advances one step. The wheels in the first and last slots do not rotate.

The cipher is not self-reciprocal like the Enigma, so in order to decrypt, signals must pass through the cipher wheels in the reverse direction. Stepping, and the control of it, of course, is the same as in the encryption process.

The internal configuration of the SIGABA serves as its daily key. It includes

- the choices and ordering of five rotors as cipher wheels (five numbers, 0-9)
- the orientation (forward or reverse) of each of the five cipher wheels
- the choices and ordering of five rotors as cipher wheels (five numbers, 0-9)
- the orientation (forward or reverse) of each of the five control wheels
- the ordering of the five index wheels (five digits, 1-5)
- the settings of the five index wheels (five digits)

The lists of five cipher wheels and five control wheels must include all ten of the large rotors; none can appear more than once in the key. Here is an example of a daily key as it might appear in a codebook:

day	control	cipher	index
today	4 1R 9 0 3R	2 6R 5R 7 8	13 26 34 42 59

For each new message, the operator chooses five letters at random for the control-wheel settings and derives the settings of the cipher wheels from them (see below). These ten letters serve as the message key, although the operator only needs to send the control-wheel settings to the receiver.

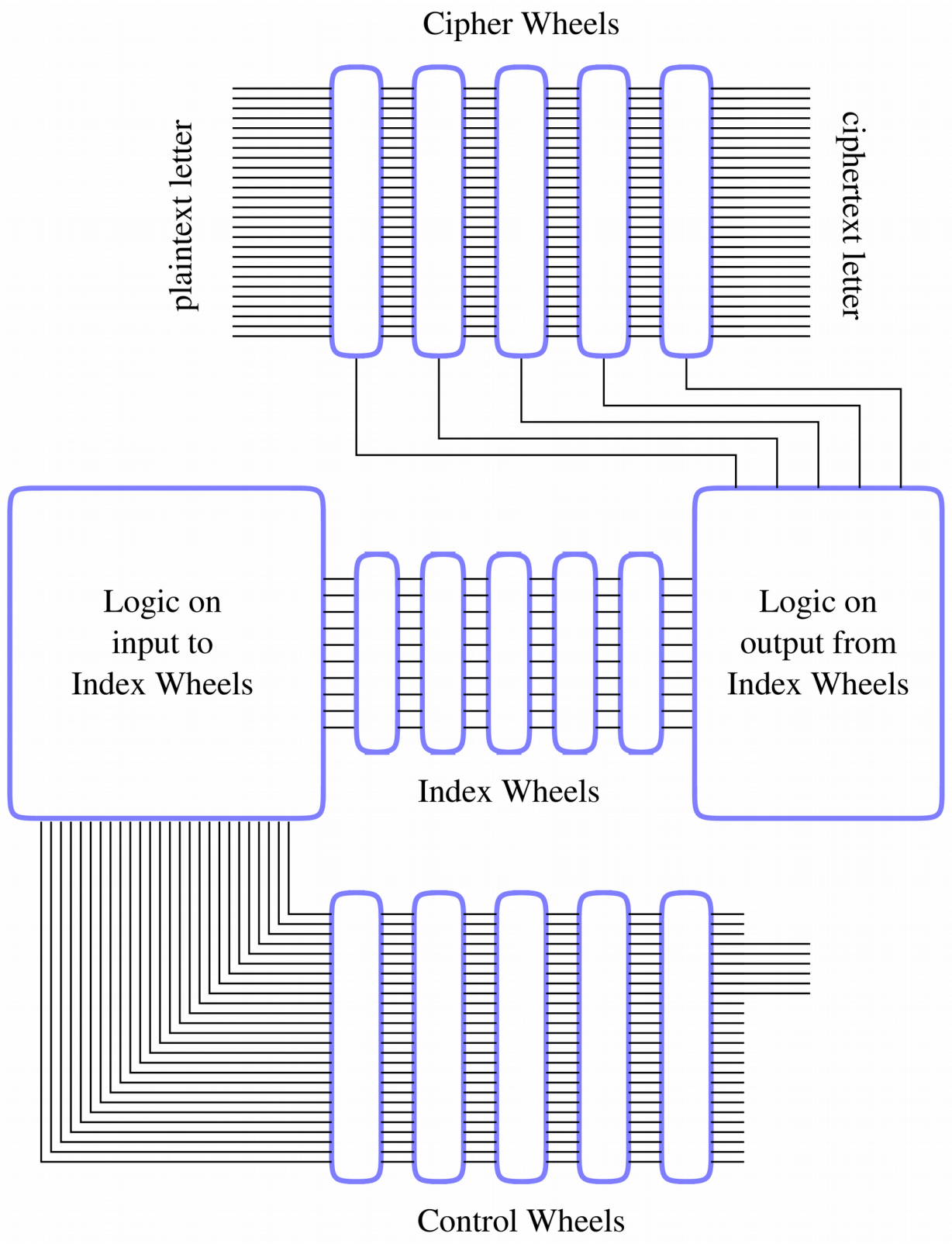


Figure 179.2: Functional overview of SIGABA.

rotor	input ABCDEFGHIJKLMNOPQRSTUVWXYZ
1	INPXBWETGUYSAOCHVLDMQKZJFR
2	WNDRIOZPTAXHFJYQBMSVEKUCGL
3	TZGHOBKRVUXLQDMPNFWCJYEIAS
4	YWTAHRQJVLCEXUNGBIPZMSDFOK
5	QSLRBTEKOGAICFWYVMHJNXZUDP
6	CHJDQIGNBSAKVTUOXFWLEPRMZY
7	CDFAJXTIMNBEQHSUGRYLWZKVPO
8	XHFESZDNRBCGKQIJLTVMUOYAPW
9	EZJQXMOGYTCSFRIUPVNADLHWBK
[1]0	YCHLQSUGBDIXNZKERPVJTAWFOM

Table 179.1: Alphabetic permutations of the SIGABA cipher and control rotors, in forward orientation and at position A.

rotor	input 1	2	3	4	5	6	7	8	9	10
1	8	6	10	2	5	9	3	7	4	1
2	4	9	2	1	6	10	3	8	7	5
3	5	1	9	7	2	6	4	3	10	8
4	4	10	9	1	6	3	7	2	8	5
5	7	5	10	8	2	4	6	3	9	1

Table 179.2: Wirings of the SIGABA index rotors. Permutations listed are for rotors in position 0.

inputs	output
0 0	0
0 1	1
1 0	1
1 1	1

Table 179.3: The logical OR *operation. The output of* OR *is on (*1*) if either input is on, and off (*0*) if both inputs are off.*

input line to index wheels	logical expression
1	OFF
2	B
3	C
4	D OR E
5	F OR G OR H
6	I OR J OR K
7	L OR M OR N OR O
8	P OR Q OR R OR S OR T
9	U OR V OR W OR X OR Y OR Z
10	A

Table 179.4: Logic applied to input to the index wheels in CSP-889. The first column contains the number of the control line entering the index wheels. The second column is the logical expression applied to the output of the control wheels.

control line for stepping cipher wheels	logical expression
1	(1) OR (10)
2	(8) OR (9)
3	(6) OR (7)
4	(4) OR (5)
5	(2) OR (3)

Table 179.5: Logic applied to output from the index wheels. The first column contains the number of the control line determining the stepping of a cipher wheel. The second column is the logical expression applied to the outputs of the index wheels, where "(n)" indicates the signal on line n.

Some attention should be spent explaining how the cipher-wheel settings are derived from a five-letter message key. First, the operator sets all of the large rotors to the letter O. Start with the control wheel in slot 1 and advance it one letter (if in forward orientation, then one earlier letter alphabetically; if in reverse orientation, then one letter later alphabetically). Then send the signal on the four lines (F, G, H, I) through the control wheels (in reverse direction), logic circuits, and index rotors and allow them to advance the cipher wheels as if the machine were in normal operation enciphering a text. Continue with this procedure until the control wheel in slot 1 is set to the first letter of the message key. Note that if that letter is O, then it must take 26 steps and go all the way around. Now advance the control wheel in slot 2 until it shows the second letter of the message key, and advance the cipher

wheels after each step. Do the same for the remaining three wheels. Any time the target is O, the wheel must go all the way around. When the procedure is finished, the letters showing on the cipher wheels are the second half of the message key. Suppose we start with the example daily key:

day	control	cipher	index
today	4 1R 9 0 3R	2 6R 5R 7 8	13 26 34 42 59

Choose five random letters as the first half of the message key: WICSO. Remember that when one of them is O, that wheel must go all the way around. With the given internal settings, the procedure gives us PUJRC as the cipher-wheel settings.

Now we can attempt an example of encryption using the daily key above and message key WISCO. We already know that the initial cipher-wheel settings are PUJRC. Start with a plaintext:

THIS MESSAGE IS ENCRYPTED WITH SIGABA

The first step is to replace all Zs with X (there are none), then all spaces with Z:

THISZMESSAGEZISZENCRYPTEDZWITHZSIGABA

To encipher the first letter, let us find the alphabetic permutation for the cipher-wheel bank. We know what rotations are, from Table 20.1. The reflection that we need has this permutation:

$$X = (\mathtt{AZYXWVUTSRQPONMLKJIHGFEDCB})$$

This is b_{25} in Table 54.1, and is the same as $R_1 \circ z$ or $z \circ R_{25}$, where, as you recall, z is the reversal of the alphabet. A signal first enters the cipher-wheel bank into rotor 2 which is in forward orientation and set to P (=15). The effect of this rotor is $R_{15}^{-1} \circ W_2 \circ R_{15}$ (we have kept the notation of "W" for "wheel"). The signal then passes through rotor 6, which is in reverse orientation and set to U; its action is $R_{20} \circ X \circ W_6^{-1} \circ X \circ R_{20}^{-1}$ (notice where the inverses of rotations appear, and that X is its own inverse). Considering all five rotors in the cipher bank, the action of the entire bank is (where we have decided to drop the symbol for composition) (remember that things work from right to left in operator notation)

$$\begin{aligned} P_{\text{cipher}} &= (R_2^{-1}\, W_8\, R_2)\,(R_{17}^{-1}\, W_7\, R_{17})\,(R_9\, X\, W_5^{-1}\, X\, R_9^{-1})\,(R_{20}\, X\, W_6^{-1}\, X\, R_{20}^{-1})\,(R_{15}^{-1}\, W_2\, R_{15}) \\ &= R_{24}\, W_8\, R_{11}\, W_7\, R_1\, z\, W_5^{-1}\, z\, R_{11}\, z\, W_6^{-1}\, z\, R_{16}\, W_2\, R_{15} \\ &= (\mathtt{CETVLDQWOUZFYIHPMRJABNGKSX}) \end{aligned}$$

The first plaintext letter is therefore enciphered to A. We now have to find the stepping of the cipher wheels. To do so, we find the permutation for the control-wheel bank. We apply the same technique as above to find it when the wheels are set to WICSO.

$$\begin{aligned} P_{\text{control}} &= (R_{14}\, X\, W_3^{-1}\, X\, R_{14}^{-1})\,(R_{18}^{-1}\, W_0\, R_{18})\,(R_2^{-1}\, W_9\, R_2)\,(R_8\, X\, W_1^{-1}\, X\, R_8^{-1})\,(R_{22}^{-1}\, W_4\, R_{22}) \\ &= (\mathtt{CPKVFDZLIBOEQAJHSTXYGNWMRU}) \end{aligned}$$

Remember that the signals pass through the control bank in reverse direction, so we need the inverse of the permutation:

$$P_{\text{control}}^{-1} = (\texttt{NJAFLEUPIOCHXVKBMYQRZDWSTG})$$

The inputs on lines `F`, `G`, `H`, and `I` are mapped to `E`, `U`, `P`, and `I`. From Table 179.4, we see that the active inputs to the index-rotor bank are on lines 4, 6, 8, and 9. We now have to play the same game with the index-rotor bank. With our key, the overall permutation of the index rotors is

$$\begin{aligned} P_{\text{index}} &= (R_9^{-1}\, I_5\, R_9)\,(R_2^{-1}\, I_4\, R_2)\,(R_4^{-1}\, I_3\, R_4)\,(R_6^{-1}\, I_2\, R_6)\,(R_3^{-1}\, I_1\, R_3) \\ &= R_1\, I_5\, R_7\, I_4\, R_8\, I_3\, R_8\, I_2\, R_3\, I_1\, R_3 \\ &= (1\ \ 4\ \ 7\ \ 8\ \ 9\ \ 5\ \ 10\ \ 3\ \ 6\ \ 2) \end{aligned}$$

where now the rotations are of ten items and I_n represents the nth index rotor's permutation, from Table 179.2. We see that the inputs (4, 6, 8, 9) are mapped to outputs on lines 8, 5, 3, and 6. From the logic in table 179.5, we find that the cipher wheels that step are in slots 2, 3, 4, and 5. Because the wheels in slots 2 and 3 are in reverse orientation, they advance to the next letter alphabetically; so `U` goes to `V` and `J` to `K`. The wheels in slots 4 and 5 are in forward orientation, so they advance to the previous letter alphabetically; `R` to `Q` and `C` to `B`. The cipher wheels are now set to `PVKQB` for the encipherment of the second letter. The center control wheel advances, so the control bank is now set to `WIBSO`. We can find the new permutation of the cipher-wheel bank:

$$\begin{aligned} P_{\text{cipher}} &= (R_1^{-1}\, W_8\, R_1)\,(R_{16}^{-1}\, W_7\, R_{16})\,(R_{10}\, X\, W_5^{-1}\, X\, R_{10}^{-1})\,(R_{21}\, X\, W_6^{-1}\, X\, R_{21}^{-1})\,(R_{15}^{-1}\, W_2\, R_{15}) \\ &= (\texttt{KCAIXJTBFLRMUPDHZYNWSGVEQO}) \end{aligned}$$

The second letter of the plaintext, `H`, is enciphered to `B`. If we continue in this manner, we find that the full ciphertext is

`ABOTTYSYZJAQWNETZPSCCUQMUFSPJRUJTDUEI`

A later version of the SIGABA machine, known as *CSP-2900*, is similar to the CSP-889, with these differences:

- The set of active lines into the control wheels is expanded to six: `D`, `E`, `F`, `G`, `H`, `I`.
- The logic applied to the output of the control wheels to form the input to the index wheels is modified to be what is listed in Table 179.6.
- The cipher wheels in slots 2 and 4 rotate in the opposite direction to those in slots 1, 3, and 5, which rotate as in CSP-889.

input line to index wheels	logical expression
1	U OR V
2	B
3	C
4	D OR E
5	F OR G OR H
6	I OR J OR K
7	L OR M OR N OR O
8	S OR T
9	W OR X OR Y OR Z
10	A

Table 179.6: Logic applied to input to the index wheels in CSP-2900. The first column contains the number of the control line entering the index wheels. The second column is the logical expression applied to the output of the control wheels.

Reading and references

Mark Stamp and Richard M. Low, *Applied Cryptanalysis: Breaking Ciphers in the Real World*, Hoboken: Wiley, 2007, section 2.4.

George Lasry, "Functional Description of SIGABA." This document can be found in the supplementary materials for some challenges on MysteryTwister, in this zip file: https://mysterytwister.org/media/challenges/add/mtc3-kopal-25-SIGABA-CSP889-01-add.zip

George Lasry, "A practical meet-in-the-middle attack on SIGABA," Proceedings of the 2nd International Conference on Historical Cryptology, HistoCrypt 2019, Linköping University Electronic Press, 2019, 41-49, https://ep.liu.se/ecp/158/005/ecp19158005.pdf

> Note that the above paper has an error in the table of index input logic. The following paper has the correct table.

George Lasry, "Cracking SIGABA in less than 24 hours on a consumer PC," *Cryptologia* 47:1 (2023) 1-37, DOI: 10.1080/01611194.2021.1989522

Timothy J. Mucklow, The SIGABA/ECM II Cipher Machine: "A Beautiful Idea," Center for Cryptologic History, [U.S.] National Security Agency, 2015, https://www.nsa.gov/portals/75/documents/about/cryptologic-heritage/historical-figures-publications/publications/technology/The_SIGABA_ECM_Cipher_Machine_A_Beautiful_Idea3.pdf

Operating Instructions for ECM Mark 2 and CCM Mark 1, transcribed at https://maritime.org/doc/crypto/ecm/index.php

Crypto-operating Instructions for ASAM 1, transcribed at https://maritime.org/tech/ecminst.php

Programming tasks

1. Create a function that finds the alphabetic permutation of a reversed rotor. Tabulate the reversals of the ten rotors in Table 179.1.

2. Implement enough of the machine in Python so that you can find the initial cipher-wheel settings when given a five-letter message key.

3. Add the remaining pieces to simulate the entire SIGABA CSP-889. Inputs should be the daily and message keys and, of course, the plaintext. Allow for the message key to have five or ten letters.

4. Now simulate the CSP-2900.

Exercises

1. Find the cipher-wheel settings for these five-letter message keys for the CSP-889 version of SIGABA, given today's daily key (see above in the text of this unit for the key).

   a. `SMFCJ`

   b. `OTCDP`

   c. `XGDRF`

2. Encipher this text with version CSP-889 and today's daily key and message key `MSRZN IVLDO`. Do not include punctuation, and remember to replace spaces with `Z` in the plaintext, if your program does not already do so.

   Science shall extend the bounds of knowledge and power, adding unimaginable strength to the hands of men, opening innumerable resources in the earth and revealing new secrets and harmonies in the skies.

   (from "The Glory of Peace" by Charles Sumner, in *Standard Selections*, edited by Robert I. Fulton, Edwin P. Trueblood, and Thomas C. Trueblood)

3. Decipher this text with version CSP-889 and today's daily key and message key `NRJCK`. Remember that `Z` in the plaintext is space.

```
QMNTJQSULAPPIQHXDDKDBVFALCMOAYVXGJKRYGHOGVJHKMLUCLEAHBBKNTFWBQNI
QVHILLUHFSEBSALAGJIJEZLSHWXSOKAMWSAVEMXQRGHXMRKATEUDUTHAYILMYJBG
JNYCFKOWVKRGZTKJHTOEYEDPNAVLPYCKBSZROQZNXNZZCMNPSRKDLAYORMERYYKE
YDFCYUHQJJSXIPJDYLXOJXYXTTSCADSRATQNVXUSUGBQIVIVCMNFQJAETNGXMVHT
TJVCUKAUXEMGWYESDFKGXSKKLCXLYHLQXQLGTBUAFKTJZTLQDKXHEHOZVRUMUSJD
MMMIXOABZXXQDPZNWONRILGWLDOMWZOKEPRJUFEROUCIGNQBGHGKBCZKEIKINDCU
CQZZYXICWOBKENPNXAONRJKHKXECLQCUGLOZMCIQOSJLXCLBGAIWPEBJFTZALQUZ
XUJPNYTKQPBKDUUGLITRBIJYMZWZERSNQXMIBEDYJFFVMRZJDWSJTYGJJMZBQAIZ
ZWMAJZCZTQCZHTVASCFEXXOMQLCJDQOCEPIGTLGAIXSSVOKWTWPDDDZHTMVYSXIP
KXMGGEQDBOLZHBEYNFDEHGLPLQLNHREYREOFJTZQTYZOCHMNPBQFF
```

4. Decipher with version CSP-2900, the following daily key, and message key `ZIHMF`.

day	control	cipher	index
tomorrow	9R 8R 2 5 3R	0R 7 4R 1 6	29 30 18 58 44

```
RCWWMQXIVNASRUDGPRFKXLOYMUFUKDHHWEZBDMDVKARBUNPBBMEIMPEZPVUTILMO
NIBVQSAJPQPJHUKUYSDXXUVBFRULWVDCADCRDXFJOYHTZLKRQSNSJGBCVBRUEWKV
RXWEQHBDLFCEJQJSIEAUQCWRUZNRFOVMIUVFETDSSRLHCDAHUIFDMYVHUPPEUGHH
ERDCITZSSVRQOSHAMPENKJKDAHZRLXHTCYGCHONEFTQMDNQGEYGHKQJEHKXWSCWQ
ELMBVPPPWLEFCWRSEHOEZGXFBYRQWIDEDONXLMECOTYFHEMNWLAWSDDGSKEJKSUR
IYDPJKLXKUKBCFIDSRVJJYVFDMEQDYDMZGZJVAEJVRSOTLODMQYKPSYHKOPPVNQI
KUSRTKOPSLOMWKCREVIIJMGOQPZOBTJFKSBFZMUKSOUGFMODAWRREWANHPPSMRWY
YAHRAUVSGOBABIHTYVSWFVIMCEYEDWVQRURYQZBIQVZKPIOINSMKABDDHFNZBVTA
NVSDZYOILCBMDRIELW
```

Unit 180
Фиалка

Fialka, also known as *M-125*, or *Фиалка* in Russian, is a cipher machine used by the Soviet Union during the Cold War. It had ten rotors, irregular stepping, a reflector with special properties, and a punchcard reader for implementing an alphabetic permutation on the way into and out of the rotor assembly. Unlike Enigma and SIGABA, the Fialka's rotors permute the Russian alphabet, or at least most of it.

The Russian alphabet (a version of Cyrillic) is

АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ

Take note that Ы looks like two, but is a single letter. Russian cryptographers typically treat Е and Ё as the same letter; some Russian civilians do as well. The full alphabet has thirty-three letters, but the rotors and punchcard of Fialka are designed to handle only thirty. This is the modified alphabet used by the machine:

АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЫЬЮЯЙ

The letter Й ("short i" or "и краткое") has been moved to the end, probably so that the ten rotors of the machine could be labeled А-К without confusion. Letter Ё is understandably missing, but so are Ъ and Э. After one of the many Russian revolutions, there was an orthographic revision of the language, and Ъ ("hard sign" or "твёрдый знак" or "turdy snack" as one student may have called it) was dropped at the ends of words. It is now a rather rare letter, and I cannot even think of a word that contains it right now. Letter Э is also rare, and almost always appears in words derived from foreign words; a suitable replacement is Е.

Versions of the Fialka machine were used in different regions of the U.S.S.R. and countries in the Warsaw Pact, and we don't have complete information about every variation. They differed in their rotor wirings and in how keys on their keyboards could be used for Latin letters and symbols. To distinguish a set of rotors from another, a designation such as "3K" is stamped on them, along with the letter denoting which rotor we have from the set (А through К). In Tables 180.2, 180.3, 180.4, and 180.5 are the wirings for series 68, used in Russia; 3K, used in Poland; 5K, used in Hungary, and 6K, used in Czechoslovakia. For variants of keyboard mappings to Latin letters and symbols in Poland, East Germany, and Czechoslovakia, see Table 180.1. At the time that this book is being written, there is no information about other versions of rotors or keyboard layouts.

Figure 180.1: Fialka cipher machine. In the photo on the right, the cover is open and the rotors are visible. Photos from Tom Perera (EnigmaMuseum.com).

region	character set	Cyrillic key АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЫЬЮЯЙ
Poland	Latin: symbol:	FADULT5PBRKVZJGHCNE7QWXIOSM8Y2 Ą%-6)4 9?3(=5+ĘŁ:,2 01/78Ž.°'
East Germany	Latin: symbol:	F8DULT5PBRKVZ4GHCNEA327IOSM96J V:W6)4 9/3(X5JÄÜY,2 01F78Q.-Ö*
Czechoslovakia	Latin: symbol:	FADULT5PBRKVZJGHCNE7QWXIOSM8Y2 ŘÄČ6Ě4 9"3Ö=5ÜŽ+:,2 01/78Š.%-

Table 180.1: Substitutions for Latin letters and symbols used by Fialka. Ж *and* Ф *serve as symbol shift and letter shift, respectively. Non-Cyrillic letters were laid differently onto the keyboards in different regions of the U.S.S.R. In East Germany, the usual* Ä=AE, Ö=OE, Ü=UE, *and* ß=SS *or* SZ *can also be used.*

The Fialka cipher machine is very much like what an Enigma might have become if it had evolved under the control of the workers who have seized control of the means of production in order to lead the world into a brighter future. It has a keyboard, rotors, and reflector, but instead of the plugboard it has a punchcard reader for greater efficiency. The number of rotors has been increased to ten, and they have thirty contacts on each side. They are installed in any order into the machine beginning at the left end near the reflector (slot 1) and continuing to the end near the entry plate (slot 10). The reflector has been modified to avoid the weakness of Enigma that letters could not be enciphered to themselves. And the punchcard permutes the entire set of inputs, unlike Enigma's limit of the use of ten transpositions. We will now spend some time looking more closely at these components before we put the pieces together and see how a simulation might work.

The keyboard has thirty buttons for letters, of course, for the thirty letters listed above. But it also has a space bar, and it maps space to Й. Unfortunately, this means that yet another letter is unavailable. When Й is encountered in a plaintext, the output printer writes a space. The keyboard for machines used in various regions of the Soviet world had different substitutes for Latin letters. Those that are known are listed in Table 180.5. Between the keyboard and the punchcard reader the wiring of the machine carries a permutation which we call *S*:

$$S = (\text{СЩЙОЫХЕУАПЯФГЮШБЦЧТМЖДЬЗКИРНЛВ}) \tag{180.1}$$

The punchcard reader comes next. It holds a paper card with a 30×30 grid that unfortunately is not printed. Nevertheless, there can be exactly one hole punched in each row and exactly one in each column of this grid. Such a thing can be thought of like a 30×30 matrix with a single 1 in each row and column and 0 everywhere else (see Unit 123 for our introduction to matrices). Such a matrix, when multiplied by a column vector, performs a permutation on it. The punchcard reader takes the matrix defined by the holes in the card and applies the permutation to the thirty data lines coming to it from the keyboard (after passing through *S*). In Figure 180.2 is a reasonable facsimile of a Fialka punchcard. To read its permutation, look at the perforation (row of holes) near the top. If you discard the leftmost and rightmost holes of the perforation, then the remaining thirty define the columns of the grid. As you read from left to right, the letter that goes into writing down the permutation is which row is punched. In Figure 180.2, the first column has the fifth highest hole, so must be Д. The second column has the twenty-fourth highest hole, so must be Ш, etc. The permutation on this card is

$$P = (\text{ДШНЯЖЧВЙИБАЮХПЩТКГУСЫМЬЗФЕРОЛЦ})$$

As the signal leaves the punchcard reader and heads toward the rotors, it encounters another permutation built into the machine's wiring which we call *T*:

$$T = (\text{ЮОФШБРАКХЛСНУЙДЕЗПЧЩЬТВЯЮМЦЖИГ}) \tag{180.2}$$

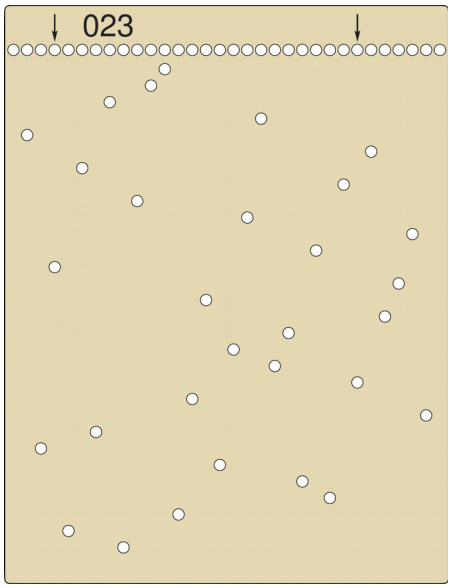


Figure 180.2: Facsimile of a punchcard for use in Fialka. The number indicates the day of the year in which it should be used. The arrows indicate the holes in the perforation through which alignment pins will protrude (pins are not placed symmetrically so that the card would not be placed in the machine upside-down). The permutation that this card represents has been randomly generated and should not be used in real-world communication.

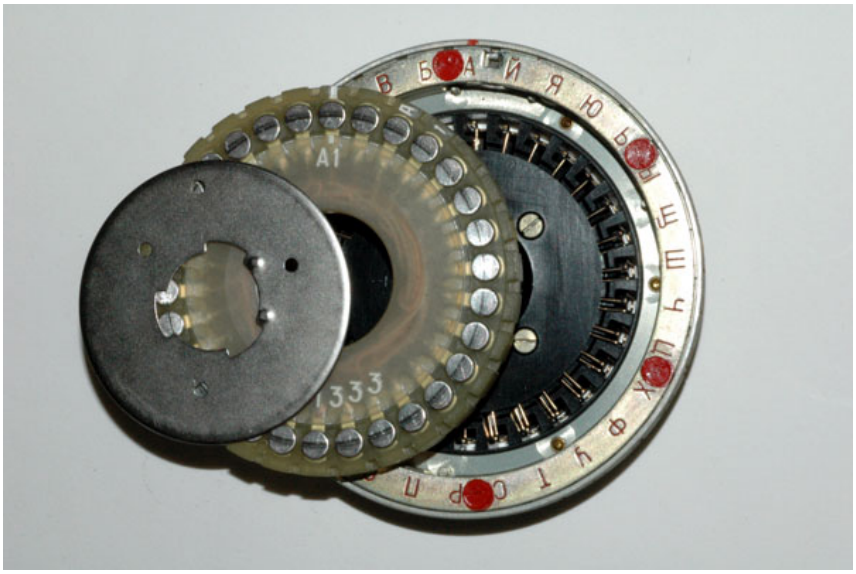


Figure 180.3: Disassembled configurable rotor from a Fialka machine. The red mark on the rim at the top is the indicator mark for the ring setting. The wiring core is A *and side 1 is showing. The white mark at the top of the wiring core is the indicator for its placement in the rotor frame. Photo from Tom Perera (EnigmaMuseum.com).*

Rotors have thirty contacts on each side, and thirty Russian letters on the rims. They come in two varieties: adjustable/configurable and non-adjustable/non-configurable. Non-configurable rotors cannot be changed. They can be installed into different slots in the machine and rotated to different positions (i.e. so that they display a different letter to the operator), but they cannot be installed in reverse orientation and their alphabet ring cannot be shifted, etc. The permutations in Tables 180.1, 180.2, 180.3, and 180.4 are for the non-adjustable rotors in series 68, 3K, 5K, and 6K. Each series has ten rotors, designated with one of the Russian letters А, Б, В, Г, Д, Е, Ж, З, И, К. Each has thirty contacts on each side, so effects a permutation of the subset of the Russian alphabet with thirty letters given earlier. The permutations in the tables are for a signal entering on the righ-hand side of a rotor and exiting on the left-hand side. The permutation when the signal returns and passes left to right is the inverse. For dealing with the rotations of rotors with thirty positions, we need a new set of rotation permutations. The first is

$$R_1 = (\text{БВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЫЬЮЯЙА})$$

The rest can be generated from it. As we have seen before, the permutation of a rotor W that has been turned by n steps is its unrotated permutation W sandwiched between two rotations:

$$R_{-n}\, W\, R_n \tag{180.3}$$

The configurable rotors have three components: a rotor frame, an alphabet ring, and a wiring-maze core. The frame has the thirty letters on its side, so that the alphabet ring and wiring core can each be aligned to one of those letters. In addition, the core can be flipped over so that one of two sides can be facing outward/leftward (sides “1” and “2”) (the frame cannot be installed into the machine in reverse orientation). Recall that in Unit 179 we saw that installing a rotor W into SIGABA in reverse orientation gave us $X\, W^{-1}\, X$, where X is a reflection about the axis that runs through letters A and N. With Fialka, when we place a wiring core into a rotor frame so that side 2 is facing outward, we do something similar, and we need the reflection about the axis that passes through (Russian letters) А and Р. So our new X is

$$X = (\text{АЙЯЮЬЫЩШЧЦХФУТСРПОНМЛКИЗЖЕДГВБ}) \tag{180.4}$$

Things are a little more complicated now, since the core can be rotated when placed in the frame as well as flipped over. Suppose we take a rotor in its base configuration $W_{\text{base}} = W(0, 0, 1)$, which is the same as the non-adjustable version, and which has the ring setting at А=0, the wiring-core indicator mark aligned to А=0, and the core with side 1 facing outward. The effective new rotor with ring setting r and core alignment m is

$$W(r, m, 1) = R_r^{-1}\, R_m\, W(0, 0, 1)\, R_m^{-1}\, R_r \tag{180.5}$$

When the core is flipped, the new rotor is

$$W(r, m, 2) = R_r^{-1}\, R_m\, X\, W(0, 0, 1)^{-1}\, X\, R_m^{-1}\, R_r \tag{180.6}$$

rotor	input АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЫЬЮЯЙ
А	ГИЖЬУЦШЙВДБТЛАФЧЫОЯЮКПЕХЩРЗНСМ
Б	ПЧЯДЗТЙКЮОЫЛХГФНЩМШВБСИЦАУЖРЕЬ
В	ДЩОГЯАЬНЦРЖЛФПЙБУЫСЧВКШЗМТЕИЮХ
Г	ФЧТХНПМАУЫСЙЖЩОКЯЗИГЦДЬРБЕЛЮШВ
Д	ДФЗКГОЧЬЙШИНЕЫМАЛЩБЖСВРТЮХЯПУЦ
Е	ЬТБДРКУЯЮМИГЫХЩАЛЕЖФСПНЙЦШОЧЗВ
Ж	ЫЙЯВРЕОЛКШБДЦУМЩХФЬЗНГИЮАПЖТЧС
З	ЗЫРМШЖЦКХЬПИГОТЩДНЯЕБФЛСАЧУВЮЙ
И	ТСДМБЮЫУРЧФВЯШПКЦХЖИЗГЬОАЩЛНЙЕ
К	КЗШЬБДЖЛГЕМЙЧТЫПЯНВЩЮРАЦОИХФУС

Table 180.2: Rotor wirings of Fialka for rotors in series 68 used in Russia. These wirings are for non-adjustable rotors and for a signal entering from the left. For the adjustable rotors, these wirings match the configuration in which the wiring maze and rotor frame have the same letter designation, А *on the ring is aligned with* А *on the rotor frame, the indicator mark on the wiring maze is aligned with* А *and side* 1 *faces outward.*

rotor	input АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЫЬЮЯЙ
А	ЧЦВЖГЗРЕКФПСШИЙМЩЛАЮЬДЯЫБТХОНУ
Б	ВШФБЕХЫЖТГСЧПУКЙНЮЯЛИЩАОЦЗЬДМР
В	ФДЖПХЬГАЦСЧНЙЕЫКРОУТЯШВМИЛБЮЗЩ
Г	РХЮЛЬВПМШЙИСГФЩЗАЯУТОКДЧЫЖЕЦБН
Д	ТПАЦУРЯЗСГВОЕЙЧДЫНЩКМХЬФЖЛШИБЮ
Е	ИОНФШЗБЕДУЛЮЙВТПЖЩРАМЧЬЯСКХГЦЫ
Ж	ЖИДЫЕГУВЗЮЦМХШЧКНАРЯБЩЬПТЛОСЙФ
З	ЯЬПНЗБЩМЕЧИТШАОХСКВЛЦЖРГУЫДЙЮФ
И	ДУБЬФЫЖЛРТВНГЧЮХЕШЯЙПСИМЗЦЩКАО
К	ФШЗЩУАСДПЬИМЦКТВРЙГОЖЧЛБЯЫЮХЕН

Table 180.3: Rotor wirings of Fialka for rotors in series 3K used in Poland. These wirings are for non-adjustable rotors and for a signal entering from the left. For the adjustable rotors, these wirings match the configuration in which the wiring maze and rotor frame have the same letter designation, А *on the ring is aligned with* А *on the rotor frame, the indicator mark on the wiring maze is aligned with* А *and side* 1 *faces outward.*

rotor	input АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЫЬЮЯЙ
А	НЛКПАФШБГИЬСЫВМЗДРУЦЧЮОЩТЕЙЯЖХ
Б	ЕУЦАХНЮЫЙКИЖМЧСШЩЗФРГБПДТЯВЬЛО
В	ЩПЮЖУТОКШЕРИГФВЬЦДЗЯАМХЧСЫНБЙЛ
Г	ЧВШНГЫТПЛЕЯЦЩЖИХФБОСУЙЗКМАРЮДЬ
Д	РЦИБЕДАМГСОХТПЮЬЗЛЯВЩКУЫЧНШФЖЙ
Е	КЕЗФЧЬЫГЯТСХШЛОЮАДНМЦВРУЖИЩЙБП
Ж	ХЙЫВРЮФИЩАЖСКЕЬЧМБЦШОЛЗГУДНПЯТ
З	ШСЛФЙЯУКЖХЕТБАДНЬМПЩГЧОЮИВЫРЗЦ
И	СИБГФЬРКХЖПЯЫЩТОЛЙЦМЮЧНЗАЕДВШУ
К	БМДЗНЯКУЩШРЖЦЬЫЮЙЕГИЛТСЧАВОФПХ

Table 180.4: Rotor wirings of Fialka for rotors in series 5K used in Hungary. These wirings are for non-adjustable rotors and for a signal entering from the left. For the adjustable rotors, these wirings match the configuration in which the wiring maze and rotor frame have the same letter designation, А *on the ring is aligned with* А *on the rotor frame, the indicator mark on the wiring maze is aligned with* А *and side* 1 *faces outward.*

rotor	input АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЫЬЮЯЙ
А	НЦЗТФМЮГПЬВДРОЧЫАЩСЛЙКШЖЕХЯБИУ
Б	ФЗДПГЮХАШНЯМОЧЩЖИЙЬВЛТСУЦКБЫЕР
В	ЯЛГЦШРТБЧВСЗФДЮМПЫЙЖХУНКЬЩИАОЕ
Г	ГМУЯШЧЖЙПАФОТБРЬКЩСЮЕХЛЗЦДИВЫН
Д	ТБПЖФЮЗНЧМУЬГШКОЛЕЙВСЫЦАЯЩРХДИ
Е	РГОШЧУЙВАЗЬНИДЯКПЫЦЖЩСФЛБЕХЮТМ
Ж	ЫЧЖДНЗШЙЯФЦИМКЩРВХУТГАЮЬЕБПСЛО
З	РЦОЙШПСФГЖЬМЕНЩХАДЫЗЛЧЯЮВТКУБИ
И	МАСЯЕГЖЛПВХЩИЫЙНЦФКШЬОЮЧБДУТРЗ
К	ИХЗРЩДФЦГЬОУАПЙБКСЖШМТЯВЧЕНЮЫЛ

Table 180.5: Rotor wirings of Fialka for rotors in series 6K used in Czechoslovakia. These wirings are for non-adjustable rotors and for a signal entering from the left. For the adjustable rotors, these wirings match the configuration in which the wiring maze and rotor frame have the same letter designation, А *on the ring is aligned with* А *on the rotor frame, the indicator mark on the wiring maze is aligned with* А *and side* 1 *faces outward.*

Stepping of the rotors occurs after each letter is enciphered and is irregular. Rotors in slots 1, 3, 5, 7, and 9 (counting from left to right) are ganged together, and their motions are interrelated. They only rotate in the forward direction, i.e., so that the letter describing its positions changes to the following one in the thirty-letter subset of the alphabet (with wrap-around). The rotor in slot 9 steps every time. The rotors in slots 1, 3, 5, and 7 cannot rotate if an advancement-blocking pin is present on the rotor in slot 9 at a position 20 further than its current position. If the pin is not present, then the rotor in slot 7 steps. The rotors in slots 1, 3, and 5 cannot rotate if a pin is present on the rotor in slot 7 at a position 20 further than its current position. Without that pin, the rotor in slot 5 steps. The same behavior continues, so we see that a rotor in an odd-numbered slot cannot advance if it is blocked by a pin on a rotor in any other odd-numbered slot to its right. Rotors in slots 2, 4, 6, 8, and 10 work together and rotate in the reverse direction, i.e., so that the letter indicating its current position changes to the one preceding it in the alphabet. The rotor in slot 2 is the one that steps every time. Each of the others is blocked from stepping if there is a pin in a position 17 further than the current position of any rotor in an even-numbered slot to its left. The positions of the stepping-inhibiting pins for series 68, 3K, 5K, and 6K are in Tables 180.6, 180.7, 180,8, and 180.9.

rotor	pin locations АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЫЬЮЯЙ
А	×× × × ×× ××× ×× ××× ×××
Б	× ×××× ×× ××××× ×××× ××××× ××
В	× × × × × × ××× ××
Г	× ×××× ×× ××××× × ××× × ××
Д	×× ×× × × ×
Е	× × ×××× × ××××× ×× ×××
Ж	×××× × × ×× ×× ×× ×
З	× × × × ×× ×
И	× ×××× × × ×× ××
К	×× ××× × ×× × × ×

Table 180.6: Locations of stepping-inhibiting pins on the rotors in series 68.

rotor	pin locations АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЫЬЮЯЙ
А	× × ×× × × ×× ×× × × ×
Б	× × × × × ×××××× ×× × ×××
В	× × × × × × ×
Г	× ××××× ×× × ×××× ×××××× ××× ×
Д	× × ×× × × ×
Е	× × × ×××× × ×××× ×
Ж	× ×××××× ×××××× × ×××× ×××× ×
З	× ××× × × × ×
И	×××× × × ×× × × ×××××× ×
К	× × × ××× ×× ×× ×××

Table 180.7: Locations of stepping-inhibiting pins on the rotors in series 3K.

rotor	pin locations АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЫЬЮЯЙ
А	×××× × ×× ×× × ×
Б	× × × × × × ×
В	× × × × × ××
Г	× × × × × × ×
Д	× × × × ××× × × × ×
Е	×××× × × × × × ××
Ж	× ×× ××× × × ×× × ××
З	×× × × ×××× ×× ×× × ×××× ××
И	×× ×× ×××× × × ×××× ×××
К	×× × ×××××× ××× ××× ×× ××

Table 180.8: Locations of stepping-inhibiting pins on the rotors in series 5K.

rotor	pin locations АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЫЬЮЯЙ
А	××× × × × ×× ×× ×× ×××× × ××
Б	× × × × × × × × ×× × ×
В	× × × × × × ×
Г	× × ××× ××××× ×××× × ××× ×
Д	× × × × × ××
Е	× ××× ××× ××××× ××× × ×
Ж	××× ××× ××× × ×××××× ×
З	×× ×× × × ×××××
И	× ×× × × × × ×
К	××× ×× ××× ××××× ××× ××× ×× ××

Table 180.9: Locations of stepping-inhibiting pins on the rotors in series 6K.

Unlike Enigma, Fialka has only one reflector. It is permanently installed in the machine; it cannot be removed; it cannot be rotated. And to improve on the Enigma, it has some innovations. While most of the connections are wired in pairs, like in any other self-respecting reflector, a signal on connector 12 (counting from zero) indicates that a plaintext letter should be enciphered to itself. This eliminates the weakness that Enigma had in its inability to do this. In Fialka, since one connector has this property, any given plaintext letter has a one in thirty probabililty of being its own ciphertext letter; this is just what would be expected from a process that randomly assigns ciphertext letters. Since it is impossible to send a signal back again on the same connector, some circuitry in the reflector instead sends a signal back to the keyboard to tell it to bypass the entire encryption process and send the plaintext letter to the output. Nevertheless, when we are creating a simulation of the Fialka, we can ignore this physical impossibility and simply allow the reflector's permutation to map Н to Н. But now we have another problem: with one connection used for itself, there is an odd number remaining. We cannot map another connection to itself, because that would spoil the appearance of randomness. To solve this problem, the makers of the Fialka introduced another bit of circuitry that performs a 3-cycle on connections 15, 17, and 23. This breaks the self-reciprocity of the reflector, so some more circuitry must be added to reverse this 3-cycle when the machine is in decryption mode. Here is the permutation of the reflector during encryption. We call it *U* as we did for Enigma. Above it we write the rotor alphabet so that we can see the 3-cycle more easily.

$$\begin{array}{l} \quad\;\; \text{АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЫЬЮЯЙ} \\ U \;=\; (\,\text{ЧЕФЮОБМСЦЛКЖНДЯТЗШЬВЩИАРХЙУГПЫ}\,) \end{array} \tag{180.7}$$

Notice the 1-cycle (Н) and the 3-cycle (РТШ). All other letters are in 2-cycles.

We are finally ready to put all of the pieces together. In Figure 180.4 is an overview of the Fialka cipher machine. When a letter is processed, these things happen:

1. A key for the letter is pressed on the keyboard. If the key is the space bar and the machine is in encryption mode, then the character is changed to Й by the keyboard. The signal leaves the keyboard.
2. The signal passes through permutation S (Equation 180.1).
3. The signal passes through permutation P of the punchcard reader.
4. The signal passes through permutation T (Equation 180.2).
5. At the entry plate to the rotor assembly, there is a rotation by 3 because the letters indicating the positions of the rotors are not centered in the machine.
6. The signal passes through the ten rotors in reverse order (right to left, slot 10 to slot 1). Despite the reversal of rotor order, the permutation of each is in its forward direction.
7. Between the rotor assembly and the reflector is a rotation by −3. See number (5) above.
8. The signal passes through permutation U (Equation 180.7) if the machine is in encryption mode. If the machine is in decryption mode, the permutation is U^{-1}.
9. A rotation by 3 when entering the rotor assembly.
10. The signal passes through the ten rotors in forward order (left to right, slot 1 to slot 10). The permutation of each is the inverse of the permutation in step (6).
11. Rotation by −3 at the entry plate.
12. Permutation T^{-1}.
13. Permutation P^{-1}.
14. Permutation S^{-1}.
15. If the machine is in decryption mode and the resulting signal is Й, then the printer outputs a space. Otherwise, it prints the letter that it receives.

The overall permutation effected by the machine during encryption is (ignoring the bits about space bar and Й)

$$F = S^{-1}\, P^{-1}\, T^{-1}\, R_{27} \prod W_i^{-1}\, R_3\, U\, R_{27} \prod W_i\, R_3\, T\, P\, S \tag{180.8}$$

Remember that operators act on what comes to them from the right and so this product of operators works from right to left. We have used $\prod W_i$ and $\prod W_i^{-1}$ as shorthand for

$$\begin{aligned} \prod \mathrm{W}_i &= \mathrm{W}_1\, W_2\, W_3\, W_4\, W_5\, W_6\, W_7\, W_8\, W_9\, W_{10} \\ \prod W_i^{-1} &= W_{10}^{-1}\, W_9^{-1}\, W_8^{-1}\, W_7^{-1}\, W_6^{-1}\, W_5^{-1}\, W_4^{-1}\, W_3^{-1}\, W_2^{-1}\, W_1^{-1} \end{aligned}$$

where W_i is the permutation of whatever rotor is in slot i in whatever position it currently is in. A configurable rotor is constructed as in Equations 180.5 and 180.6. All rotors are rotated to their current positions according to Equation 180.3.

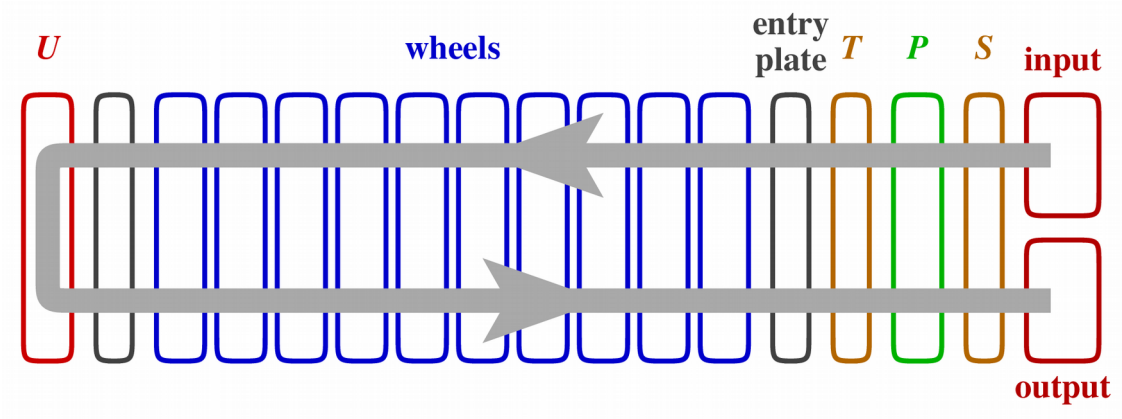


Figure 180.4: Functional overview of the Fialka cipher machine. Input from the keyboard (30 wires) passes through permutation S, the punchcard reader P, permutation T, a rotation for the entry plate, ten wheels, a rotation between the wheels and reflector, the reflector U, and back again in reverse to the output.

The internal settings of the Fialka machine form the daily key. For machines with rotors with fixed wiring, the daily key is a list of ten rotors in the order in which they are installed on the spindle, from left to right (from the reflector toward the punchcard). For machines equipped with configurable rotors, the specific contents of the daily key are five lists of ten items and the punchcard. The first item in each list describes what is placed in the first slot, etc. The lists describe

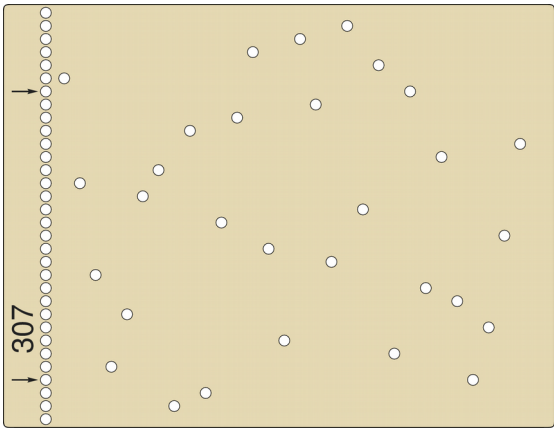
- the choice of rotor frame (Russian letter)
- the ring setting (Russian letter)
- the identity of the wiring core (Russian letter)
- whether the wiring core is installed upside-up or upside-down (digit 1 or 2)
- the position of the indexing mark on the wiring core (Russian letter)

The initial positions of the rotors serve as the message key.

For example, if we are working with a machine with non-configurable rotors from series 3K, and our key is this one, which was randomly generated:

daily key: `ЕЗКАВ ИГЖДБ`

message key: `ВЫДЮЛ ЗХСКЧ`



The permutation given by the punchcard is

P = (ЗКЬГЦПЮДЫШВТОЯЛФЕБЖЩЙИМСЧАХНРУ)

Here is a typical message that we can encrypt:

THIS MESSAGE IS ENCRYPTED WITH FIALKA

The first thing to do is rewrite the message in Russian, with Й in place of spaces.

ЕРШЫЙЬУЫЫБПУЙШЫЙУТСКЯЗЕУВЙЦШЕРЙАШБДЛБ

The machine is loaded with non-configurable rotors such that rotor Е is in the first slot, rotor З in the second, К in the third, etc. The rotors are initial set to the positions given by the message key: the first at В, the second at Ы, etc. The resulting permutation from Equation 180.8 is

УЛНЮЫЙПФСЩБЧВХЖШИТАЗОРМЦКДЯГЬЕ

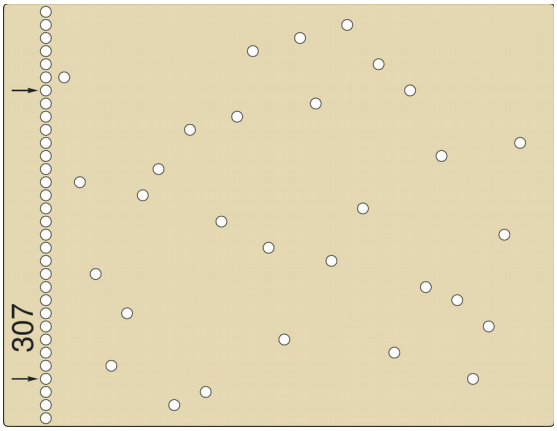
This permutation maps the plaintext letter Е to Й. Next, since there is a steeping-inhibiting pin at Н on rotor З in slot 2 (17 steps ahead of its current position, Ы) (Table 180.7), the rotors in slots 4, 6, 8, and 10 do not step at this time. There are no pins blocking the rotors in slots 7 and 5 from stepping, but those in slots 3 and 1 cannot step. So the rotor positions go from ВЫДЮЛЗХСКЧ to ВЩДЮМЗЦСЛЧ and we are ready for the second letter of the text. We continue until we have the full ciphertext:

ЙЩНШВЯХГЬЛЮЫЧЮРФУЧЙМКЗЛЕНШЕРЙГЦЫЮЗМДБ

Now let's expand the daily key for a machine with configurable rotors. The new settings were also randomly generated.

daily key: ЕЗКАВ ИГЖДБ
ШЖВЙМ ЛПХЦЕ
БЗГКИ АДЖВЕ
21121 11221
ВЧЩЫР ЯЛПБИ

message key: ВЫДЮЛ ЗХСКЧ



The first slot contains a rotor comprised of the Е frame with the Б wiring core installed with side 2 facing outward, ring setting Ш=23, and core alignment В=2. The other slots contain rotors configured according to the respective columns in the key. The permutation on the punchcard is the same as before. With this key, the same plaintext becomes this ciphertext:

МТЧЬГДХМАБФПЬЫОСФЕГЯУЮТЩОРШЦЧЛРЧЛПГИТ

Reading and references

Thomas B. Perera and David Hamer, “General Introduction: Russian Cold War Era M-125 and M-125-3MN Fialka Cipher Machines,” 2005, http://w1tp.com/enigma/mfialka.htm

Thomas B. Perera and David Hamer, “Rotor Technical Descriptions and Wiring and Stepping Data for the M-125-MN and M-125-3MN/-3MP3 Russian Fialka Cipher Machines,” 2005, http://w1tp.com/enigma/mfr.htm

Paul Reuvers and Marc Simons, Fialka M-125: Detailed description of the Russian Fialka cipher machines, version 2.0, 2009-06-22, https://www.cryptomuseum.com/pub/files/Fialka_200.pdf

If you see “36” in the table for the rotor wirings of series 5K, it should be “26”.

Crypto Museum pages on the Fialka:

main page:	https://www.cryptomuseum.com/crypto/fialka/index.htm
functional overview:	https://www.cryptomuseum.com/crypto/fialka/block.htm
rotors:	https://www.cryptomuseum.com/crypto/fialka/wheels.htm
rotor wirings:	https://www.cryptomuseum.com/crypto/fialka/wiring.htm
reflector:	https://www.cryptomuseum.com/crypto/fialka/magic.htm
punchcards:	https://www.cryptomuseum.com/crypto/fialka/card.htm

Python tips

Python version 3 does everything in eight-bit Unicode by default, so there is nothing special that you need to do (in Python 2 you needed to set the encoding of your files in a special comment). Printing to console may be a problem if you don’t have that set up right.

The Russian letters are at these Unicode points (in hexadecimal):

А-Я	U+0410 through U+042F
а-я	U+0430 through U+044F
Ё	U+0401
ё	U+0451

Of course, it may be easier to just open this document in software that allows you to copy and paste this alphabet:

АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ

Remember that Ы is one letter and that often, Е and Ё are treated as the same letter.

The string methods `.upper()` and `.lower()` work on Russian, too.

Programming tasks

1. Simulate the Fialka in Python. This can be a good exercise in using a non-English alphabet and creating files in Unicode.

2. If you are ambitious, try installing the `Pillow` package in Python and use it to create a punchcard reader that takes a graphic file and outputs the permutation. Specifications for the physical size and positions of the holes can be found in the reference by Reuvers and Simons. For information about `Pillow`, you might start here: https://pillow.readthedocs.io/en/stable

Exercises

1. Use a Fialka with series-3K non-configurable rotors and the Polish keyboard to encrypt this English text with the given key. Use the punchcard in Figure 180.2. Discard punctuation but retain spaces, which become Й before encryption. Express the ciphertext in two ways: as Cyrillic output and as Latin output.

   daily key: `ИЗГЖВ БКДАЕ`
   message key: `ИЫШЬЮ ГДФХЯ`

   There was once an emperor, very great and mighty, and he ruled over an empire so large that no one knew where it began and where it ended. But if nobody could tell the exact extent of his sovereignty everybody was aware that the emperor's right eye laughed, while his left eye wept. One or two men of valour had the courage to go and ask him the reason of this strange fact, but he only laughed and said nothing; and the reason of the deadly enmity between his two eyes was a secret only known to the monarch himself.

   (from “The Fairy of the Dawn” in *The Violet Fairy Book* by Andrew Lang)

2. Use a Fialka with series-68 non-configurable rotors to encrypt this Russian text with the given key. Use the punchcard in Figure 180.2. Discard punctuation and spaces. The lower-case Russian letters almost always look like their upper-case counterparts; the only ones that do not are Аа, Бб, Ее, Ёё, and even then б is the only weird one.

   daily key: `ВИКАБ ГЖДЕЗ`
   message key: `ДЩСЧИ СХЫРС`

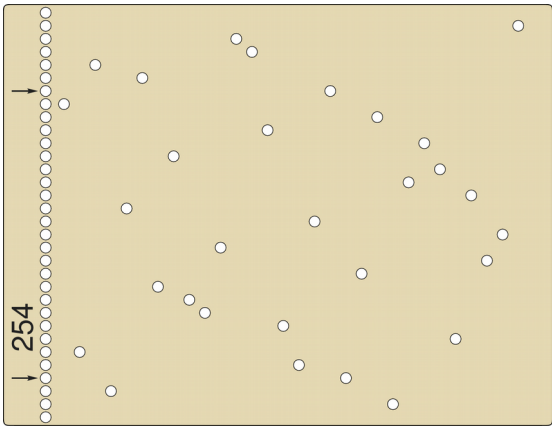
   Возможно ль? вместо роз, Амуром
   насажденных,
   Тюльпанов, гордо наклоненных,
   Душистых ландышей, ясминов и лилей,
   Которых ты всегда любила
   И прежде всякий день носила
   На мраморной груди твоей—
   Возможно ль, милая Климена?

   (from «Красавице, которая нюхала табак» by Aleksandr Sergeevich Pushkin)

3. Decrypt this ciphertext which was prepared with a Fialka using the series-6K configurable rotors and the Czech keyboard. The key is given, and the punchcard is shown below. Spaces, but not punctuation, were kept in the English plaintext.

daily key: ИДАЖЕ ЗВБКГ
ДЧВЩЫ НЫСАЕ
ГИДВБ ЕЗКАЖ
12112 11212
ЛВЩВФ ХАРВТ

message key: ЬЙФШЯ ЬЦЫНЦ

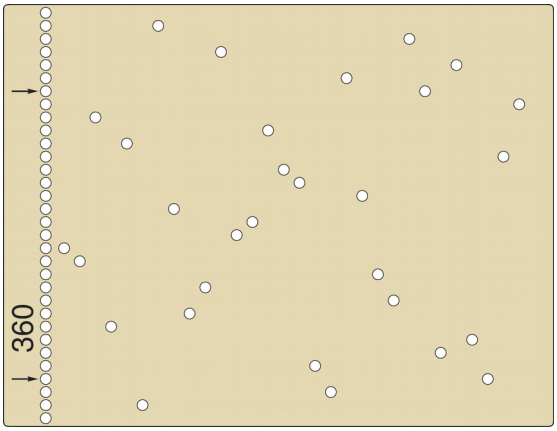



```
ШЙРЙНАЖТМРВТПЩЛШЯДЦСОСХТЛЖБСХПЮИХЧФЛАЧРКНЯЩГНЛОИНКЙБВСПМШЧЕЫ
ДИШЙЗНЕВДЬВЕЯЬСРТХКШЖФМЫЬЯЬЗЧРКНБШОЖЧЩВПЗДНТЦЫСБЗЧУЬРОХЛПХДФ
АСУГЗЖЗХРЫЖДЩУЙСЙИЫСДЙВРТНБАСКУАЛВМКЛНЦЗТЙКТРМЖЩФАЕФМУХЖИЫШР
ЧБЦОЫЛИТМЛЩЮНВМАМТРСШТКЗПЦЦНЙФЖЗЗТСНШЗБГКЬЛЬЕФПКОЕИЮЖХГЧДЖМБ
ШУХЬСТМЫБДВХУЫРРШСЫПЮЧЮЩФКГЫДЬЯПЕБСЮЖСИРЛБТФХПТЦЖСЬЕЦУЖКФЫАМ
ЦЯФЦЬХВБЧТОЬХПРПЮЫНОИЖФЬЖЧНХДКУОМРЙСЯВГХХНЖВЧЖХБШАНКМЖХЖФЩЦЛ
ВОСЦЗЮЗЮРЧЮАЙЛБАДЛШБЧЖЗЬИЖАЮЬЙЗЯПВХГГЩЖЮГВКПРФАЕХСЦРРЛЕЮШТНФ
ССЧЮПУМЙЛВЯДЮЗВАФЩАУЩЖАФРСУЩОФЯФБ
```

4. Find the missing parts of the message key. The text is in English using the Polish keyboard mapping. Spaces, but not punctuation, were retained in the plaintext. The rotor set is from series 3K. The punchcard is shown below.

daily key: КГБДИ ЖЗАВЕ

message key: ПШХГБ ?????




```
ТТТАГЩЖБШПТИЯЯСЕХДЮИКЛАПДЮОЗОМЩДСЛМФОФЛАЧТЗУЯЖЬДГЫДЙМАНСАЩРФ
ЯРВЮШДТХДФЫЫЧВРЮРАЛЗКЗШШГЬХДЩГРНЙПУЖЦНЙЯУСАЬФСИЧЬУИХЖТШКСТЯМ
АДРЖТОЩМПОЬФЙЙАДТРИОЫХРХБСШЕЯВЬЕУВЮЕТШЖГПЧОЖАХГШЬЯЬНМЬКПЖЬНЮ
ГФНОШАЕИЩЖЛЙЧЖФУОВФВКЧЧМГХРЖНААБЫДЧЯЮЬЕЙЬТЖХЛКДКЙЕКЬЗБЛЫУРНФ
МЕКДУПРЧЦБЬСЕХЩОШАННЗИЗБУГШЮЛЙПУЙСВКЬЖЮДФЬЧУААЩНЩИИЛМСУФЧДТГ
ЯЗТХЮУЦЦУНМГЬВЫНБЗЖЬФХМУРБАФЧОЕАЕЯВНЧРЮОЗЕЧРРЙУГФШУЧЛМЯТСМЮЕ
ИАЧББЮЗЮРТЕБФЗУМЧЫЬТДЬМПВЫБШБ
```

5. Learn the Russian language.

6. Используй твою симуляцию Фиалки чтобы зашифровать сообщение Кремлю.

Unit 181
Boolean variables and logical operations

In the next cipher machines that we will see, characters are encoded with electric signals that can be on (1) or off (0), and signals pass from one part of a machine to another with similar values. The basic thing that can have a value of 0 or 1 is called a *bit*. When we are talking about manipulating bits with logic, we may label them with letters and call them *boolean variables* (named after George Boole), and we often use T or TRUE to represent 1 and F or FALSE to represent 0.

The first thing we might want to do to a boolean variable is apply an unary operation to it. An *unary operation* (*unop*) is an operation that acts on a single variable; in arithmetic the negation operation − is an unary operation: $-x$ has the value that is just as far from zero as x but on the other side of it. Suppose we have a boolean variable A and want to see what sorts of unary operations can be applied to it. We can tabulate all possible values of A in a *truth table* and add a column for the output of every possible unary operation. There are four possibilities:

A	UNOP_0	UNOP_1	UNOP_2	UNOP_3
0	0	0	1	1
1	0	1	0	1

Two of these are trivial: UNOP_0 always gives FALSE and UNOP_3 always gives TRUE. Operator UNOP_1 gives the same value that it receives; electrical engineers call this operator a *buffer*, but for us it is useless. Operator UNOP_2 is the interesting one: it flips the bit of A. This operation we call NOT, and we write "NOT A" or "$\neg A$" (or some people write "$\overline{A}$" or "$\sim A$"). At the risk of being overly pedantic, its truth table is in Table 181.1. We can forget about the other three unops.

A	$\neg A$
0	1
1	0

Table 181.1: The NOT *operation.*

Next we look at binary operations. A *binary operation* (*binop*) is one that acts on two variables. You are already familiar with this concept, since addition (+), subtraction (−), multiplication (·), and division (/) are binary operations on real numbers. Since we are dealing with two boolean variables, each of which can have one of two values, there are $2^4 = 16$ possibilities:

A	B	BINOP_0	BINOP_1	BINOP_2	BINOP_3	BINOP_4	BINOP_5	BINOP_6	BINOP_7
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

A	B	BINOP_8	BINOP_9	BINOP_{10}	BINOP_{11}	BINOP_{12}	BINOP_{13}	BINOP_{14}	BINOP_{15}
0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

Again we have some trivial and useless operations: BINOP_0 and BINOP_{15} are constants, and BINOP_3, BINOP_5, BINOP_{12}, and BINOP_{10} are A, B, $\neg A$, and $\neg B$. That leaves ten interesting binary operations.

The AND *operation*

BINOP_1 is TRUE when and only when its operands A and B are both TRUE, so we can rename it the AND operation. It is equivalent to multiplication (modulo 2) (see the unit on modular arithmetic), and it has an identity element (1), and all non-zero elements (just 1) have inverses (still 1). Symbols for this operation are AND, $\wedge$, and $\cdot$.

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Table 181.2: The AND *operation.*

The OR *operation*

BINOP$_7$ is TRUE when A is TRUE or B is TRUE, and FALSE otherwise, so we call it the OR operation. Symbols for this operation are OR and $\vee$.

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Table 181.3: The OR *operation.*

Exclusive or

BINOP$_6$ is TRUE if and only if exactly one of its operands is TRUE. Because it excludes the case when both operands are TRUE, we call rename it the *exclusive or* operation, or XOR. It is equivalent to addition modulo 2 (see the unit on modular arithmetic), and it has an identity element (0), and all elements have inverses ($-0 = 0$ and $-1 = 1$). It is also the same as inequality, since it is TRUE only when the operands are different. It is also the negation of a bi-implication (see below), meaning that the truth of one operand implies the falsehood of the other. Symbols for this operation include XOR, $\oplus$, $\veebar$, $\not\leftrightarrow$, and $\not\equiv$, but we will usually use XOR and $\oplus$.

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Table 181.4: The XOR *operation.*

Nor

BINOP_8 is TRUE when A OR B is FALSE, and FALSE when A OR B is TRUE, so we call it the NOR operation. Symbols for it include NOR, $\downarrow$, and $\bar{\vee}$.

A	B	$A \downarrow B$
0	0	1
0	1	0
1	0	0
1	1	0

Table 181.5: The NOR *operation.*

Not and

BINOP_{14} is the opposite of AND, so we call it the NAND operation. Symbols for it include NAND, $\uparrow$, and $\bar{\wedge}$.

A	B	$A \uparrow B$
0	0	1
0	1	1
1	0	1
1	1	0

Table 181.6: The NAND *operation.*

Implications

A statement like "if A then B" means that whenever A is TRUE then B is also TRUE. It also means that when A is FALSE, then B can have any value. We say that A *implies* B, and write $A \rightarrow B$. From the table of binary operations, we see that BINOP_{13} fits this description: it is TRUE if A and B are both TRUE and FALSE if A is TRUE but B is FALSE; if A is FALSE then the implication is vacuously satisfied, and BINOP_{13} is TRUE. There is no reason that we cannot have implication in the other direction. This is called the *converse* of $A \rightarrow B$, and it is the operation BINOP_{11}. If the operands imply each other, then they must be equal to get a TRUE result. We call this *bi-implication* or *equivalence* or *exclusive nor*. Symbols for it

include $\leftrightarrow$ and $\equiv$, and XNOR. This operation is the opposite (the NOT) of XOR, and is the one we used to call BINOP_9. Finally, BINOP_2 and BINOP_4 are the NOTs of $\rightarrow$ and $\leftarrow$.

A	B	$A \rightarrow B$	$A \leftarrow B$	$A \leftrightarrow B$	$A \nrightarrow B$	$A \nleftarrow B$
0	0	1	1	1	0	0
0	1	1	0	0	0	1
1	0	0	1	0	1	0
1	1	1	1	1	0	0

Table 181.7: Implication operations.

Python tips

Python has a data type called `bool`. A variable of this type can have value `True` or `False`, which are defined as constants in the Python language. However, in a logical expression, any non-zero value is treated as `True`, and any zero value or empty item, such as an empty string or list or dictionary, is treated as `False`. Logical operators `and`, `or`, and `not` do what you expect. The (bitwise) operator `^` can be used as XOR between two `bool` variables.

Reading and references

Any book on symbolic logic.

Python reference: https://docs.python.org/3/reference/expressions.html#boolean-operations

George Boole, *The Mathematical Analysis of Logic: Being an Essay towards a Calculus of Deductive Reasoning*, 1847, https://www.gutenberg.org/ebooks/36884

George Boole, *An Investigation of the Laws of Thought*, 1854.

Wikipedia has many pages about boolean arithmetic and operations. Start with https://en.wikipedia.org/wiki/Boolean_algebra

Exercises

1. Use truth tables to show these facts for any boolean variables A, B, C. Tabulate all possible values for the inputs and check that the column for the left-hand side matches the column for the right-hand side for each equation. This is called *proof by exhaustion*, not only because it is tiring, but also because it exhausts all possibilities.

   a. $A \rightarrow B \;=\; (A \wedge B) \vee (\neg A)$

   b. $(A \wedge B) \wedge C \;=\; A \wedge (B \wedge C)$ (associativity of AND)

   c. $(A \vee B) \vee C \;=\; A \vee (B \vee C)$ (associativity of OR)

   d. $A \wedge (B \vee C) \;=\; (A \wedge B) \vee (A \wedge C)$ (distributivity of AND over OR)

   e. $A \vee (B \wedge C) \;=\; (A \vee B) \wedge (A \vee C)$ (distributivity of OR over AND)

   f. $A \vee (A \wedge B) \;=\; A$ (absorption)

   g. $A \wedge (A \vee B) \;=\; A$ (absorption)

   h. $(A \wedge B) \vee (B \wedge C) \vee (C \wedge A) \;=\; (A \vee B) \wedge (B \vee C) \wedge (C \vee A)$

2. Show that De Morgan's laws hold for any boolean variables A and B:

   a. $\neg (A \wedge B) = (\neg A) \vee (\neg B)$

   b. $\neg (A \vee B) = (\neg A) \wedge (\neg B)$

3. Convince yourself that any binary operation can be written as an expression containing only the operations AND, OR, and NOT (and parentheses). Now convince me.

4. Write a convincing argument that any n-ary logical operation can be written as an expression containing only the operations AND, OR, and NOT (and parentheses). An n-ary operation is one that acts on n operands. We can think of such things as *binary functions* with n arguments (a function that takes n boolean values and returns one boolean value). If your argument is sound, then you have proven that {AND, OR, NOT} is a *complete* set of operators and that any binary function can be written in terms of these three operations.

5. Show that {AND, NOT} is a complete set of operators. Hint: use Exercise 4 and De Morgan's law(s).

6. Show that {OR, NOT} is a complete set of operators.

7. Show that {NAND} is a complete set of operators. That's right—everything can be written in terms of NAND.

8. There is another binary operation that forms a complete set by itself. What is it? Prove it.

Unit 182
Exclusive or

The exclusive or (XOR) operation is special for cryptographers, so it deserves extra attention. Recall from Table 181.4 that its truth table is

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

We also mentioned in Unit 181 that XOR is equivalent to addition modulo 2. It should therefore not surprise you that it has these properties (for all boolean variables A, B, C):

- $(A \oplus B) \oplus C = A \oplus (B \oplus C)$ (associativity) (182.1)
- $A \oplus B = B \oplus A$ (commutivity) (182.2)
- $A \oplus 0 = A$ (identity element) (182.3)
- $0 \oplus 0 = 1 \oplus 1 = 0$ (inverses)

While you would expect the above properties simple because ({TRUE, FALSE}, XOR) is an abelian group (the same as $(\mathbb{Z}_2, +_2)$), it also has some other very nice properties:

- $A \oplus 1 = \neg A$ (182.4)
- $A \oplus A = 0$ (nilpotence) (182.5)
- $A \oplus \neg A = 1$
- $A \oplus B = C \quad \Rightarrow \quad A = B \oplus C$
- $A \oplus (\neg B) = (\neg A) \oplus B = \neg (A \oplus B)$

That fourth one is important, because when we use XOR as an encryption mechanism, we can apply the same key to decipher as we use to encipher. What we mean is clearer if we replace A and B with P (plaintext) and K (key), while C denotes the ciphertext:

$$P \oplus K = C \quad \Rightarrow \quad P = C \oplus K \tag{182.6}$$

$$(P \oplus K) \oplus K = P$$

Reading and references

Any good textbook on symbolic logic.

Wikipedia: https://en.wikipedia.org/wiki/Exclusive_or

Exercises

1. Use a truth tables or manipulate equations to show that for any boolean variables A and B,

   a. $A \oplus B = (A \vee B) \wedge \neg (A \wedge B)$

   b. $A \oplus B = (A \vee B) \wedge ((\neg A) \vee (\neg B))$

   c. $(A \oplus B) \oplus B = A$

2. Use truth tables or manipulate equations to prove each of the properties of XOR listed in this unit.

Unit 183
Baudot encoding

During and after World War II, texting (“SMS” [“short-message service”] for European readers) was done with large devices, called “teleprinters,” that did not fit easily into one’s pocket. Bits were very expensive then, and most users could not afford a complete byte of eight bits. Therefore, teleprinters used five parallel transmission lines, and a means of encoding characters with five bits (voltage on and off for each line) was needed. By this time, however, punctuation and digits had been invented, and message senders needed more than the thirty-two possibilities afforded by five bits. Hence, the symbol and letter shifts were invented. A special character, called *symbol shift*, changes the character set to one of digits and punctuation. Another special character, the *letter shift*, returns the teleprinter to the character set containing letters. Upper-case letters were all that existed, so there was no such thing as that sort of shift key. The entire scheme, including the assignments of bit patterns to characters, is called *Baudot* encoding. In Table 183.1 we see the bit patterns and the characters assigned to them for both letter and symbol mode. In addition, we see the notation used by cryptanalysts at Bletchley Park during World War II as they worked to break the cipher machines that used it. These machines used innovative ways to flip and/or rearrange bits with rotors. Unlike the Enigma and such machines, in which the rotors defined permutations of the alphabet, these new machines used rotors to control the flipping or permuting of the five bits that encoded the characters.

Python tips

The carriage return is written “`\r`” in strings and bytes, and line feed is “`\n`” (think “return” and “new line”). The end of a line usually has both of these: `\r\n`. A null character is written “`\0`”. The “bell” character is “`\a`”, and if your terminal supports it, it will be converted to a sound (this author’s terminal flashes a bell icon but makes no noise). It is possible that the “enquire” character is “`\x05`”, but we will not need this character in this book.

bits	letter mode	symbol mode	Bletchley notation
00000	[null]	[null]	/
10000	E	3	E
01000	[carriage return]	[carriage return]	4
11000	A	-	A
00100	[space]	[space]	9
10100	S	'	S
01100	I	8	I
11100	U	7	U
00010	[line feed]	[line feed]	3
10010	D	[enquire]	D
01010	R	4	R
11010	J	[bell]	J
00110	N	,	N
10110	F	!	F
01110	C	:	C
11110	K	(	K
00001	T	5	T
10001	Z	+	Z
01001	L	)	L
11001	W	2	W
00101	H	£	H
10101	Y	6	Y
01101	P	0	P
11101	Q	1	Q
00011	O	9	O
10011	B	?	B
01011	G	&	G
11011	[symbol shift]		+
00111	M	.	M
10111	X	/	X
01111	V	;	V
11111		[letter shift]	8

Table 183.1: Baudot symbol table.

Programming tasks

1. Implement an encoder/decoder that can translate between text containing letters, punctuation, and digits and five-bit Baudot.

2. Implement an encoder/decoder that can translate between five-bit Baudot and the Bletchley notation.

3. Create a function that takes text in Bletchley notation and prints it with spaces, punctuation, digits, and end-of-line characters.

4. Take your textual corpus (Task 1 of Unit 3), including spaces, punctuation, and occasional digits, and encode it with Bletchley Baudot. Use the result to compile a table of frequencies for the 32 symbols. Put this table into a data structure that you can import into your Python scripts.

5. Make a copy of your function that calculates the index of coincidence (Unit 12) and modify it to operate on strings of Baudot characters. Take some texts of various lengths (at least 100 characters) in order to find a range for the IoC of English text.

6. Take your textual corpus (Task 1 of Unit 3), including spaces, punctuation, and occasional digits, and encode it with Bletchley Baudot. From the result, compile a table of tetragram frequencies. The table will have 32^4 (just over a million) entries.

7. Use the table from Task 6 and make a modified copy of your tetragram-fitness function (Unit 11) that evaluates the fitness of a piece of text in Bletchley Baudot encoding. Experiment with texts of various lengths in order to find a range of fitness for typical English.

Unit 184
Siemens and Halske T52

The Siemens and Halske model T52 (code-named “Sturgeon” by British analysts) was an encryption machine that acted on the output (for encryption) or input (decryption) to a teleprinter. A teleprinter used five data lines and encoded/decoded its messages in Baudot (see Unit 183). The T52 performed two sorts of operations on the data in those five lines: it could flip a bit (change it from 0 to 1 or from 1 to 0), or it could swap two bits. Which bits were flipped and which were swapped was determined by ten wheels and some circuitry. Unlike the wheels in machines like the Enigma, the wheels of the T52 did not act on letters directly; rather, they provided control signals to the circuitry that performed operations on the five bits of the Baudot-encoded characters. The output of each wheel is mapped to the control of one bit operation by something like a plugboard and optionally some additional circuitry. There were five versions of the machine, which we will examine below; they differ in how the wheels rotate and in what additional circuitry they have.

The ten wheel are designated A, B, C, D, E, F, G, H, J, and K (“I” was skipped, probably because it looks like a Roman numeral). Each has its own number of positions: 73, 71, 69, 67, 65, 64, 61, 59, 53, and 47, respectively. These numbers are all coprime, so that if the wheels advance regularly (one step each after encrypting each character, as in some versions of the machine), then they will not return to their original configuration until 893,622,318,929,520,960 characters have been encrypted. At the time that the machine was in use, that was considered a big number. The wheels have pins that determine whether it emits a control signal that is active (“on” or “1”) or inactive (“off” or “0”) for each of its positions, which are numbered, starting from 1. The pin patterns of the ten wheels are given in Table 184.1.

```
A    ·××·×××···×··××··×··××××··×××·×·×·×××··××··×××·××·····×××···××××·×·····×
B    ·××××··××××××···×···×·×··××·×××××·····×××···××·×××·····××·×·×·×××·····×
C    ·×××××·×·××··×·××····××·×····××××·×××·×××···×··×··×××····××····××···×
D    ·××·×·×···×××·····××·×····×··×××··×××··×·×××···××··×··×·××××··×···×
E    ·×···×···×·×××××××·×·××××××··×·×··××·×····×·×××××··×××·×·×·××···×
F    ·×·×·×××××····×·××××··×·××·××××·××··×××····×····×××·××××××···×·×
G    ·××××···××··××···×·×××××·×··×··×××·×·××·×·××·×····××·×··×·×××
H    ·×··×××××···××××·×··××××···××·×···××·×·××···××××××··××·×·××
J    ·×·××·××··××·×·××···×××···×××··×·×××·×····××××···×·××
K    ·×····×××××···×··×·×××··×·×·×××··×××·××××···×××
```

Table 184.1: Pin patterns of the wheels of the T52. A cross indicates an active position (pin is present), and a dot indicates an inactive position (no pin). The first dot is at position 1.

Bit-flipping is done by five exclusive-or logic circuits. Exclusive OR (XOR) takes two input bits and outputs a 1 if and only if exactly one input is 1. Otherwise, the output is 0. See Table 184.2. The symbol for XOR is usually taken to be ⊕, since it has a close relationship with addition modulo 2 (see Unit 19) (actually, it *is* addition modulo 2). In the T52, the two inputs to an XOR circuit are a bit of the plaintext character and a control wire. Those control wires are labeled "I" for the circuit that acts on the first bit, "II" for the second, "III," "IV," and "V." So the first bit is flipped if the control wire I is active, the second if II is active, etc.

inputs	output of OR	output of XOR
0 0	0	0
0 1	1	1
1 0	1	1
1 1	1	0

Table 184.2: The OR *and exclusive* OR *(*XOR*) operations, from Tables 181.3 and 181.4. The output of* OR *is on if either input is on. The output of* XOR *is on if and only if exactly one input is on.*

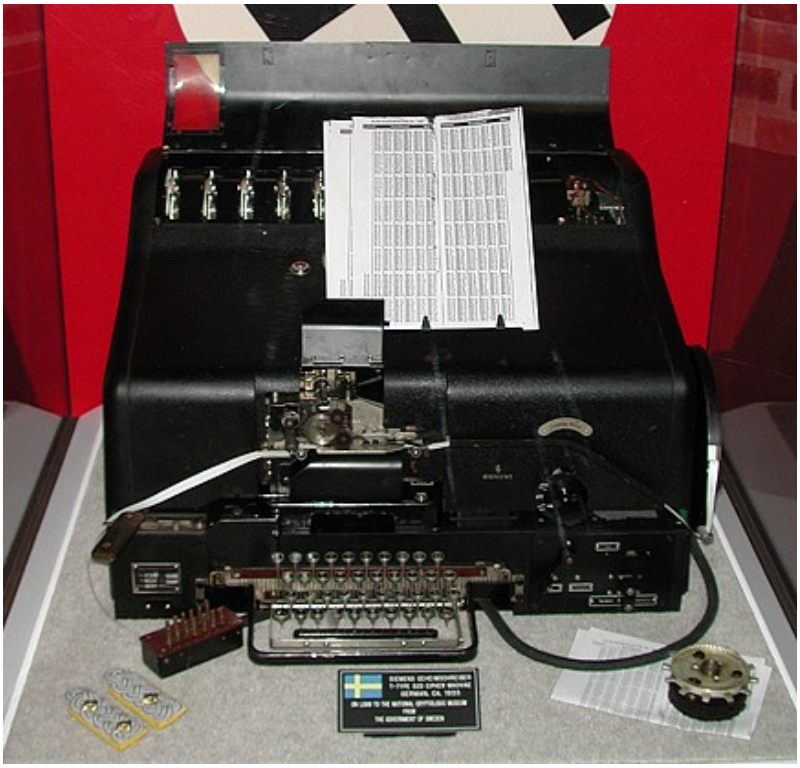


Figure 184.1: Siemens and Halske T52 version D on display at the U.S. National Cryptologic Museum. Photo credit: U.S. National Security Agency.

Bit-swapping is performed by relays that connect to the data lines and which are controlled by the control lines from the wheels in such a way that if the control line is active (1) then the bits are not swapped, but if the control line is inactive (0) then they are. There are five such relays, and each can be plugged into the data lines at two points. Therefore, we need ten points of access to the data; they are numbered and displayed schematically in Figure 184.2. A relay plugged into points 1 and 5, for example, can swap the bits on lines 1 and 4. This swapping comes *after* a swap that is wired to points 4 and 6, for example. Any possible wiring of the permutation circuit can be represented as five transpositions (swappings) in a definite order. For example, the fixed standard wiring of version C of the machine is 1-2, 3-4, 5-6, 7-8, 9-10, which corresponds to transpositions (1 5), (4 5), (3 4), (2 3), and (1 2), in that order.

There are 5! = 120 possible permutations of five bits. However, with five transpositions, there are only $2^5 = 32$ that can be implemented. There happen to be 120 ways to wire the swapping relays so that 32 permutations can be enacted, and there are 240 ways to wire them so that 30 permutations are possible. These 360 wirings were considered acceptable by the Germans, and we have listed them in Table 184.3. Any wiring not listed would result in fewer possible permutations or in loops in the data lines that disrupt the operation of the machine.

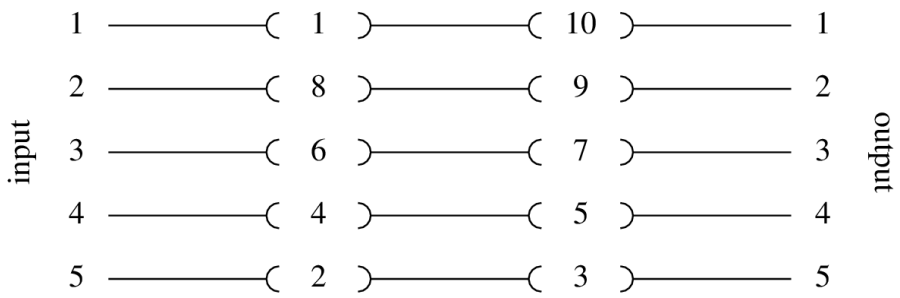


Figure 184.2: Schematic diagram of the access points and their labels for the swapping relays in the permutation circuitry. For example, a relay plugged into points 1 and 5 controls the swapping of bits 1 and 4.

The rest of this section is divided into subsections for each of the five versions of the T52. They all use the same wheels, the same XOR circuitry, and the same permutation circuitry. However, some fix the wiring of the swapping relays so that the permutation circuitry cannot be changed. Some versions introduce new features, such as additional logic circuitry between the wheels and the operations on the data bits, or irregular stepping of the wheels.

1-2 3-4 5-6 7-8 9-10	1-2 3-4 5-6 7-9 8-10	1-2 3-4 5-7 6-8 9-10	1-2 3-4 5-7 6-9 8-10	1-2 3-4 5-8 6-10 7-9
1-2 3-4 5-8 6-9 7-10	1-2 3-4 5-9 6-10 7-8	1-2 3-4 5-9 6-8 7-10	1-2 3-5 4-6 7-8 9-10	1-2 3-5 4-6 7-9 8-10
1-2 3-5 4-7 6-8 9-10	1-2 3-5 4-7 6-9 8-10	1-2 3-5 4-8 6-10 7-9	1-2 3-5 4-8 6-9 7-10	1-2 3-5 4-9 6-10 7-8
1-2 3-5 4-9 6-8 7-10	1-2 3-6 4-10 5-8 7-9	1-2 3-6 4-10 5-9 7-8	1-2 3-6 4-7 5-8 9-10	1-2 3-6 4-7 5-9 8-10
1-2 3-6 4-8 5-10 7-9	1-2 3-6 4-8 5-7 9-10	1-2 3-6 4-9 5-10 7-8	1-2 3-6 4-9 5-7 8-10	1-2 3-7 4-10 5-8 6-9
1-2 3-7 4-10 5-9 6-8	1-2 3-7 4-6 5-8 9-10	1-2 3-7 4-6 5-9 8-10	1-2 3-7 4-8 5-10 6-9	1-2 3-7 4-8 5-6 9-10
1-2 3-7 4-9 5-10 6-8	1-2 3-7 4-9 5-6 8-10	1-2 3-8 4-10 5-6 7-9	1-2 3-8 4-10 5-7 6-9	1-2 3-8 4-6 5-10 7-9
1-2 3-8 4-6 5-9 7-10	1-2 3-8 4-7 5-10 6-9	1-2 3-8 4-7 5-9 6-10	1-2 3-8 4-9 5-6 7-10	1-2 3-8 4-9 5-7 6-10
1-2 3-9 4-10 5-6 7-8	1-2 3-9 4-10 5-7 6-8	1-2 3-9 4-6 5-10 7-8	1-2 3-9 4-6 5-8 7-10	1-2 3-9 4-7 5-10 6-8
1-2 3-9 4-7 5-8 6-10	1-2 3-9 4-8 5-6 7-10	1-2 3-9 4-8 5-7 6-10	1-3 2-4 5-6 7-8 9-10	1-3 2-4 5-6 7-9 8-10
1-3 2-4 5-7 6-8 9-10	1-3 2-4 5-7 6-9 8-10	1-3 2-4 5-8 6-10 7-9	1-3 2-4 5-8 6-9 7-10	1-3 2-4 5-9 6-10 7-8
1-3 2-4 5-9 6-8 7-10	1-3 2-5 4-6 7-8 9-10	1-3 2-5 4-6 7-9 8-10	1-3 2-5 4-7 6-8 9-10	1-3 2-5 4-8 6-10 7-9
1-3 2-5 4-8 6-9 7-10	1-3 2-5 4-9 6-8 7-10	1-3 2-6 4-10 5-8 7-9	1-3 2-6 4-10 5-9 7-8	1-3 2-6 4-7 5-8 9-10
1-3 2-6 4-7 5-9 8-10	1-3 2-6 4-8 5-10 7-9	1-3 2-6 4-8 5-7 9-10	1-3 2-6 4-9 5-10 7-8	1-3 2-6 4-9 5-7 8-10
1-3 2-7 4-10 5-9 6-8	1-3 2-7 4-6 5-8 9-10	1-3 2-7 4-6 5-9 8-10	1-3 2-7 4-8 5-10 6-9	1-3 2-7 4-8 5-6 9-10
1-3 2-7 4-9 5-10 6-8	1-3 2-8 4-10 5-6 7-9	1-3 2-8 4-10 5-7 6-9	1-3 2-8 4-6 5-10 7-9	1-3 2-8 4-6 5-9 7-10
1-3 2-8 4-7 5-10 6-9	1-3 2-8 4-7 5-9 6-10	1-3 2-8 4-9 5-6 7-10	1-3 2-8 4-9 5-7 6-10	1-3 2-9 4-10 5-7 6-8
1-3 2-9 4-6 5-10 7-8	1-3 2-9 4-6 5-8 7-10	1-3 2-9 4-7 5-10 6-8	1-3 2-9 4-8 5-6 7-10	1-3 2-9 4-8 5-7 6-10
1-4 2-10 3-6 5-8 7-9	1-4 2-10 3-6 5-9 7-8	1-4 2-10 3-7 5-8 6-9	1-4 2-10 3-7 5-9 6-8	1-4 2-10 3-8 5-6 7-9
1-4 2-10 3-8 5-7 6-9	1-4 2-10 3-9 5-6 7-8	1-4 2-10 3-9 5-7 6-8	1-4 2-5 3-6 7-8 9-10	1-4 2-5 3-6 7-9 8-10
1-4 2-5 3-7 6-8 9-10	1-4 2-5 3-7 6-9 8-10	1-4 2-5 3-8 6-10 7-9	1-4 2-5 3-8 6-9 7-10	1-4 2-5 3-9 6-10 7-8
1-4 2-5 3-9 6-8 7-10	1-4 2-6 3-10 5-8 7-9	1-4 2-6 3-10 5-9 7-8	1-4 2-6 3-5 7-8 9-10	1-4 2-6 3-5 7-9 8-10
1-4 2-6 3-8 5-7 9-10	1-4 2-6 3-8 5-9 7-10	1-4 2-6 3-9 5-7 8-10	1-4 2-6 3-9 5-8 7-10	1-4 2-7 3-10 5-8 6-9
1-4 2-7 3-10 5-9 6-8	1-4 2-7 3-5 6-8 9-10	1-4 2-7 3-5 6-9 8-10	1-4 2-7 3-8 5-6 9-10	1-4 2-7 3-8 5-9 6-10
1-4 2-7 3-9 5-6 8-10	1-4 2-7 3-9 5-8 6-10	1-4 2-8 3-10 5-6 7-9	1-4 2-8 3-10 5-7 6-9	1-4 2-8 3-5 6-10 7-9
1-4 2-8 3-5 6-9 7-10	1-4 2-8 3-6 5-7 9-10	1-4 2-8 3-6 5-9 7-10	1-4 2-8 3-7 5-6 9-10	1-4 2-8 3-7 5-9 6-10
1-4 2-9 3-10 5-6 7-8	1-4 2-9 3-10 5-7 6-8	1-4 2-9 3-5 6-10 7-8	1-4 2-9 3-5 6-8 7-10	1-4 2-9 3-6 5-7 8-10
1-4 2-9 3-6 5-8 7-10	1-4 2-9 3-7 5-6 8-10	1-4 2-9 3-7 5-8 6-10	1-5 2-10 3-6 4-8 7-9	1-5 2-10 3-7 4-8 6-9
1-5 2-10 3-7 4-9 6-8	1-5 2-10 3-8 4-6 7-9	1-5 2-10 3-9 4-6 7-8	1-5 2-10 3-9 4-7 6-8	1-5 2-4 3-6 7-8 9-10
1-5 2-4 3-6 7-9 8-10	1-5 2-4 3-7 6-8 9-10	1-5 2-4 3-7 6-9 8-10	1-5 2-4 3-8 6-10 7-9	1-5 2-4 3-8 6-9 7-10
1-5 2-4 3-9 6-10 7-8	1-5 2-4 3-9 6-8 7-10	1-5 2-6 3-10 4-8 7-9	1-5 2-6 3-10 4-9 7-8	1-5 2-6 3-4 7-8 9-10
1-5 2-6 3-4 7-9 8-10	1-5 2-6 3-8 4-7 9-10	1-5 2-6 3-8 4-9 7-10	1-5 2-6 3-9 4-7 8-10	1-5 2-6 3-9 4-8 7-10
1-5 2-7 3-10 4-8 6-9	1-5 2-7 3-10 4-9 6-8	1-5 2-7 3-4 6-8 9-10	1-5 2-7 3-8 4-6 9-10	1-5 2-7 3-9 4-6 8-10
1-5 2-7 3-9 4-8 6-10	1-5 2-8 3-10 4-6 7-9	1-5 2-8 3-10 4-7 6-9	1-5 2-8 3-4 6-10 7-9	1-5 2-8 3-4 6-9 7-10
1-5 2-8 3-6 4-7 9-10	1-5 2-8 3-6 4-9 7-10	1-5 2-8 3-7 4-6 9-10	1-5 2-8 3-7 4-9 6-10	1-5 2-9 3-10 4-6 7-8
1-5 2-9 3-10 4-7 6-8	1-5 2-9 3-4 6-8 7-10	1-5 2-9 3-6 4-8 7-10	1-5 2-9 3-7 4-6 8-10	1-5 2-9 3-7 4-8 6-10
1-6 2-10 3-4 5-8 7-9	1-6 2-10 3-4 5-9 7-8	1-6 2-10 3-5 4-8 7-9	1-6 2-10 3-5 4-9 7-8	1-6 2-10 3-8 4-7 5-9
1-6 2-10 3-8 4-9 5-7	1-6 2-10 3-9 4-7 5-8	1-6 2-10 3-9 4-8 5-7	1-6 2-4 3-10 5-8 7-9	1-6 2-4 3-10 5-9 7-8
1-6 2-4 3-7 5-8 9-10	1-6 2-4 3-7 5-9 8-10	1-6 2-4 3-8 5-10 7-9	1-6 2-4 3-8 5-7 9-10	1-6 2-4 3-9 5-10 7-8
1-6 2-4 3-9 5-7 8-10	1-6 2-5 3-10 4-8 7-9	1-6 2-5 3-10 4-9 7-8	1-6 2-5 3-7 4-8 9-10	1-6 2-5 3-7 4-9 8-10
1-6 2-5 3-8 4-10 7-9	1-6 2-5 3-8 4-7 9-10	1-6 2-5 3-9 4-10 7-8	1-6 2-5 3-9 4-7 8-10	1-6 2-7 3-4 5-8 9-10
1-6 2-7 3-4 5-9 8-10	1-6 2-7 3-5 4-8 9-10	1-6 2-7 3-5 4-9 8-10	1-6 2-7 3-8 4-10 5-9	1-6 2-7 3-8 4-9 5-10
1-6 2-7 3-9 4-10 5-8	1-6 2-7 3-9 4-8 5-10	1-6 2-8 3-10 4-7 5-9	1-6 2-8 3-10 4-9 5-7	1-6 2-8 3-4 5-10 7-9
1-6 2-8 3-4 5-7 9-10	1-6 2-8 3-5 4-10 7-9	1-6 2-8 3-5 4-7 9-10	1-6 2-8 3-7 4-10 5-9	1-6 2-8 3-7 4-9 5-10
1-6 2-9 3-10 4-7 5-8	1-6 2-9 3-10 4-8 5-7	1-6 2-9 3-4 5-10 7-8	1-6 2-9 3-4 5-7 8-10	1-6 2-9 3-5 4-10 7-8
1-6 2-9 3-5 4-7 8-10	1-6 2-9 3-7 4-10 5-8	1-6 2-9 3-7 4-8 5-10	1-7 2-10 3-4 5-9 6-8	1-7 2-10 3-5 4-8 6-9
1-7 2-10 3-5 4-9 6-8	1-7 2-10 3-8 4-6 5-9	1-7 2-10 3-9 4-6 5-8	1-7 2-10 3-9 4-8 5-6	1-7 2-4 3-10 5-8 6-9
1-7 2-4 3-10 5-9 6-8	1-7 2-4 3-6 5-8 9-10	1-7 2-4 3-6 5-9 8-10	1-7 2-4 3-8 5-10 6-9	1-7 2-4 3-8 5-6 9-10
1-7 2-4 3-9 5-10 6-8	1-7 2-4 3-9 5-6 8-10	1-7 2-5 3-10 4-8 6-9	1-7 2-5 3-10 4-9 6-8	1-7 2-5 3-6 4-8 9-10
1-7 2-5 3-8 4-6 9-10	1-7 2-5 3-9 4-10 6-8	1-7 2-5 3-9 4-6 8-10	1-7 2-6 3-4 5-8 9-10	1-7 2-6 3-4 5-9 8-10
1-7 2-6 3-5 4-8 9-10	1-7 2-6 3-5 4-9 8-10	1-7 2-6 3-8 4-10 5-9	1-7 2-6 3-8 4-9 5-10	1-7 2-6 3-9 4-10 5-8
1-7 2-6 3-9 4-8 5-10	1-7 2-8 3-10 4-6 5-9	1-7 2-8 3-10 4-9 5-6	1-7 2-8 3-4 5-10 6-9	1-7 2-8 3-4 5-6 9-10
1-7 2-8 3-5 4-10 6-9	1-7 2-8 3-5 4-6 9-10	1-7 2-8 3-6 4-10 5-9	1-7 2-8 3-6 4-9 5-10	1-7 2-9 3-10 4-6 5-8
1-7 2-9 3-10 4-8 5-6	1-7 2-9 3-4 5-10 6-8	1-7 2-9 3-5 4-10 6-8	1-7 2-9 3-5 4-6 8-10	1-7 2-9 3-6 4-8 5-10
1-8 2-10 3-4 5-6 7-9	1-8 2-10 3-4 5-7 6-9	1-8 2-10 3-5 4-6 7-9	1-8 2-10 3-5 4-7 6-9	1-8 2-10 3-6 4-7 5-9
1-8 2-10 3-6 4-9 5-7	1-8 2-10 3-7 4-6 5-9	1-8 2-10 3-7 4-9 5-6	1-8 2-4 3-10 5-6 7-9	1-8 2-4 3-10 5-7 6-9
1-8 2-4 3-6 5-10 7-9	1-8 2-4 3-6 5-9 7-10	1-8 2-4 3-7 5-10 6-9	1-8 2-4 3-7 5-9 6-10	1-8 2-4 3-9 5-6 7-10
1-8 2-4 3-9 5-7 6-10	1-8 2-5 3-10 4-6 7-9	1-8 2-5 3-10 4-7 6-9	1-8 2-5 3-6 4-10 7-9	1-8 2-5 3-6 4-9 7-10
1-8 2-5 3-7 4-10 6-9	1-8 2-5 3-7 4-9 6-10	1-8 2-5 3-9 4-6 7-10	1-8 2-5 3-9 4-7 6-10	1-8 2-6 3-10 4-7 5-9
1-8 2-6 3-10 4-9 5-7	1-8 2-6 3-4 5-10 7-9	1-8 2-6 3-4 5-9 7-10	1-8 2-6 3-5 4-10 7-9	1-8 2-6 3-5 4-9 7-10
1-8 2-6 3-9 4-10 5-7	1-8 2-6 3-9 4-7 5-10	1-8 2-7 3-10 4-6 5-9	1-8 2-7 3-10 4-9 5-6	1-8 2-7 3-4 5-10 6-9
1-8 2-7 3-4 5-9 6-10	1-8 2-7 3-5 4-10 6-9	1-8 2-7 3-5 4-9 6-10	1-8 2-7 3-9 4-10 5-6	1-8 2-7 3-9 4-6 5-10
1-8 2-9 3-4 5-6 7-10	1-8 2-9 3-4 5-7 6-10	1-8 2-9 3-5 4-6 7-10	1-8 2-9 3-5 4-7 6-10	1-8 2-9 3-6 4-10 5-7
1-8 2-9 3-6 4-7 5-10	1-8 2-9 3-7 4-10 5-6	1-8 2-9 3-7 4-6 5-10	1-9 2-10 3-4 5-7 6-8	1-9 2-10 3-5 4-6 7-8
1-9 2-10 3-5 4-7 6-8	1-9 2-10 3-6 4-8 5-7	1-9 2-10 3-7 4-6 5-8	1-9 2-10 3-7 4-8 5-6	1-9 2-4 3-10 5-6 7-8
1-9 2-4 3-10 5-7 6-8	1-9 2-4 3-6 5-10 7-8	1-9 2-4 3-6 5-8 7-10	1-9 2-4 3-7 5-10 6-8	1-9 2-4 3-7 5-8 6-10
1-9 2-4 3-8 5-6 7-10	1-9 2-4 3-8 5-7 6-10	1-9 2-5 3-10 4-6 7-8	1-9 2-5 3-10 4-7 6-8	1-9 2-5 3-6 4-8 7-10
1-9 2-5 3-7 4-10 6-8	1-9 2-5 3-7 4-8 6-10	1-9 2-5 3-8 4-6 7-10	1-9 2-6 3-10 4-7 5-8	1-9 2-6 3-10 4-8 5-7
1-9 2-6 3-4 5-10 7-8	1-9 2-6 3-4 5-8 7-10	1-9 2-6 3-5 4-10 7-8	1-9 2-6 3-5 4-8 7-10	1-9 2-6 3-8 4-10 5-7
1-9 2-6 3-8 4-7 5-10	1-9 2-7 3-10 4-6 5-8	1-9 2-7 3-10 4-8 5-6	1-9 2-7 3-4 5-10 6-8	1-9 2-7 3-5 4-10 6-8
1-9 2-7 3-5 4-8 6-10	1-9 2-7 3-8 4-6 5-10	1-9 2-8 3-4 5-6 7-10	1-9 2-8 3-4 5-7 6-10	1-9 2-8 3-5 4-6 7-10
1-9 2-8 3-5 4-7 6-10	1-9 2-8 3-6 4-10 5-7	1-9 2-8 3-6 4-7 5-10	1-9 2-8 3-7 4-10 5-6	1-9 2-8 3-7 4-6 5-10

Table 184.3: Acceptable permutation-circuit wirings for the T52 versions A, B, and D.

Versions A and B

The first two versions of the machine differed only superficially; their operation was identical. They contain no new elements beyond what we have already discussed. The machine is initialized by putting the ten wheels into the positions specified by part of the key. The output of each wheel is assigned to control a bit-flipping (XOR) or a bit-swapping (one of the relays) via a plugboard; this mapping is another part of the key. After each character is encrypted, all ten wheels advance (rotate) their positions by one; this is what we call "regular stepping." A schematic diagram of the T52a/b is in Figure 184.4.

To specify the key, we need to give the assignments and starting positions of the ten wheels. The names of the wheels are implied, so if we write the key as

9-10, 1-6, I, II, 3-5, 4-7, IV, 2-8, III, V; 52, 19, 21, 50, 47, 24, 14, 46, 39, 7

we mean that wheel A starts at position 52 and is assigned to a swapping relay that is connected to points 9 and 10 (see Figure 184.2), wheel B starts at 19 and is assigned to a swapping relay between points 1 and 6. wheel C starts at 21 and is assigned to the XOR on data line 1, etc.

Let's see how to translate between the language in which the key is expressed into a set of operations on the data bits. For the example above, we see that the XOR operations are controlled by the output of wheels C, D, J, G, K. To glean which bits are swapped in the permutation circuit, we can shift things in Figure 184.2 so that points connected to the same relay are in the same vertical segment, as seen in Figure 184.3. From this we see that wheel B controls a transposition (1 3), which occurs first (if it occurs, based on the output of B). It is followed by (2 5) which is controlled by H, then (3 4) controlled by F, (4 5) controlled by E, and finally (1 2) controlled by A.

To see that we understand the machine and that your implementation of it is working, consider the following test vector. Encode the first line of Hamlet's soliloquy in Baudot:

`TO9BE9OR9NOT9TO9BE+N89THAT9IS9THE9QUESTION+C843`

Encrypt with T52a/b using the example key to get

`9NQHMTA9XYJDYSFVUOAZ9/YEIW/8GX84RAURJSRA/9KOMH9`

The encryption of the first letter goes as follows. The wheels are in their initial positions, so A's output is 0, B 0, C 0, D 0, E 1, F 0, G 1, H 1, J 0, K 1. The first character is `T`, which has bits 00001. Its first bit is XORed with 0 (the output of C) to remain 0. The second bit is XORed with 0 (output of D) to remain 0. The third is XORed with 0 (output of J) to remain 0. The fourth is XORed with 1 (output of G) to become 1. The last is XORed with 1 (output of K) to become 0. After the XOR circuitry, the data lines carry the bits 00010. The output of B is 0, so the first transposition occurs, and bits 1 and 3 are swapped (no effect). H outputs 1, so its transposition does not occur. Because F outputs 0, the transpositions (3 4) does occur and now we have 00100 on the data lines. The output of wheel E is 1, so its transposition does not occur. The last transposition, (1 2) occurs, but it swaps two zero bits. The output of the machine is therefore 00100, which is written as `9` in Bletchley notation and which represents a space.

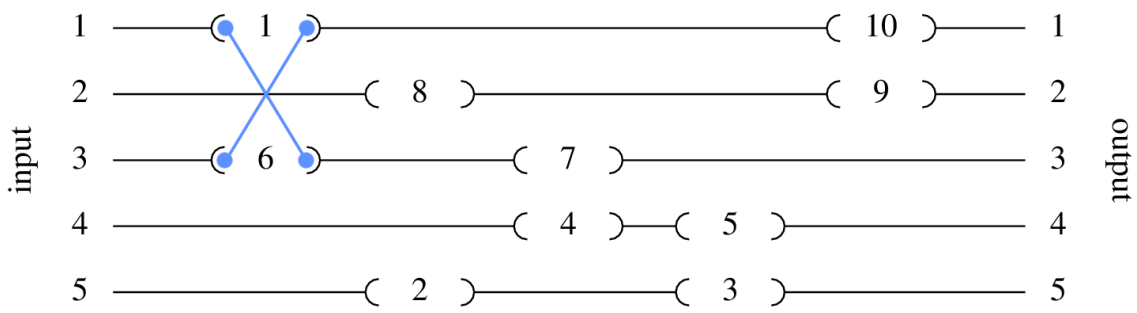


Figure 184.3: Example of shifting the access points in the permutation circuit so that we can glean the transpositions. Note that access points cannot be shifted through one another.

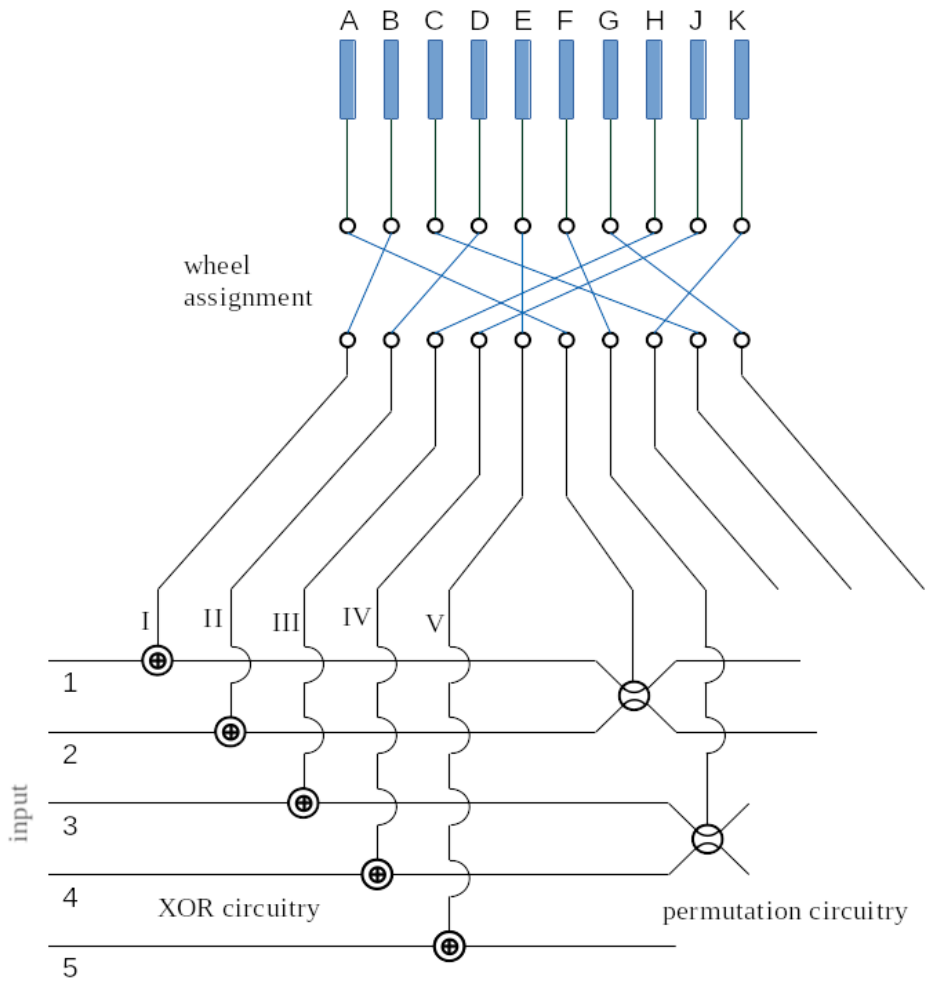


Figure 184.4: Schematic diagram of the T52a/b. The diagram is incomplete; only two transpositions are shown in the permutation circuitry.

Version C

Version C has the following changes from versions A and B:

- Between the wheels and the wheel-assignment plugboard is a set of switches for setting a message key and circuitry to shuffle the signals on the control lines from the wheels. When too many messages are sent with the same daily key, it is easy for analysts to reconstruct the plaintexts. Modifying the key for each message obviates this problem.
- After the control lines leave the wheel-assignment plugboard, they pass through what is called the “SR logic” module which uses XOR circuits to further scramble them.
- The permutation circuitry is fixed so that only the wiring 1-2, 3-4, 5-6, 7-8, 9-10 is used. The corresponding transpositions (in order) are (1 5), (4 5), (3 4), (2 3), (1 2). Because only one configuration of the permutation circuitry is used, the connections are renamed with abbreviated labels: “1” for 1-2, “3” for 3-4, “5” for 5-6, “7” for 7-8, and “9” for 9-10.

The message-key unit has five switches, each of which can be set to positions labeled `P`, `S`, `T`, `U`, `W`, `X`, `Y`, and `Z`. Each switch’s position activates up to three transposition relays that can swap the signals on the control lines leaving the wheels; see Table 184.4. The transposition relays are placed into three layers, so that layer 1 acts before layer 2, which acts before layer 3; see Table 184.5 for the transpositions and their placements into the three layers. It goes without saying that the action of the message-key unit and the wheel-assignment plugboard together are like using the plugboard with a new set of assignments; the addition of the message-key unit does not change the T52 in a fundamental way.

As an example of the function of the message-key unit, consider message key `WUPSY`. Switch 1 is in position `W`, which means that transpositions T01 and T06 are active. Switch 2 is in position `U`, so T02 and T07 are active. Switch 3 activates T03. Switch 4 activates T09 and T14. Switch 5 activates T05 and T15. In the first layer of transposition relays, T01, T02, T03, and T05 are active. From Table 184.5 we see that the control signals from wheels A and B are swapped, from C and D are swapped, from E and F are swapped, and from J and K are swapped. In other words, line B now carries the signal from wheel A, line A now carries the signal from wheel B, etc. In the second layer, T06, T07, and T09 are active. So the signals on lines B and C are swapped, D and E are swapped, and H and J are swapped. In layer 3, T14 and T15 are active, and they swap the signals on lines D and J, and E and K. The overall permutation enacted by the message-key unit in our example takes the signal from wheel A and puts it onto line C, the signal from wheel B onto line A, etc.:

A	B	C	D	E	F	G	H	J	K
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
C	A	J	B	F	K	G	D	E	H

switch number	transpositions (swappings)	switch positions P	S	T	U	W	X	Y	Z
1	T01				✓	✓	✓		✓
	T06		✓			✓		✓	✓
	T11			✓			✓	✓	✓
2	T02			✓	✓	✓		✓	
	T07	✓			✓		✓	✓	
	T12		✓			✓	✓	✓	
3	T03	✓	✓	✓		✓			
	T08		✓		✓	✓		✓	
	T13			✓	✓	✓			✓
4	T04	✓		✓				✓	✓
	T09		✓	✓		✓			✓
	T14	✓	✓	✓			✓		
5	T05	✓				✓	✓	✓	
	T10	✓		✓			✓		✓
	T15	✓			✓			✓	✓

Table 184.4: Switches in the message-key unit. Each of the five switches can be set to one of P, S, T, U, W, X, Y, Z. *The effect of each switch is to activate up to three swappings (transpositions) of the control signals from the wheels* A, ..., K. *See Table 184.5 for the action of the transpositions.*

transposition number	layer	control wires affected
T01	1	A B
T02	1	C D
T03	1	E F
T04	1	G H
T05	1	J K
T06	2	B C
T07	2	D E
T08	2	F G
T09	2	H J
T10	2	A K
T11	3	A F
T12	3	B G
T13	3	C H
T14	3	D J
T15	3	E K

Table 184.5: Transpositions in the message-key unit. Transpositions that are active in layer 1 occur before layer 2, which occur before layer 3. If a transposition is active, then it swaps the signals on the designated control lines.

output	inputs									
	I	II	III	IV	V	1	3	5	7	9
I		X		X				X	X	
II			X		X		X	X		
III		X			X	X	X			
IV	X		X			X				X
V	X			X					X	X
1	X		X			X	X			
3		X		X			X	X		
5			X		X			X	X	
7	X			X					X	X
9		X			X	X				X

Table 184.6: SR logic for version C. Each output line carries the XOR *of four input lines.*

The wheel-assignment plugboard maps the control lines A, ..., K to ten lines labeled I, II, III, IV, V, 1, 3, 5, 7, and 9. These labels reflect their rôles in controlling the XOR and permutation circuitry. But before they carry out those rôles, they pass through the SR-logic unit. Each of the outputs of this unit is the XOR of four of the inputs; see Table 184.6 for the assignments of the inputs to outputs.

The key for T52c consists of the settings of the wheel-assignment plugboard (with the abbreviated labels for permutation relays), the initial wheel positions, and the message key. For example:

9, 1, I, II, 3, 7, IV, 5, III, V; 52, 19, 21, 50, 47, 24, 14, 46, 39, 7; `WUPSY`

With this key, the first line of Hamlet's soliloquy is encrypted to

`FAYUMGR+UHOEC8/8BB+STFJKAMWX/GPZJMLGUZJZZV/JJ9G`

Version CA

Version CA is identical to version C with one exception: the SR-logic circuitry is changed (see Table 184.7) to reduce some vulnerabilities.

With the same key and plaintext as in our example for version C, the ciphertext for version CA becomes

`E9VNT9BJL9GFXMMM+/CO/JG9CYNGL9RBDHPJA4BDDSJP99O`

output	inputs									
	I	II	III	IV	V	1	3	5	7	9
I		X			X	X	X			
II	X	X	X	X						
III				X			X	X		X
IV	X		X					X	X	
V			X		X		X	X		
1	X			X					X	X
3						X	X	X	X	
5	X	X			X				X	
7		X		X		X				X
9			X		X	X				X

Table 184.7: SR logic for version CA. Each output line carries the XOR *of four input lines.*

Version D

Version D has the same features as A and B, with one addition: irregular stepping of the wheels. For each wheel, a pin is tapped at a specific offset further ahead of its current position (offsets are given in Table 184.8). The outputs of those taps are carried on ten new control lines, labeled a, b, ..., k. An eleventh control line carries the middle bit (bit 3) of the previous plaintext character and is labeled z. Whether the eleventh control line is used depends on the setting of a switch on the machine, called the "klartext feature" or "KTF" ("klartext" simply means "plaintext"), and which is set to ON or OFF. Whenever a control line is active, we say that it carries a 1 or TRUE; when inactive, a 0 or FALSE. Whether a wheel advances to the next position (steps) depends on the evaluation of a logical expression involving the truth values of the control lines. What that expression is also depends on whether the KTF is in use. See Table 184.9 for the expressions for each wheel for each KTF setting.

The key of the T52 version D resembles the key for version A/B, with the addition of the KTF setting. For example, if we take the key we used for version A/B, and set KTF off, the key is

9-10, 1-6, I, II, 3-5, 4-7, IV, 2-8, III, V; 52, 19, 21, 50, 47, 24, 14, 46, 39, 7; OFF

and the first line of Hamlet's soliloquy is encrypted to

`9983TW+WDHYV8V+8WVOJCOZSERDSU++QBXPVAR3LF+4PD8F`

When the KTF feature is used and the key is

9-10, 1-6, I, II, 3-5, 4-7, IV, 2-8, III, V; 52, 19, 21, 50, 47, 24, 14, 46, 39, 7; ON

the ciphertext is

`9989NAK43O4GQYQ8CMZVZUQZOHTAB+3FVYOR9+3LIU9CZVB`

wheel	stepping-pin offset
A	25
B	24
C	23
D	23
E	22
F	22
G	20
H	20
J	18
K	16

Table 184.8: Offsets for the pins controlling irregular stepping. The pin whose signal helps to control irregular stepping is the given number of places beyond the current position of the wheel (modulo the size of the wheel).

wheel	logical expression when KTF=OFF	logical expression when KTF=ON
A	(NOT e) OR (NOT f)	b OR c OR z
B		(NOT c) OR d OR z
C		(NOT d) OR e
D		(NOT e) OR (NOT f)
E	f OR (NOT g)	f OR (NOT g) OR (NOT z)
F	g OR h	g OR h OR (NOT z)
G	(NOT h) OR (NOT j)	
H	j OR (NOT k)	
J	(NOT a) OR k	
K	(NOT d) OR e	a OR (NOT b)

Table 184.9: Logic of the irregular stepping in versions D and E. A wheel steps if the required logical expression evaluates to 1=TRUE*. The logical expression depends on whether the klartext feature is* ON *or* OFF*. The variables in the logical expressions are the outputs of the pin from the corresponding wheel (“*a*” for wheel* A*, etc.) at an offset given in Table 184.9. The variable* z *is the middle bit (bit 3) of the previous plaintext character. Variables carry the values* 0=FALSE *or* 1=TRUE*. The* NOT *operation flips a bit (*0↔1*). See Table 184.2 for a definition of the* OR *operation.*

Version E

Version E resembles versions C and CA, with these changes:

- The message-key unit is not present.
- The SR logic is changed. See Table 184.10.
- Irregular wheel stepping (including the KTF) is added in the same way as in version D.

The key is specified as it was for versions C and CA, with the modification that the message key is replaced by the KTF setting. For example, using the key

9, 1, I, II, 3, 7, IV, 5, III, V; 52, 19, 21, 50, 47, 24, 14, 46, 39, 7; OFF

we encrypt the first line of Hamlet's soliloquy to

`SVPWX9PT3ULW3TFMKE3KC4N9CPNESJSJXVTVJ/SGOOJGIPI`

With the key

9, 1, I, II, 3, 7, IV, 5, III, V; 52, 19, 21, 50, 47, 24, 14, 46, 39, 7; ON

we get

`SVP4AV9TZWPMZRTEMQQQBYI3WAVQ+NTECMX/IHSX8/WCQTR`

output	inputs									
	I	II	III	IV	V	1	3	5	7	9
I		X		X				X	X	
II					X		X	X		X
III	X			X		X	X			
IV	X		X		X					X
V	X			X					X	X
1			X		X	X	X			
3	X	X	X	X						
5			X		X			X	X	
7		X				X	X	X		
9		X				X			X	X

Table 184.10: SR logic for version E. Each output line carries the XOR *of four input lines.*

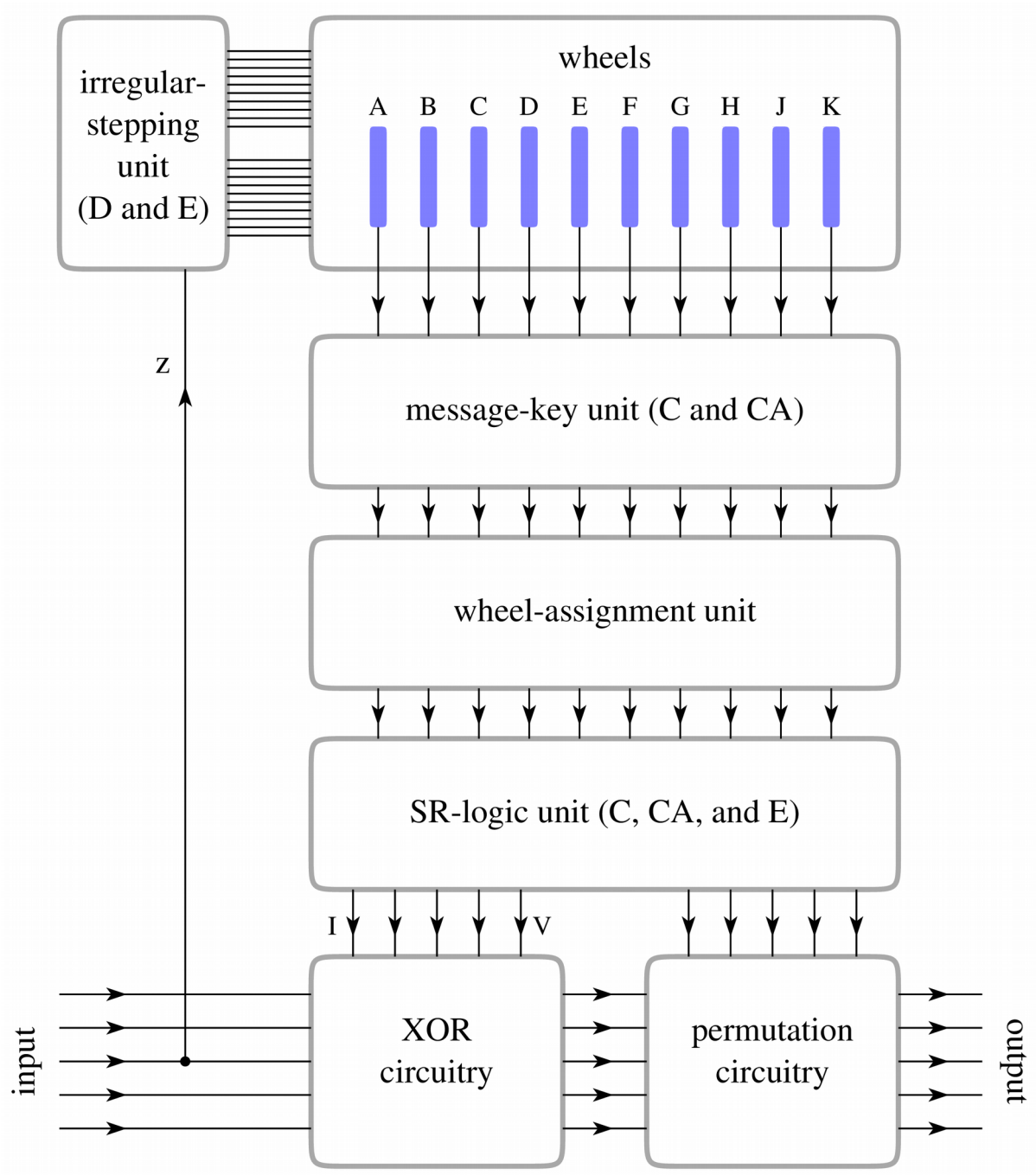


Figure 184.5: Overview of the T52. Each module is labeled according to which versions of the machine it appears in. All versions have the ten wheels. From them come ten control lines, A, ..., K. In versions C and CA, the control wires pass through the message-key unit which shuffles their signals; in other versions the control lines run directly to the wheel-assignment module, which is present in all models and which also shuffles the signals. In versions C, CA, and E, the control wires enter the SR-logic module which modify the signals in such a way that each output depends on four of the inputs. Five control lines actuate XOR *operations in the XOR module, while the other five control the permutation module. In versions D and E, there is an irregular-stepping-control module. It takes signals from the wheels and plaintext to determine which wheels advance.*

Reading and references

Frode Weierud, “Bletchley Park’s Sturgeon, the Fish That Laid No Eggs,” *The Rutherford Journal: The New Zealand Journal for the History and Philosophy of Science and Technology* 1 (2005-2006), https://www.rutherfordjournal.org/article010106.html (Figure 2 of this document has errors)

George Lasry, “T52 Functional Description.” This document can be found in the supplementary materials for some challenges on MysteryTwister, in this zip file: https://mysterytwister.org/media/challenges/add/mtc3-lasry-T52-add.zip

George Lasry, “T52 Cryptanalysis.” This document can be found in the supplementary materials for some challenges on MysteryTwister, in this zip file: https://mysterytwister.org/media/challenges/add/mtc3-lasry-T52-add.zip

Donald W. Davies, “The Early Models of the Siemens and Halske T52 Cipher Machine,” *Cryptologia* 7:3 (1983) 235-253, https://doi.org/10.1080/0161-118391857964

Donald W. Davies, “The Siemens and Halske T52e Cipher Machine,” *Cryptologia* 6:4 (1982) 289-308, https://doi.org/10.1080/0161-118291857118

Donald W. Davies, “New Information on the History of the Siemens and Halske T52 Cipher Machines,” *Cryptologia* 18:2 (1994) 141-146, https://doi.org/10.1080/0161-119491882801

Wolfgang W. Mache, “The Geheimschreiber,” *Cryptologia* 10:4 (1986) 230-242, https://doi.org/10.1080/0161-118691861065

Wolfgang W. Mache, “The Siemens Cipher Teletype in the History of Telecommunications,” *Cryptologia* 13:2 (1989) 97-117, https://doi.org/10.1080/0161-118991863817

David Kahn, “The Geheimschreiber,” *Cryptologia* 3:4 (1979) 210-214, https://doi.org/10.1080/0161-117991854089

Crypto Museum, “T-52 Geheimschreiber: Teleprinter cipher machine (STURGEON),” https://www.cryptomuseum.com/crypto/siemens/t52/index.htm

Programming tasks

1. Simulate the T52. Include a switch to determine the version (A/B, C, CA, D, E). For versions A, B, and D, include a function that checks that a permutation-circuit wiring is acceptable (there may be ways of doing this without using a look-up table).

2. Develop a brute-force attack for a situation in which the wheel assignments are known and some of the wheel positions are known. It may be faster to use the index of coincidence to reject most decryptions, and the fitness as a second stage of verification. Using a faster, lower-level language (such as C) would also be a benefit.

3. Develop a hill-climbing attack that maximizes the index of coincidence if the wheel assignments are known but their positions are not. This is not likely to work for versions D and E.

Exercises

1. Decrypt this ciphertext with version A/B and the given key.

   1-4, 2-5, IV, I, II, 3-9, 6-8, 7-10, III, V; 43, 45, 15, 67, 50, 35, 22, 29, 39, 42


   ```
   RJLQHMILUAMODRT8LX8IYVFYW4WPB9WUW/VWCNKPXW/R3F+AA3F9EZGLNJABW8KP
   IT/IRP3NKHEACWGBJ4W9DN8/NEH/T+XHT+WKPURIDTH8YV++JHACHXUMZTHUIAGI
   OPCTPTFOQDWK3ZSSAF883AEGI+RIT/XWRTNEM4DN9KBW+IJCMXO9RKBUTANRYZ4D
   YV+W3XBBUPM8EOS/HG9DYE+RE+J/WHRV8VKA/W8MQ+89Y8PR9RHJ9QDQVUY3EBTZ
   XJLXZN4WZYSOMHSV3NOD+WU4HPR/UWWOTQS+YEVPJRUFUT84ZU4HTCUF8LE/RWEJ
   3WCY4WULXTB/MLE3EG
   ```


2. Decrypt this ciphertext with version C and the given key.

   II, 5, V, 1, I, III, 3, 7, 9, IV; 53, 65, 32, 5, 42, 56, 38, 42, 29, 38; WXPSY


   ```
   4S8/9ZXBINQY+DO3ZEOYBFXAOYDRG9KCTRD9PFJSIJMU+EIDWJUQZYKKW+S8BV3E
   +FNYWXOMUUBZVSYVSKSJLHLY+ZRDPDMWDFWLIBPOC4CNXME9IIMO+PBAACFS+GU3
   YSSG9NEYPFS8QB/DYFNM9KV+3MYYH3MR4R3B+PUBFDKEVBFOGVZFQ4/8QTRPEMRJ
   OROSCM4+TG4ANYMKAQ9KSTMDRXCXFWFSY/TMXG8V+/U3C+T9H3DHZVHW3JHD343R
   ILJLCR4RYETNSTW9VT9U3BPXDA/NGF
   ```


3. Decrypt this ciphertext with version CA and the given key.

   1, 5, V, III, I, IV, II, 9, 3, 7; 53, 17, 56, 31, 65, 30, 44, 57, 24, 32; PZSXZ


   ```
   9O8+KIVIASQEWPBPARL4VCGI9FW4/YCNZK/RCMSOX+ZVCGDFU4O+/FMDLSU3W/8D
   EWK88CG/G94FD8NRDVLCFXYO/EOPEII3CESZMNNIHU83TMUZ//AIMSBMPYONZPVJ
   9PWTFZDEJIOSTY+PEANV+PGVDKOYK/CMT9HE8+CHH//CKSLQVGJOH+BQDP+BNVAF
   MK8LWVA3DLP/YI9/O93HYYVUC4S9E4KECDAQXURZUOCFCE9SR9WKHACIF3DMNCH3
   X9Z+C/EN44L4GW8
   ```


4. Decrypt this ciphertext with version D and the given key.

   2-5, 7-10, II, 3-6, III, V, 4-8, IV, I, 1-9; 46, 7, 30, 52, 51, 10, 60, 33, 21, 8; OFF


   ```
   R/XJY3AB+MYJ/344VCJHW8GHTF8QFTVAOXCG9MDH8FMR9+YWBQOPYEFV3D94BUQT
   FB8B394FS33DMBC4+KAU93UP+UAQ4QTUMFGFURDIYT+4ISLVKWVKTELTNIME3L/H
   Z8SXJNHOMOQFIITEHWG8ZPD+D4N/TLJ+/FRPO8MI/J4I8TB+URQBPV+4V3STNPFO
   LE/L+WH/3NZ9HK8WPOWXUS9SODIOPR3+SCNZ9H3OM+WJ8CCGA4LBDZU4+38SKATI
   IKE4FK4BOEST9HVQ4VKBGAD3DW8MX/UMTIOS/PUURDBZGEMG+NA/XQC8/MHLGIUR
   OAXSRBZ+RJ3ADO/M/SPRY4CKGODFUASTVE+LAJOMHRVPBVMDX494+/PCSZINQJCM
   EJKDCQT/RCRICUG9XBSXEQO4KUMMKNMX4+PPYV/QPYYPI9LFZ+OMQWSJP9VZBP8I
   ASZ+/9IUZZ+H9+W8MGMQPH4YX4ROK4NR4+SAJLRQXZKQTZCGJCZFN+APELPAL8N8
   IS
   ```


5. Decrypt this ciphertext with version E and the given key.

   III, 9, IV, 5, 1, II, I, 7, 3, V; 56, 32, 59, 59, 11, 27, 29, 10, 52, 41; ON

```
RAZIGAH+ESFJD4G/S8HOWUSQ9ZSKDZOR9WATHMEAV/RBDQDJWYYBRCDUHSOTS
YNVT4/HJWWMIVLDFIPXKOPMZD9CHSRINYSDXYHVBRKUHCRQZQQL/VDOGUBSA+
BHKDWRP8EAL8KJXYA8/KSJD9KRP4HPH9EPE/N9ZLGTRT9GEG+NSNPY39KZVNP
9ABPTDS9GCEEKVFIQW
```

6. The following ciphertext was encrypted with a T52a/b. The wheel assignments and half of the initial wheel positions are known. Brute-force the remaining wheel positions and decrypt the text.

   wheel assignments: A IV, B I, C 6-9, D 8-10, E 2-4, F III, G 1-5, H V, J 3-7, K II
   wheel positions: A 11, B 27, C 4, D 15, E 48

```
+3RAYZQMHNJWFF8N8OFESRUYUTM/4VKONQK8RCTNA4TTMTQ3TYZKMDEFWK3/NNKI
MCICDEK+OWDVHOE9DERPYV3DHS4+LC/MR4SCJBWUISWKMZM8TCOX/E+FJHRCFM8U
XTNTTPJF49QOEZISIJWNHOGIC8CAWOUCD3/YFW84LENYTNI+H9BBOLJBQRQLDAVP
N+A/BCMO4WTJ4PVSWPP4MS8UJVYP3LLPDJNGEBUZ9AT3HRHO/OCQ8KGS+UILYYYY
+CKYCEKGHELMAFYSJTVH/ZWDW8EOHB+IE3YS8LAHSETWP3CJAKYKP9O9M9ZCOR
```

   This exercise is similar to a challenge from MysteryTwister (https://mysterytwister.org), which can be found at https://mysterytwister.org/media/challenges/pdf/mtc3-lasry-14-T52-01-en.pdf.

7. The following ciphertext was encrypted with a T52c. The wheel assignments and half of the initial wheel positions are known, as is the message key. Brute-force the remaining wheel positions and decrypt the text.

   wheel assignments: A 7, B 9, C IV, D II, E V, F 1, G III, H 5, J I, K 3
   wheel positions: A 33, B 52, C 34, D 4, E 9
   message key: ZXYYS

```
G/PBVQSM9YLM+E4HKUXFPN4+GGCVKNRMDBRX3JLKI4+JDEGF+DT4RO9RDS8NWZLY
TZG+LM44V9TVXYMZTYW99LEHT3NVOJNARQ9T9FIO/XKLS8KM98OIRPMM9+GH/WP8
+Q/GEWQU9XXWJDCQ83W/3ZNZM/YM+AFWLY8JXH+YBMMQ4QHX+PERX+FNMSF/F9JZ
C/X8UDXNB+J8CSSMEIYLB4+TMXZPBG+LAOHSPFENHPA3/CZ4I9LSWDFJOBRJEA8C
8FPGROMGWS4SGCD9YLVTJGNG+ABFGXFVI+9KIC/M8EMQBZWSLP99STHU3AYC/VCB
YWUGNO9M/MUFQGZIK9Q99+OTU8HEFDA+MWJD9FQU9+CIBNEUKXXP34POB3CMD4Y/
UDWBIPUNLCFPNYT/AQPOQ9IKI98JKF9DCR+9/OES89MCEYP+SCNEQIR8YHOLYGJM
NJB+XNHV4TV4DLKSS8PTNOM/DJTZESIKWYJOF3U4SGFIO8M4VUYOSPF+IXXZXBW3
RQS38
```

   This exercise is similar to a challenge from MysteryTwister (https://mysterytwister.org), which can be found at https://mysterytwister.org/media/challenges/pdf/mtc3-lasry-15-T52-02-en.pdf.

8. The following ciphertext was encrypted with a T52d. The wheel assignments and half of the initial wheel positions are known, as is the KTF setting. Brute-force the remaining wheel positions and decrypt the text.

   wheel assignments: A II, B I, C 4-6, D 9-10, E V, F 3-7, G 5-8, H III, J 1-2, K IV
   wheel positions: A 1, B 43, C 6, D 62, E 2
   KTF : ON

```
B9JCH4CCDNHCJ83JXHCNG+GM+NLYIBFRNOV9BEWLAB+QDOT3DHJQUHEXHJJ+/HYO
KPZ3SZP9C+X3M98J9JOWB/GZLMEUXVDWYNF8ZC9OVYYCLALAQVS/8V9XVUKDEWMQ
```

```
JVT3O/BXSDRZXCN/GSH3E+DPALBSVCSUV+XP9N9APFJL3LGDGP+B4++SQLUJGFMW
3ZKGIHWRWRN+TQTM9WNGSM93UFB+ZVHOBUBA3YMUJGBAPC9IGYGMJZ+WFXR4E9ZP
POYQ9CTPL9XGOIESCGGX+XRMBYWTVPYQB3R4MYXJFZTGN8TJ/W/UNNY/IUY98YAS
KISW/CPWC/RYAWTKE3VFWG/PSUZ/4V4AR9CMQLFXPJ8MULXB4H8NTVKYU+JXZDT8
ZQR+8CQQRODGJJH/KFTTW/DLQF+FCMLJTTPRUTPLYZQDA+EQ4R/3DCA+IY34LD8H
+EBYSYOSHRSP39
```

This exercise is similar to a challenge from MysteryTwister (https://mysterytwister.org), which can be found at https://mysterytwister.org/media/challenges/pdf/mtc3-lasry-16-T52-03-en.pdf.

9. Below is a ciphertext in Bletchley Baudot notation. It was encrypted with a T52 version A or B. Part of the key is known:

   wheel assignments: A 4-9, B III, C IV, D 3-6, E V, F I, G II, H 1-7, J 2-8, K 5-10

   Use your hill-climbing attack to find the wheel positions and decrypt the text.

```
XOYDGSTFC9EN8QVGMADUIHYXBQARRML+/YDQP8KCR93SF+G8WQ+EQKU/33V8XHIK
HUU9F3YQWXJLJO/BKQ9M+SOETV/FA8L83CSR//CIIW+E8XNNAPC8J+LBQLJ/8UTR
HFZOB3QT/PA+M4V/K3FIQBG8/MLEJ8R/4FZLEOIX9+3LF/F8VVMBP3GQG4NPRTZP
ZMDDKGDWHL3F+LLCBPSBJFC/B8TYB4CZ/NTBHRMJO/JDXAEJUK/UIT8CO4MN+ILK
BNU/NFDMIIJ/XKJYY3YKCN8YIFKVEE9UUFHXI9FNNQ++FTOKFUMYD/B8DHCICAYQ
NJLZESKWQJ+MRY8GRMWWDJESVTVC/PFMMMZFO8J3/ORSAIG9LDJZBOMDEIX3QMRI
83IF+TLYWUELHMAGOSPMMIT3YJ/XTQQYGN4G8HTBNWFGHNBR9JH+FUC3PXFHBAEV
QQ3ITBMOHY9NGWQXPTNKB/GCL3XRSCUR3HR4X8QPP4BTD3VI4SATOJNJIKNAUHF/
FSJZIO+ETSWEMF84M3DMIBESKD8LY88BARJNMBAX8HOYYAFQYBK9GQHXPCGXILRL
34SCJ9REJRKZQRGIYRYRQEK8RPPQUKNKGALWKM+PAOWGJ4NK9GWKPDDYKX3SMCBC
OL99MRTTGWZ/KSZVZGGGQX9TDT+BBW4PABVFFKFIMLMZVQY+F/GHA+YC+8J3YS+M
RL/XPOJA+8OJWRBZ/9SQ/XG8QGLRHAMI/GANR9BQDDGJ3RCXR+9DKE4UMPT8HXAM
SMNSX/FZTOSI33QFA+J+QMVAI4ZX+H3GO9YTLRF/PY3ZURW+DRLGA3UG/+IOHTMG
BYLL9LCOJI3M/QKJFHAQEDT8RVTCLKISTVIGF3XAIBBNMSRCDWBG9FXJDDKWVDTQ
/98VQTQT3N4M8IGYXKN+B3CXEH+EWK8TAKIJU9+3DVOPVZEOIYBARD
```

This exercise is similar to a challenge from MysteryTwister (https://mysterytwister.org), which can be found at https://mysterytwister.org/media/challenges/pdf/mtc3-lasry-17-T52-04-en.pdf.

10. The broken T52a: Imagine a T52 version A that is broken so that its transposition relays stick and the permutation that it uses is set by the initial configuration of the machine and does not change as a text is encrypted. Find a way to break this ciphertext that was produced by the broken machine:

```
/PGU9VJYZJROE3T9398VEE8XJPT+ICVESQ44XJIDDB8EKZCUEBLS+JRZ/T3HGWU+
8NNUDKUQLPOMXUS/AOOSBRWW3W3PKFDUQQQE+S3CZW9FFE4SEOHWBGLQO9XOOHMQ
LMRNLKQDFEORUDL8W/DGIWZVJ98GMUNBQO4MX8B3F+8RD9CCQHUKKI3CNZ+RKVFA
+AGFULBDCOD/GZ/3AXY9/WG+TVM/TLMI3SD9HDN9FWX4DT/W9KLFDCWOPAITUL/4
B+CXQBSMB+3AWASW+ES4JBYQBXKMTDWOF+/+PJYE+MS84FSJ4GW84YLRVOM+4PCZ
D8UAXP8XXFFORMRMJHLVSIMSXQSNCPQFCYGCX+4QF/W9+BDWQW+ZVQRULCRPOZGQ
RE/4EPZRLUWJ8IIK8QU3ZKSKN/H3WSGG98EGT4LDH4Q4BF+JE3I+4C9MZ9SZ9NP3
E+LT88JYWYODU9EH+9CZAWHODMLWV/JWFQPEPBAZBG+O8QYKN4//YKJWTLYI+NTA
S9IS4F+HE+OP8COJ8IDI+8JQ8NYFPX+AZCVM+Q/OPBZ+8FLJWKPVWPTIU3X9D/V8
OI9WO4S/YE43Y93M+MWDXT+ZFBLQJJ44J/+FP9NSQ3W8UJY+FZ4+FL9CO+E88MSQ
+JLVDOJGJTKJ+M8SAIWPOCXAYCLVDBK8O/VG9LC8OG/PDAH+WN/3P8HEY8+KHI/A
```

8Y3NEWBUAVG+UKXHU9FIPPBAIOURYFWUANA+BX3IWOOUKZCPYAHMKTG3OJOIR3L+
BQGLE+XDIRQTC3JEYHZ9RAHW4CM3PJ4DQYSV8XUE3X8MEFMW9NQSNBFMTDS4UOLK
VEKKNXQ8A/IQXHKCFTDSQ/ANREIVIBQA+CK3EQMX4QFEZYZ8ZT8EYSW9/WSQOJRO
Q8PQAX3MSA9TDZBAUWLNRJIX4RBHPK/EJLUVBBV8U3EW9XFUNVS83SCLNP8FAO4Q
QSNDEQW4QNYCERWF8AAYUT+HQ9RVPY3OO49VBW8YTRYAIBKJMPJUAEHPOXQPC/VV
K3TQA/UXPHJHT9GN83+HXTDUHQPVJ84RB+VMZBQ+O9QXAEODXL/QSG/PLEIZTWHM
+NIMJTJ8A3VIRJ8/XFJ98ZRSYJI+4R//ZCAYO83ADNXQZF3EINPIKZG/WM9MARBE
Q+IEYGRW+LL4U+FKZHB9OWAJC9F9+KESKZ89MKHWEWGZ+KHAWDVNENPEIXQERIIY
QQKCZEBJELLEBXWCDORBEMK3WBQZTAPXLGZTBVZRWFYJPEGAD/PDXLHDGKBEMIAD
YHTU/+JDTEGB4MQY+JJHVBCQYMDLCUFLQLNV/LHVNHYW3A++ANMIXZAMRQBANLL+
FMGNQ8VQGOV88SR4G+U9UE+UES/44XY8JXYK3G9FRVNSUU4SEWNSKDVMJLQS4NSE
CKWYK/OV

Hint: ILUVW, FRPPXWH WKH SHUPXWDWLRQ SDVW WKH ARULQJ. WKHQ IRU HDFK GDWD OLQH WUB DOO SRVVLEOH ZKHHOV DQG VWDUWLQJ SRVLWLRQV DQG FKRRVH WKH RQH WKDW PDALPLCHV WKH LQGHA RI FRLQFLGHQFH. BRX PDB KDYH WR GR WZR OLQHV DW RQFH LI WKHUH DUH PRUH WKDQ RQH JRRG FDQGLGDWHV IRU KLJKHVW LRF.

Unit 185
Lorenz SZ40 and SZ42

The *Lorenz SZ40* and *SZ42* formed a family of teleprinter cipher machines used by Germany in World War II. They were so named as an abbreviation of *Schlüssel-Zusatz* (“cipher add-on”). British cryptanalysts nicknamed them *Tunny*, which is their way of saying “tuna fish.” Like the T52, SZ40/42 used wheels with pins to modify the bits of baudot-encoded messages. Unlike T52, the pins of the wheels in SZ40/42 could be added or removed, and only XOR operations, but not permutations, were used. Look back at Table 181.4 for a definition of the XOR operation. A fun property of XOR is that

$$A \oplus A = 0$$

(Equation 182.5) for any bit (0 or 1) A. Therefore, if we use XOR to encrypt, then the same operation is used to decrypt. If we write P for the plaintext, C for the ciphertext, K for the key, and think of “$\oplus$” as acting on all the bits in parallel,

$$C = P \oplus K$$

$$C \oplus K = (P \oplus K) \oplus K = P \oplus (K \oplus K) = P \oplus 0 = P$$

(Look back at Equation 182.6.) This property makes life easier for German cipher clerks. The rest of this unit will be spent on describing the various members of this family of machines.

Figure 185.1: Lorenz SZ40 on display at the U.S. National Cryptologic Museum. Photo from U.S. government.

There was an initial model, the “old SZ40,” that was more of a prototype than anything else. It had a small production run and was used for only a short time. It has ten wheels, and all of them stepped after each letter was enciphered. They have fixed pins. They are grouped into five sets of two, so that the signal from two wheel affects one bit of the baudot data, i.e., each bit is XORed with the output of two wheels:

$$b_i \rightarrow b_i \oplus X_i \oplus Y_i$$

(i = 1, ..., 5). By giving the wheels sizes that are coprime, the period of the machine is lengthened.

The proper SZ40 has twelve wheels with variable pins. Pins can be flipped into place (“active,” denoted by × or 1) or out of place (“inactive,” denoted by · or 0). Like the old version, the proper SZ40 XORs the output of two wheels to each bit of data. However, only five of those ten wheels step after each letter is enciphered. The additional two wheel control how the remaining five wheels step.

Five of the wheels are called *Spaltencäsar* or *chi* (χ) wheels. The output of the first one is XORed into the first bit of the plaintext character, the output of the second into the second bit, etc. Another five of the wheels are called *Springcäsar* or *psi* (ψ) wheels. Again, the output of the first of these is XORed into the first bit of the data, etc., so that a bit is altered thus:

$$b_i \rightarrow b_i \oplus \chi_i \oplus \psi_i$$

The remaining two wheels are called *Vorgelege* or *mu* (μ) or *motor-control* wheels. More about them soon. Each of these twelve wheels has a different number of pins around its rim. See Table 185.1 for those numbers. A pin can be in active or inactive position. Only if a pin is in an active position can it conduct a signal out from the wheel. The wheels have numbers on their rims, like in other machines we have seen, to indicate which pin is being probed at the moment.

Stepping of the wheels occurs after each character is enciphered. All of the chi wheels advance by one position. If the output of μ_2 is a positive signal, then all of the psi wheels advance by one. If the output of μ_1 is a positive signal, then μ_2 steps. Finally, μ_1 steps every time. Got it? Take a look at Figure 185.2 for a sketch of the encryption process.

We need to discuss rules imposed by the creators of the Lorenz machines for the setting of active and inactive pins on the wheels. For each wheel, there is a prescribed number or range for the number of active pins; see Table 185.1. For the chi wheels, there can be no more than five active pins in a row, or five inactive in a row. Furthermore, the number of segments (sets of pins with the same setting all in a row) is prescribed also. For the psi wheels, the number of segments is determined by the number of active pins on μ_2; see Table 185.2. For μ_1 the constraints are that there can be no more than five inactive pins in a row, and no more than fifteen active pins in a row. For μ_2, there can be at most five inactive or six active pins in a row.

wheel	pins	active pins	segments
χ_1	41	20 or 21	20
χ_2	31	15 or 16	16
χ_3	29	14 or 15	14
χ_4	26	13	12 or 14
χ_5	23	11 or 12	12
μ_1	61	30-36, 38-50	
μ_2	37	9-23	
ψ_1	43	22	26-34
ψ_2	47	24	28-38
ψ_3	51	26	32-42
ψ_4	53	27	32-48
ψ_5	59	30	36-48

Table 185.1: Wheel sizes and key restrictions for the Lorenz SZ40 and SZ42. A "segment" is a group of contiguous pins in the same state. The number of allowed segments on chi wheels depends on the number of active pins on μ_2.

active pins on μ_2	allowed segments on psi wheels				
	ψ_1	ψ_2	ψ_3	ψ_4	ψ_5
9	34	38	42	42	48
10	34	36 or 38	40	42	46
11	34	36	40	40	46
12	32	36	38	40	44
13	32	34	38	40	44
14	32	34	38	38	42
15	30	34	36	38	42
16	30	32	36	38	42
17	30	32	34 or 36	36	40
18	28	32	34	36	40
19	28	32	34	36	38
20	28	30	32 or 34	34	38
21	28	30	32	34	38
22	26	30	32	34	38
23	26	28	32	32	36

Table 185.2: Key restrictions on the psi wheels in LZ40/42.

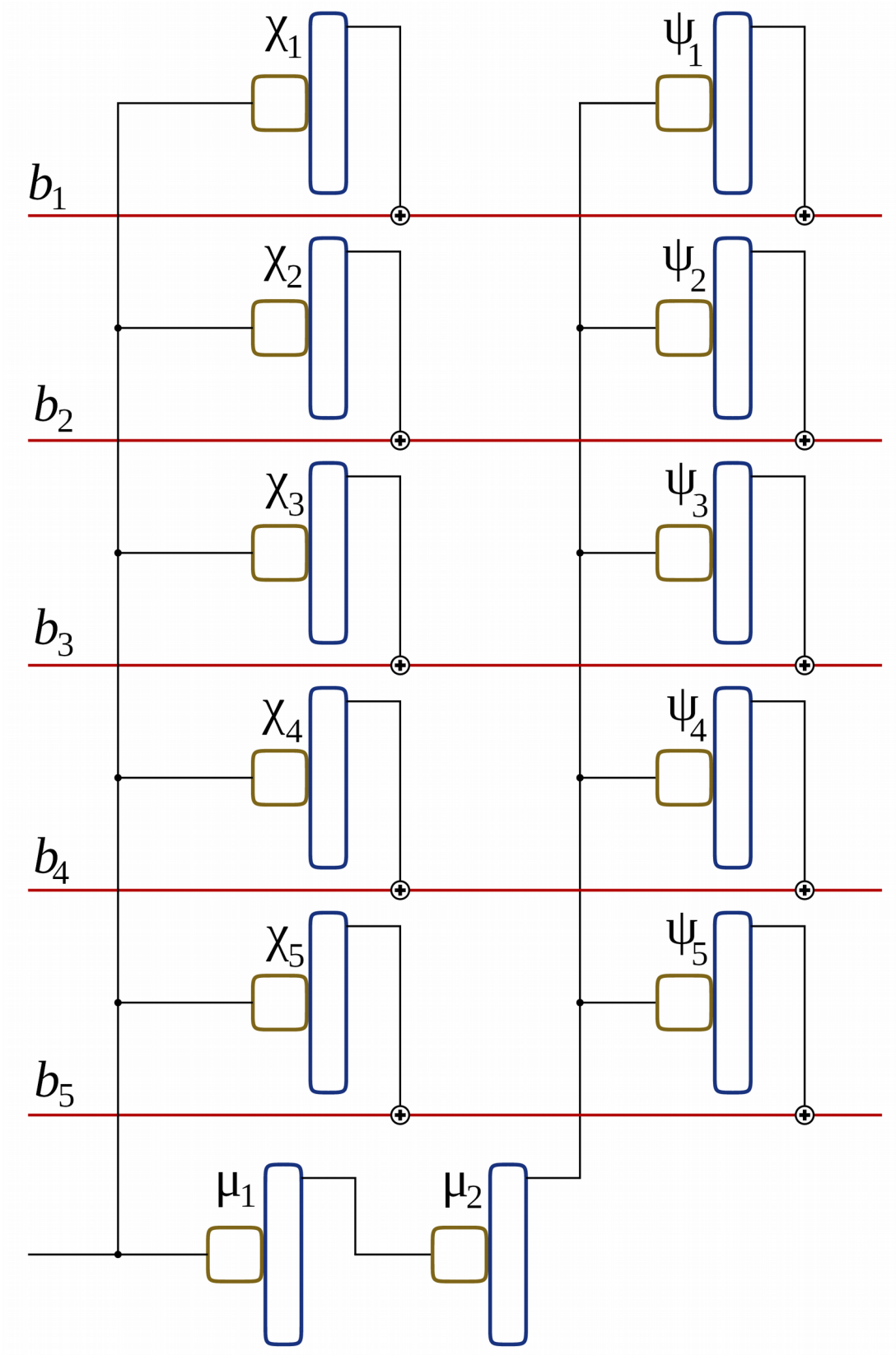


Figure 185.2: Simplified diagram of SZ40.

The key for the machine consists of the pin layout and starting position (counting from 1) of each of the twelve wheels. For example, this is a valid key:

```
χ1:   35   ·····×××··×·××××·××··×··×·×·×××····××××·×
χ2:   22   ××××···××·×·×·××··×··×××····×··
χ3:   17   ··××····××··×·××××·××·×·××···
χ4:    9   ××××·××·×·×·····×···××·××·
χ5:    5   ···×××·×·×··×·×××××·×··

ψ1:   32   ·×·×····××·×··×·×·×××·××··××·××·×·×·××·×··×
ψ2:   41   ·××··×·×·×·×·×··×·····××·×·××××·×··×·×·×××·×·××
ψ3:    3   ×·×··×·×××·×·×·×·×····×·×·××××·××·×·××·×··×·××··×··
ψ4:   10   ·×·×·×××··×××···×·×·×·×···×·×·××·×·××·××·×·××··×··×·×
ψ5:   23   ×·××·×··×·×·×·×·×·×××··×·×·××·××·×··×·×××·×·×···××··×···××·

μ1:   59   ··××···×··×××··××··×·××××·····××··×··××··××·×·×××····×·××××××
μ2:   12   ××××··×····×·×···×··×××·×··×·×···××··
```

Notice that for χ_3 and χ_5 we have the correct numbers of segments, due to wrapping around the wheel.

Suppose we want to encipher a message with the example key above. Start with a plaintext:

```
THIS MESSAGE WAS ENCRYPTED WITH LORENZ SZ40
```

The zeroth step is to encode the message in Baudot, and list the bits for each character:

```
THIS9MESSAGE9WAS9ENCRYPTED9WITH9LORENZ9SZ+RP8
00010011110101110100010011010000000101011100 1
00100000011001100001101000011000101000000111 1
01111101100010011011011000101011000010110001 1
00000100001000000011100001000000011010000110 1
11000100001001000000011100010110110001001101 1
```

The machine is in the state χ = 01111, ψ = 01100, μ = 11. To encipher the first letter, we xor the bits of T with the bits of χ and ψ:

```
T    0 0 0 0 1
χ    0 1 1 1 1
ψ    0 1 1 0 0
---------------
3    0 0 0 1 0
```

Next, the chi wheels advance to 36, 23, 18, 10, 6. Since μ_2 outputs 1, the psi wheels also advance, to 33, 42, 4, 44, 24. Since μ_1 outputs 1, μ_2 advances to 13. Lastly, μ_1 advances to 60. The encipherment of the next letter is

```
H    0 0 1 0 1
χ    1 1 1 0 1
ψ    1 1 0 1 1
---------------
O    0 0 0 1 1
```

Now, all chi wheels advance, as usual, but because μ_2 is at position 13 which outputs 0, the psi wheels do *not* advance. Since μ_1 outputs 1 at its current position, μ_2 advances. Then finally μ_1 may advance. The next letter is enciphered:

I	0 1 1 0 0
χ	1 1 0 1 0
ψ	1 1 0 1 1
P	0 1 1 0 1

The full ciphertext is

3OPOD9NKZK/+/3SVKYRH8OMRONZGHIYSNXGMRFPEFFXDN

In Unit 151 we mentioned that we must never reuse the key in a one-time pad. The keystream of the Lorenz SZ40 ($\chi \oplus \psi$) is like the key of an OTP. It is not truly random, but appears to be. But the problem arises when two messages are enciphered with the same key. Suppose we have two such messages, P_1 and P_2. Encipher them by XORing bit-by-bit with the *keystream K*:

$$C_1 = P_1 \oplus K$$

$$C_2 = P_2 \oplus K$$

But when an adversary adds the two ciphertexts together,

$$C_1 \oplus C_2 = (P_1 \oplus K) \oplus (P_2 \oplus K) = P_1 \oplus P_2 \oplus K \oplus K = P_1 \oplus P_2 \oplus 0 = P_1 \oplus P_2$$

The result does not include the key at all! With only a few such messages, the plaintexts can be disentangled, and the keystream can be recovered. Such messages are called *messages in depth*, and they must be avoided for security.

To combat the problem of lazy cipher clerks reusing keys and producing messages in depth, new models, the SZ42A and SZ42B, were introduced with features to modify the control of the stepping of psi wheels, and some of them were dependent on the plaintext. By using plaintext information in the encryption process, different messages are encrypted to different ciphertexts, even with the same key. There are four options (see Table 185.3), two of which incorporate a bit from the plaintext. These are called *limitations* and they cause the psi wheels to advance even when the output of μ_2 is 0. All of them involve the output of χ_2 from the encipherment of the previous character. This bit is called “chi2 one back” and is notated with an overline. In addition, one or two bits can be incorporated: the output of ψ_1 one character previous (“psi1 one back”) and/or the fifth bit of the plaintext character two letters back (“P5 two back”) (denoted with a double overline). The one, two, or three bits are XORed together and then inverted (XORed with 1). If the result is 1, then the psi wheels advance. If the result is 0, then the output of μ_2 determines whether they advance. So that you can check your emulation of the machine, here are the encipherments of our example message with each of the four limitations added to the key. With chi2 1 back,

3OPODW9TCT9/E3KHGKH4W/W/LBYVSQMWDWWYW+JLVZVRK

With psi1 1 back,

```
3OPOVN9TCJ9/E3KHS/JY9RBG3IJ/CXNHOJKDES9W8C/GT
```

With P5 two back,

```
3OPOVNWTCT9/E3KHS/JY9RBGLIJ/T3W+TLCYW+JLZCVQI
```

And with psi1 one back P5 two back,

```
3OPODW9UBU89W+NDO89RWXWT3MH8CXNHOJK3MRYMTT/HZ
```

There was a third version of the SZ42. The SZ42C used output of the chi wheels to control the stepping of the psi wheels. The two motor-control wheels were removed. This model never made it to production.

name	notation / limit bit
chi2 one back	$1 \oplus \overline{\chi}_2$
psi1 one back	$1 \oplus \overline{\chi}_2 \oplus \overline{\psi}_1$
P5 two back	$1 \oplus \overline{\chi}_2 \oplus \overline{P}_5$
psi1 one back P5 two back	$1 \oplus \overline{\chi}_2 \oplus \overline{\psi}_1 \oplus \overline{P}_5$

Table 185.3: Limitations on stepping of the psi wheels in Lorenz SZ42.

Reading and references

Orr Huettenhain and Dr. Fricke, OKW/Chi Cryptanalytic Research on Enigma, Hagelin, and Cipher Teleprinter Machines, TICOM report I-45, 1945, https://drive.google.com/file/d/0B7sNVKDp-yiJOWYxZWFmNDgtODUyMS00Y2FiLThkNWItYmQ5N2JmMzEyMzIz

Irving John Good, Donald Michie, and Geoffrey Timms, *Breaking Teleprinter Ciphers at Bletchley Park: General Report on Tunny with Emphasis on Statistical Methods (1945)*, Wiley, 2015, ISBN 9780470465899 and 9781119061601, https://doi.org/10.1002/9781119061601

George Lasry, Solving a Tunny Challenge with Computerized “Testery” Methods, Proceedings of the 3rd International Conference on Historical Cryptology, HistoCrypt 2020, 96-105, https://www.ep.liu.se/ecp/171/013/ecp2020_171_013.pdf

Wolfgang Mache, “Geheimschreiber,” *Cryptologia* 10:4 (1986) 230-242, https://doi.org/10.1080/0161-118691861065

Programming tasks

1. Make a function that checks the validity of a key. This is a project in itself.

2. Implement the Lorenz SZ40.

3. Add the stepping-control features of SZ42A and SZ42B to your implementation.

Exercises

1. Decipher this text that was encrypted with SZ40. Use the key from the SZ40 example above.

```
3OPODXJICJ3BC8YKRYHVICKVS4+XPOBF+44FUYP4TLUJLSUWKQGWZMINNCPD4B9Y
YT8Q4KEVI9Z+9PB+NUOUZF43DPJ+ZNYGW8UL/ZK8EHDJ4Y3ZLF3Z4WSPMENB9SKA
QKSFDZGS4GR9QTGZHX+PNMDALX9PX4NPGDJEVUIP9UVY4A8Q3IAAW3CR3T9XYGYV
DKKOBVP+CLWSRJBV9CORTW/E3+HDERUMEFDQQKYPF3IRWHC3K+VUGCBEH/DPS4HD
QUFYOLTGDAPVJPPEZJ8MA+Z3HRHGVK8GYYV9DQTBETMDGJPGLY4D+XVPUJ
```

2. Decipher this text that was encrypted with SZ42. Use the key from the SZ40 example above, but add the limitation P5 two back.

```
ADY4V+KBRM9/WO+WITARUFPKYWM3GUAPEA8TYVCX9+SLCB9AKFW+XKJ4ATFJ+/9R
U9M+Y9ZH3VR44NECFJYNOBATSYYXCRGMDIWT+PQDZX34VNT+LZT8ZZIEDU4RDHYU
ZJDMZJDPJU9VDBBUKHKD/ROY84ONHYRE9C8+IK+HDHPRWHR/IMJRKOFHMO8A3D//
RIOKCYXDJRJUEG/SCKUBC8Y9I/84B8V9UFKYXN4F9HWUNTBW/9XKWUFZXLWE4994
N/4RTOOWZW9YDGMALQUICFMQYC/ZCXRDLBLBJWTHVA4QQE/LDQ4AL9DOYUHD839C
/I+O9RIGFOKVO/SYEUE83L9TWGNPXCXUZIC4LTPF
```

3. These five messages were encryped by an SZ40 with the same key. See if you can recover (most of) the plaintexts and the keystream.

```
SPBMKROMZCNARFNBOQ9RKOUJPQO+Y4BWB4L4JJQ/DBSDGNNCABTHN3WUJFGNQNK+
TQMLFPVHV+OKNF4NKZIVMFO9YI3RFJWBLNA8RASTWOWR+LSRAKRIRMK/FXHSUGYF
RM33RUAQFKKDPKRRNLHIGHXTVHBKDO+
```

```
WBXUXPD4YSBAOEJYXLQCK4LCNLEJWTOEYN/3USYDHBGO/DATOYLMJ/VMFA8OM9SJ
LPCEMJO4TVEFC8R3BNLWOYNU9QIWSNK9QOKWFIKQP9DFWIEXLK8HTVWCBTROH9Y9
V8LW/H/DGDXDJ9UW
```

```
DCYSJRYXYY+JI4N/JJOPFA9XLJFCTK8UPOUKKJL/9+KFCCAUJFUSKO3+4F/YTJFJ
UL/HT8GMQCRKR3TNEFMWKFD+383WFQ9QUNJRADWONYZ++SUZ8KWJLJYYYQPIAAY8
T43H/D93
```

```
/3OIJPQERNOHE3NGNQOW3S9NRJKY/XTFQRIFDRY3MVTDCJACJFR4EEZGAHWTCDYJ
LAPDFSJBC+DUEKJYE9DZWF/4HOKVSPDNTU8R/QKOWZLC9CCFYIJLQWVZCL9HX+9
```

```
SDKSPG8AFU9/LSJ/BYOXKQFDYJ4IZ4NITNT8L3IS9HVF94DWDSB9Y/LBXOCTRMM8
YXLXSBUHH++KBCRVZYZPT3Q4NQI3/IU/BTUPQWR/ZVF3TILSGKR8IBLNWERIT8QK
RCSPJAA8+Q
```

Unit 186
Purple

The *Purple* machine was a cipher machine used by the Japanese during World War II. It gets its name from the purple binder in which American cryptanalysts kept their notes; its Japanese name was Kyunana-shiki obun injiki (九七式欧文印字機), which means "'97 European-character machine" ('97 denoted the year 2597 in the Japanese calendar). It was also called *Type B Cipher Machine*, as it was the successor to the Red (Type A) machine. The machine is different from the rotor and teleprinter machines: it uses switches to choose permutations, and therefore when a switch steps the new permutation is unrelated to the previous one.

The Japanese language is based on a syllabary, where every syllable ends in a vowel or N. To preserve this structure and so that ciphertexts can be pronounceable by cipher clerks, vowels and consonants are encrypted separately. Japanese uses two copies of the syllabary and thousands of Chinese characters; this makes the decision to use Latin letters seem sensible, and Red and Purple both encrypt messages transcribed into the twenty-six European letters. The vowels, `AEIOUY`, called "sixes" because there are six of them, are passed through one switch (the "sixes switch") that permutes them. The consonants (the "twenties") pass through three switches that permute them. See the tables at the end of this unit for the permutations of each switch, and Figure 186.2 for a diagram of the machine.

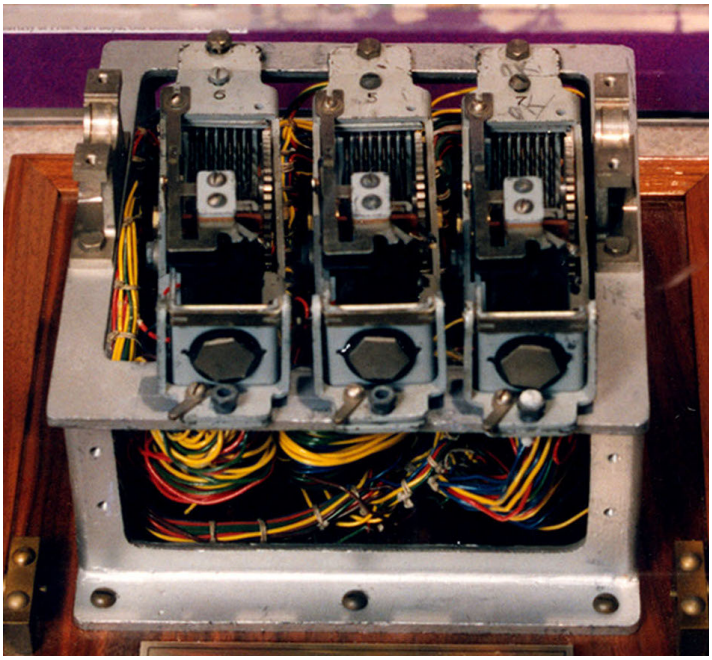


Figure 186.1: Largest surviving piece of a Purple machine. Photo from United States Air Force.

The way the machine works is as follows. The operator presses a button on the keyboard for the plaintext letter. This sends a signal on the wire for that letter. If it is one of the sixes, then it enters the sixes switch. Based on the position of that switch, the signal leaves the sixes switch on another line. The permutations for each of the twenty-five positions of the switch are in Table 186.1. If the plaintext letter is one of the twenties, then the signal is permuted by three switches. First is the “left” switch, then the “middle” switch, and finally the “right” switch. The output of the right switch leaves the machine. Each of the three twenties switches has twenty-five positions, and each position gives a different permutation of the twenty letters; see Tables 186.2, 186.3, and 186.4 for the permutations. After each letter is enciphered, the sixes switch advances to the next position. Before operation, the three twenties switches are assigned to the rôles of “fast” switch, “medium” switch, and “slow” switch. Only one of the twenties switches advances after each letter is enciphered. Whichever is the fast switch advances by one after each letter is enciphered, except when one of the other two advances. When the sixes switch advances from 25 to 1, the medium twenties switch advances by one and the fast switch does not advance. When the sixes switch advances from 24 to 25 and the medium switch is at 25, the slow switch advances by one and the others do not. Any switch that advances from position 25 comes around back to position 1.

Purple inherited its split of the alphabet into sixes and twenties from its predecessor, Red. However, the feature becomes moot with the introduction of a plugboard that permutes all twenty-six letters on the way into the machine and inversely on the way out. The full machine (P) is a conjugation of the bare unplugged machine (U) is between the monoalphabetic substitution (S) and its inverse:

$$c = P\,p = S^{-1}\,U\,S\,p$$

If it is not obvious, we tell you that the substitution shuffles which letters are the sixes and which are the twenties.

The key for Purple consists of these things:

- the initial position of the sixes switch (one number)
- the initial positions of the twenties switches (three numbers)
- the choices of which of the twenties switches are slow, medium, fast
- permutation on the plugboard (simetimes split into sixes and twenties)

For example, our key might be

sixes:	21				
left:	4	middle:	18	right:	9
fast:	middle	medium:	right	slow:	left
plugboard:	`TQCLNR PSBYGJZMFOWUDKVHAEXI`				

If we split the plugboard into sixes and twenties this way, then what we mean is the inverse permutation, so that `TQCLNR` → `AEIOUY` and `PSBYGJZMFOWUDKVHAEXI` → `BCDFGHJKLMNPQRSTVWXZ`. We should always try to be clear on exactly what we mean when we give a plugboard permutation for Purple. With this key, let us encipher the text

`THIS MESSAGE ENCRYPTED WITH PURPLE`

The first letter, `T`, is translated to `A` by the plugboard. This is one of the sixes, and from Table 186.1, we see that a signal on line 1 emerges from the sixes switch on line 3, so this letter is enciphered to `I`. The plugboard translates it to `C`. After the first letter is enciphered, the sixes switch advances to 22 and the fast twenties switch (middle) advances to 19. The second plaintext letter, `H`, is mapped to `T` by the plugboard, which is on line 16 entering the left twenties switch. From Table 186.2 we see that it emerges on line 6. The signal enters the middle switch and emerges on line 16. This enters the right switch and emerges on line 19. The plugboard translates this to `X`. The full ciphertext is

`CXEEEKVXMSXBNNRDEQEZEHLAAUCOTE`

Without the plugboard, we would get

`CBURDAWMOCAODJHURHEVBODBLEHFXE`

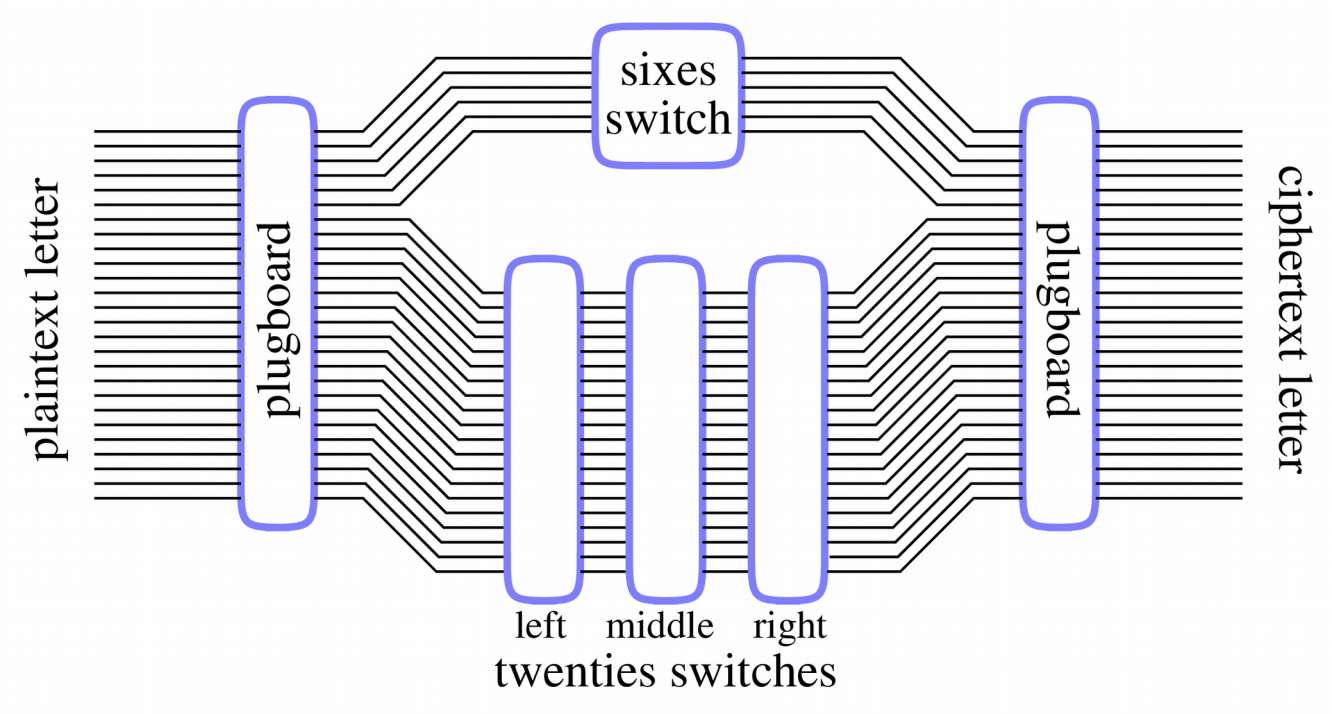


Figure 186.2: Diagram of the Purple machine.

switch position	**input** 1	2	3	4	5	6
1	2	1	3	5	4	6
2	5	4	2	6	3	1
3	1	5	6	3	2	4
4	4	3	2	1	6	5
5	3	6	1	4	5	2
6	2	1	5	6	4	3
7	5	4	6	3	2	1
8	3	6	1	4	5	2
9	6	3	5	2	1	4
10	5	4	3	1	2	6
11	2	1	6	3	4	5
12	6	5	4	2	1	3
13	2	3	1	5	6	4
14	4	2	5	1	3	6
15	1	3	4	6	5	2
16	5	6	3	2	1	4
17	6	2	4	5	3	1
18	4	1	2	3	5	6
19	1	2	3	6	4	5
20	2	5	1	4	6	3
21	3	4	6	5	2	1
22	1	5	2	4	6	3
23	4	6	5	2	3	1
24	3	4	6	1	5	2
25	6	2	4	3	1	5

Table 186.1: Permutations of the sixes switch of the Purple machine.

switch position	**input**																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	4	7	13	6	17	1	8	11	10	5	16	18	9	3	15	12	20	14	2	19
2	6	17	9	1	2	18	20	10	19	15	12	13	14	5	8	3	4	11	16	7
3	2	19	12	17	20	4	13	15	18	11	6	8	3	14	5	9	1	10	7	16
4	14	9	1	4	13	5	17	7	12	16	15	10	18	2	19	6	11	20	8	3
5	19	16	10	8	6	2	15	3	20	9	18	14	5	13	12	11	17	7	1	4
6	20	1	8	18	19	7	5	12	3	13	2	11	10	4	14	15	16	9	6	17
7	8	10	19	12	11	3	2	17	5	6	13	20	7	16	18	1	9	4	14	15
8	1	20	14	15	7	12	3	13	16	10	17	5	11	6	9	4	18	8	19	2
9	9	14	20	17	12	15	7	4	2	18	3	16	19	8	11	10	1	6	13	5
10	17	13	5	7	10	16	11	2	4	8	20	1	15	9	19	14	3	12	18	6
11	2	5	13	8	16	17	18	9	7	11	4	19	12	15	10	3	6	14	20	1
12	18	4	16	2	1	7	12	11	17	14	19	9	5	10	3	8	13	15	6	20
13	16	6	11	20	17	19	10	8	9	3	7	15	14	12	1	5	2	13	4	18
14	13	11	4	9	12	8	3	5	14	17	1	2	20	18	6	7	19	16	15	10
15	10	3	18	5	9	15	4	6	12	20	11	1	17	16	7	2	14	19	8	13
16	4	17	15	16	18	20	14	1	13	19	6	5	11	8	2	10	7	3	12	9
17	15	13	2	19	3	14	1	20	11	12	10	17	6	9	16	18	5	4	7	8
18	12	7	6	3	19	13	16	18	15	1	9	14	2	4	17	20	8	5	10	11
19	11	12	16	14	15	10	2	19	3	8	13	4	1	7	20	6	18	17	9	5
20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	7	18	12	11	4	20	9	14	1	5	16	3	8	19	10	13	6	15	17	2
22	5	3	17	18	8	11	6	16	13	7	14	15	4	20	2	19	10	1	9	12
23	9	16	19	10	7	14	13	20	8	4	5	11	12	17	18	1	15	2	3	6
24	3	8	10	13	1	9	19	2	6	18	20	7	16	11	4	15	12	17	5	14
25	19	15	7	3	14	12	18	4	5	2	8	6	20	1	13	17	9	16	11	10

Table 186.2: Permutations of the left twenties switch of the Purple machine.

switch position	**input**																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	3	8	7	18	4	15	20	10	2	9	11	13	16	19	1	14	5	12	6	17
2	17	4	20	5	11	2	14	7	6	15	12	1	19	18	3	8	10	16	9	13
3	6	14	18	1	3	17	15	4	16	19	13	7	11	10	8	5	12	2	20	9
4	14	3	10	19	12	1	6	11	17	13	2	8	20	5	9	18	15	7	16	4
5	9	2	4	6	17	13	1	18	5	20	14	10	3	8	16	15	7	11	19	12
6	11	12	14	15	1	19	4	9	6	5	17	16	10	3	20	13	2	18	7	8
7	18	7	3	2	20	16	19	1	8	11	4	9	6	13	14	10	12	17	5	15
8	2	17	11	16	10	20	12	6	18	4	8	19	13	7	5	3	14	9	15	1
9	10	6	17	7	5	12	3	2	1	16	11	20	8	15	4	19	9	14	13	18
10	17	20	12	13	9	10	16	5	11	1	3	2	15	18	19	6	8	4	14	7
11	8	16	20	4	13	7	2	19	14	18	1	11	9	3	12	15	17	5	10	6
12	15	1	2	20	16	14	9	17	3	4	10	12	5	6	7	8	18	13	19	11
13	4	13	6	9	15	11	10	12	8	2	16	17	7	14	3	1	5	19	18	20
14	19	10	5	14	18	9	15	20	7	8	13	4	6	12	17	2	11	3	1	16
15	5	19	17	16	11	8	6	13	12	7	9	3	18	2	10	20	15	1	4	14
16	20	4	2	5	6	18	16	15	9	10	7	14	17	8	13	12	19	11	3	1
17	13	5	1	3	8	2	11	16	15	17	18	6	19	4	14	7	20	9	12	10
18	7	12	19	11	1	14	13	18	4	5	15	10	9	16	2	17	6	20	8	3
19	12	9	15	8	5	16	17	7	13	18	20	19	3	1	10	11	4	6	14	2
20	3	18	10	17	7	4	19	1	20	11	2	13	12	6	15	16	9	8	5	14
21	4	15	11	18	19	3	8	17	10	14	12	5	13	20	6	9	1	16	2	7
22	1	20	13	12	2	5	10	8	9	15	6	3	14	7	19	18	16	17	11	4
23	16	14	9	10	8	6	18	2	19	3	5	20	4	17	11	1	13	15	7	12
24	8	9	16	20	14	12	7	3	13	6	19	15	2	11	18	4	17	10	1	5
25	2	11	8	6	10	20	5	14	16	12	15	18	1	9	4	7	3	13	17	19

Table 186.3: Permutations of the middle twenties switch of the Purple machine.

switch position	**input** 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	6	20	4	15	17	8	1	13	14	7	3	10	12	18	19	9	11	16	2	5
2	9	4	8	12	20	18	14	7	11	13	15	5	6	3	1	17	2	19	10	16
3	5	1	14	7	19	11	15	18	9	8	2	4	13	10	12	16	20	17	6	3
4	16	10	2	5	11	7	20	12	4	15	14	3	19	13	17	1	18	9	8	6
5	18	12	6	4	15	9	13	11	5	14	20	1	8	17	7	3	19	2	16	10
6	19	13	18	17	3	4	6	5	2	12	15	7	1	9	16	20	8	10	14	11
7	11	3	13	1	8	15	2	9	18	6	19	12	14	16	4	10	5	20	7	17
8	12	17	20	3	9	2	19	7	1	4	18	10	15	8	14	6	11	5	16	13
9	13	14	11	16	1	12	9	10	17	18	7	8	5	2	6	19	4	3	20	15
10	10	13	5	14	7	16	18	17	3	9	1	6	19	11	8	2	12	4	15	20
11	7	9	3	18	6	20	16	2	19	10	8	11	17	5	4	12	15	13	1	14
12	1	8	15	10	11	13	6	16	5	20	12	2	4	7	9	14	3	17	18	19
13	5	7	1	2	19	14	12	8	16	3	20	4	10	9	15	13	17	6	11	18
14	17	12	20	6	4	3	11	19	1	5	2	13	9	15	10	18	7	14	8	16
15	15	18	10	13	17	19	8	1	12	9	16	6	2	3	11	4	14	20	5	7
16	16	5	19	4	14	9	18	17	15	20	10	8	7	13	3	2	6	1	12	11
17	2	15	16	11	13	5	3	20	10	17	14	9	6	1	18	7	12	8	19	4
18	3	6	9	8	12	17	5	10	16	11	4	14	18	20	13	15	1	7	2	19
19	8	3	12	20	18	6	7	14	13	1	5	19	11	4	2	9	16	15	17	10
20	20	7	14	9	8	4	10	3	2	16	6	5	17	12	15	11	13	19	18	1
21	13	19	17	12	16	10	15	4	18	6	1	20	3	8	11	5	14	7	9	2
22	9	11	10	5	2	14	17	15	20	3	13	18	16	19	7	1	8	6	4	12
23	7	16	19	10	5	1	13	18	6	2	11	17	15	14	20	4	9	12	3	8
24	4	18	15	17	3	12	2	1	7	19	9	16	20	6	5	8	10	11	13	14
25	14	2	7	19	10	13	4	6	8	12	17	15	1	5	16	11	3	18	20	9

Table 186.4: Permutations of the right twenties switch of the Purple machine.

Reading and references

Mark Stamp and Richard M. Low, *Applied Cryptanalysis: Breaking Ciphers in the Real World*, Hoboken: Wiley, 2007, section 2.3.

William F. Friedman, Preliminary Historical Report on the Solution of the “B” Machine, https://cryptocellar.org/files/purple-history.pdf

Barjol Lami, Gledis Kallco, Nicholas Guo, and Sean Shi; “Cryptanalysis of Purple, Japanese WWII Cipher Machine,” 2019, https://courses.csail.mit.edu/6.857/2019/project/24-Lami-Kallco-Guo-Shi.pdf

Wes Freeman, Geoff Sullivan, and Frode Weierud, “Purple Revealed: Simulation and Computer-Aided Cryptanalysis of Angooki Taipu B,” *Cryptologia* 27:1 (2003) 1-43, DOI: 10.1080/0161-110391891739

Kenneth J. Bures, “Cracking PURPLE: cryptanalysis of the Angooki Taipu B switch tables,” *Cryptologia* 45:1 (2021) 1-43, DOI: 10.1080/01611194.2019.1706064

Kenneth J. Bures, “Cracking PURPLE: the identification of homologs in the cryptanalysis of the Angooki Taipu B cipher machine,” *Cryptologia* 47:5 (2023) 436-448, DOI: 10.1080/01611194.2022.2064200

Wikipedia, “Type B Cipher Machine,” https://en.wikipedia.org/wiki/Type_B_Cipher_Machine

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, pages 18-24 and elsewhere, https://ia600606.us.archive.org/30/items/B-001-001-264/B-001-001-264.pdf, https://archive.org/details/B-001-001-264

Programming tasks

1. Simulate the Purple machine cipher. Check that it can reproduce the examples in this unit.

Exercises

1. Encipher this text from with the example key given in this unit (with the plugboard).

   > The flowers were white, with streaks of golden orange upon the petals; the heavy labellum was coiled into an intricate projection, and a wonderful bluish purple mingled there with the gold. He could see at once that the genus was altogether a new one. And the insufferable scent! How hot the place was! The blossoms swam before his eyes.
   >
   > (from “The Strange Orchid” by H. G. Wells)

2. Decipher this ciphertext with the example key given in this unit (with the plugboard).

```
DBVCUQNJMCHFYRNSOLJFPTLIRTUIAHWXTNPVIVFMSVSCESFRBSHVCCGHNWSQEPYL
ORQUSFUFSXLPPTZMRFYRVQEALHZTJNMGIKLDWUHZPJHQECYCLFFLTDTITRGBTACC
ANWKGLODVLSVPCXFPTLIKWTGYQELUTSDKZMQZGARYDCSPJYFTOSIYTKAVLRWKXZP
SJDENHMFBDHZRUCUDSRUGLWJCYUOWRSTCPVJJTZZOYACBYTGDTABLQPNPUQKLSBQ
KMTJYCJVONEJEDXXNKAJTEZPQARNSQRMXSITWLWQNJBCALDCJEXTTPEIXZVBTXLE
YARMGXJVCYOXLVPQ
```

Unit 187
Attacking Purple with cribs

If we know the settings of the Purple machine at any point in a text, we can run the machine forwards or backwards and find its settings at any other point. So for the purposes of illustrating the method of attacking it with a crib, we may as well look at cases in which we know the plaintext from the beginning of the text; in other words, a known-plaintext attack.

Let's just work an example and explain along the way. Take this ciphertext and plaintext pair from *Riders of the Purple Sage* by Zane Grey:

```
YBUMVFPNMDCXUYRRQNGEQYAQLEWKCLHDZRDQUZSJNYBBWGINJSZEYZODOAVXEOQG
SWFGFDZUQTYHOSBXTOUQXLQBUPCHOSDLEQLYZEJHZBOQQSOKOJGBCXGBNCXOHWJF
KATSEPTUUHOYIDJZDMWZBBNJWKZYSBUXXNIQAIMNZXIWZOQOGJLIUBVEZHVTZDNL
OBETYGONLXUZGXWQFPGQATHFETKNZVCSTAQWNFVZJCRQFIXYMUKPEMTMBIBEXLSR
OTNVZHIVMSYGDLINFKWZHWRECUQBSWREKFCZJSKMWURPJSDXODELJIFMAFNHWLYO
GSCBDZUDSRNSOIOMFQFSDOIXDSVEKGLOTCEWKCNPPYVVMLQJDLNFSJYFBOLFEFDK
RNNWRVXYAPXRAUPVFAMMJSZOHEYMMWWNJZGIRRZHSRWSGEELICRWXNPAHPJKRDUT
PNLCMFFWNYJAOGSJVTCGGOUHNOQMIOOXTNQUTFODYSFJLVZMTRROPPKRTYYJCTZO
UMDIOATKSFSEYY

ITWASTHEMOMENTWHENTHELASTRUDDYRAYSOFTHESUNSETBRIGHTENEDMOMENTARI
LYBEFOREYIELDINGTOTWILIGHTANDFORVENTERSTHEOUTLOOKBEFOREHIMWASINS
OMESENSESIMILARTOAFEELINGOFHISFUTUREANDWITHSEARCHINGEYESHESTUDIE
DTHEBEAUTIFULPURPLEBARRENWASTEOFSAGEHEREWASTHEUNKNOWNANDTHEPERIL
OUSTHEWHOLESCENEIMPRESSEDVENTERSASAWILDAUSTEREANDMIGHTYMANIFESTA
TIONOFNATUREANDASITSOMEHOWREMINDEDHIMOFHISPROSPECTINLIFESOITSUDD
ENLYRESEMBLEDTHEWOMANNEARHIMONLYINHERTHEREWEREGREATERBEAUTYANDPE
RILAMYSTERYMOREUNSOLVABLEANDSOMETHINGNAMELESSTHATNUMBEDHISHEARTA
NDDIMMEDHISEYE
```

The first thing we want to do is identify the sixes and the twenties. For that purpose, we tabulate the ciphertext letters of each plaintext letter.

plaintext letter	ciphertext letters
A	MADOCK
B	GFJYQPNU
C	OD
D	KCOMDA
E	NXQSBEZVGUYTHLFWIPJR
F	QFSBWZUHNY
G	JXBWIQLET
H	PREZSUBYIGNFVJX

I	YNGTSXQWHZJFEPLI
J	
K	OM
L	YSHLIBGPRNXW
M	MCDAOK
N	UNYXBHLJPIETFSWZRVQ
O	DOKCMA
P	XFEWVQU
Q	
R	EHIQZLXJTVRNSGPY
S	VQRJBHFSTUWENGLYXI
T	BFYGLUWZETPHQXVSRI
U	WNQXZTRFHJ
V	EUG
W	URQXNTJPIZSFW
X	
Y	LZWQBFNJY
Z	

Any letter that has more than six images (what mathematicians call the thing you get after applying a mapping, such as the encryption function) is one of the twenties. Furthermore, any letter with an image that is one of the twenties is also one of the twenties. And any letter that is an image of a twenty is also a twenty. The sixes must form their own closed group. We now know the sixes and the twenties, but not in any particular order, and definitely not in the order as mapped by the plugboard.

sixes:	ACDKMO
twenties:	BEFGHIJLNPQRSTUVWXYZ

Next we are going to determine the initial setting of the sixes switch *S*. Start by tabulating the mappings of the sixes and arrange them according to the position in the text, modulo 25 (the number of switch settings). In this table, we take the first position in the text as zero. Right away we notice that for position mod 25 equal to 2, there are two fixed points: M→M and O→O. By studying Table 184.1, we know that this means that at position 2 in the text, the sixes switch can be in position 1, 5, 8, 10, 14, 15, 18, 19, or 22. Position mod 25 equal to 14 also has two fixed points. So positions 2 and 14 (12 apart) can have sixes-switch positions 10 and 22, or 14 and 1, or 18 and 5. Position 16 has a chain of five: A→D→M→O→C. This is either a five-cycle with a fixed point (A→D→M→O→C→A and K→K) or a six-cycle (A→D→M→O→C→K→A). Sixes switch position 24 has a five-cycle plus fixed point, so position 2 in the text is consistent with switch position 10. Switch position 7 has a six-cycle, so position 2 in the text is consistent with switch position 18. Switch position 3 does not have a five- or six-cycle, so we can eliminate switch position 14 for text position 2. Suppose text position 16 is switch position 24. Then its fixed point is K, which must be 5, since the permutation is (3 4 6 1 5 2). Text position 13 has A→K, so A must be 4, because the permutation of switch position 21 is (3 4 6 5 2 1). In text position 14, switch position 22, 4 is a fixed point, so that is consistent. But now suppose that text position 16 is switch position 7. Text position 17 has a cycle of length at least three, but switch position 8 does not. This is not consistent. We conclude that text position 16 has switch position 24. This means that at the start of the text, the sixes switch is in position 8.

position mod 25	mappings of sixes	position mod 25	mappings of sixes
0	C→O A→A D→M	13	A→K D→M→O
1	D→O→A→C	14	A→A D→D O→M
2	A→D→K O→O M→M	15	A→C K→M O→D
3	A→M→O→K D→C	16	A→D→M→O→C
4	D→O M→A	17	D→O→K M→A
5	M→D→C	18	A→K C→D
6	A→D O→O M→K	19	A→O→D→K
7	M→A→O→C D→D	20	A→M D→D
8	M→M A→C D→K	21	M→C D→A
9	O→D→O M→A	22	A→A O→C D→M
10	A→M→C O→O	23	A→O M→K
11	A→O→K M→M	24	M→O→D
12	K→O→M→D A→A		

Now let's see if we can deduce the plugboard mapping for the sixes. We already know that text position 16 has switch position 24, which has a five-cycle and a fixed point, which is 5. From the table above, that fixed point must be K. So we have 5=K. We also saw that in text position 13 with switch position 21, A→K, and the permutation there is (3 4 6 5 2 1), so 4=A. At text position 2 with switch position 10, A→D→K, and the permutation is (5 4 3 1 2 6), so 1=D. Also at that point, O and M are fixed points, which are 3 and 6 (but we don't know which is which yet), so the remaining letter C must be 2. At text position 3 with switch position 11, A→M, and the permutation is (2 1 6 3 4 5), so 3=M, leaving 6=O. We can now conclude that the sixes portion of the plugboard is DCMAKO.

Now that we know the starting position of the sixes switch, we not only know its position at every point in the text, but we also know when the fast twenties switch advances (every point) and when the medium switch advances (when the sixes switch advances from 24 to 25). There are $3! \cdot 25^3 =$ 93,750 possibilities for the initial settings of the twenties switches. This is a manageable number with modern computers. Our strategy is to try them all and find ones that are consistent with the plaintext-ciphertext pair that we have. A good place to start is with fixed points. The chance that a random permutation of twenty items has a fixed point is about 65%, which is rather high. But keep calm and carry on. The fixed points of letters in the twenties in our texts occur at these positions:

position in text	fixed letter	position in text	fixed letter
17	N	385	N
51	E	388	R
68	F	420	R
80	T	426	W
85	L	429	E
131	S	450	L
132	E	480	T
187	T	496	T
279	E	515	I
286	R	522	S
339	S	523	E
347	E	524	Y

The chances that any one position might have a fixed point may be 65%, but the chances that all of these positions have fixed points is about 0.003%, and that positions 51, 132, 279, 347, 429, and 523 all have the *same* fixed point is so much smaller. It looks like we will be able to narrow down the possibilities to one or a few.

We have 93,750 settings for the twenties switches to check, so let's begin with the first one: left L = middle M = right R = 1 and fast = left, medium = middle, slow = right. For these settings and what we know about the sixes switch, here are the overall permutations of the three twenties switches at the positions of interest in the text:

position in text	overall permutation of twenties switches																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
17	12	5	19	1	8	9	18	10	6	4	20	2	13	16	17	11	7	15	14	3
51	5	13	19	16	7	1	20	6	4	18	15	11	14	8	3	10	9	17	12	2
68	2	19	14	16	1	15	17	18	5	9	6	10	20	3	4	12	8	7	13	11
80	4	9	13	19	15	2	5	14	1	20	6	3	7	17	10	11	18	12	8	16
85	18	15	17	14	8	13	7	5	16	12	19	10	20	6	11	2	1	3	9	4
131	2	20	8	3	10	16	13	17	1	5	9	7	4	6	14	18	19	11	12	15
132	10	1	9	20	13	19	7	5	16	11	2	14	18	4	6	8	3	17	15	12
187	6	20	8	14	19	10	7	2	3	12	11	13	15	9	1	17	4	16	5	18
279	7	10	13	8	1	15	6	2	20	11	17	5	19	14	3	18	12	16	4	9

286	5	14	17	18	16	19	11	7	15	9	13	12	4	20	1	10	3	8	6	2
339	14	3	1	2	7	4	9	13	6	11	15	8	10	16	5	17	18	12	20	19
347	10	20	18	19	4	7	8	9	2	6	11	5	15	12	14	1	17	13	16	3
385	9	3	18	1	17	6	14	13	5	8	10	20	19	4	7	11	16	12	2	15
388	20	19	7	11	5	14	4	15	16	3	6	9	10	1	17	12	18	2	8	13
420	9	15	1	3	14	6	8	12	7	11	20	17	5	10	13	18	19	4	16	2
426	8	14	6	12	17	11	19	10	3	16	4	1	20	15	13	9	2	7	5	18
429	11	18	19	4	8	13	17	16	15	2	12	20	9	7	1	6	10	14	3	5
450	10	20	6	7	11	2	19	4	3	16	15	17	5	9	12	13	8	1	18	14
480	13	11	9	16	4	2	12	20	14	8	17	5	1	3	7	6	10	19	15	18
496	4	6	7	15	19	9	1	18	20	8	17	12	5	2	11	3	16	10	14	13
515	13	9	17	10	16	1	7	5	15	2	14	3	11	20	18	12	4	8	6	19
522	17	9	14	6	5	11	15	19	3	2	4	18	1	20	13	12	10	8	16	7
523	5	3	4	9	15	10	18	2	11	8	17	13	12	1	20	14	6	19	7	16
524	1	14	6	10	18	20	9	7	4	16	2	19	11	5	3	17	8	15	13	12

Position 17 has a fixed point (13), but the fixed point at 385 (6) does not match. Nine of these permutations don't even have fixed points. So these settings are inconsistent with our plaintext-ciphertext pair. After searching all 93,750 possibilities, we find four settings that give fixed points at all of the relevant twenty-four positions in the text:

left *L*	middle *M*	right *R*	fast	medium	slow	fixed point(s) at 17	fixed point(s) at 385
20	15	2	middle	left	right	4	6
8	18	25	middle	right	left	7	14
21	4	16	middle	right	left	2	2, 10
22	11	15	right	middle	left	12	2

Only one of these has the same fixed point at positions 17 and 385, which is what we need. So that must be the correct key.

Can we now find the plugboard settings for the twenties? Here are the fixed points for sixes = 8, left = 21, middle = 4, right = 16, fast = middle, medium = right, slow = left:

position in text	overall permutation of twenties switches																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
17	17	2	14	8	1	18	20	11	4	12	15	10	6	5	13	7	19	9	16	3
51	13	6	5	18	3	16	15	11	17	9	12	7	8	19	2	4	1	10	14	20
68	16	6	17	10	14	3	11	8	19	18	5	2	7	4	15	12	9	1	20	13
80	20	15	9	16	2	14	6	12	5	8	11	4	13	7	18	1	17	10	3	19
85	3	18	5	8	20	6	4	12	14	11	2	10	17	1	15	13	7	19	16	9
131	13	1	6	8	10	20	16	4	18	14	11	12	7	9	2	17	3	5	15	19
132	12	7	5	14	11	4	10	17	1	13	3	8	18	15	16	6	9	2	19	20
187	19	14	8	11	13	12	7	20	2	5	3	9	16	15	4	18	17	10	6	1

279	10	15	9	17	2	5	8	11	6	7	16	18	19	14	1	4	3	12	13	20
286	8	19	5	4	12	6	2	20	15	1	13	7	9	11	3	16	10	18	17	14
339	3	15	17	20	14	10	16	9	6	7	11	4	8	5	2	1	19	13	18	12
347	8	10	19	15	16	18	13	7	5	12	6	1	4	9	3	14	2	11	17	20
385	5	2	1	16	13	3	4	19	17	10	8	15	12	7	9	6	18	20	14	11
388	17	6	3	7	12	15	16	9	20	11	14	19	10	8	5	4	2	18	13	1
420	12	8	17	5	1	11	16	15	4	3	20	13	14	7	10	19	9	18	2	6
426	6	17	5	15	11	4	8	7	9	1	2	19	13	18	12	16	3	14	20	10
429	18	11	14	12	15	1	7	13	5	2	17	4	6	8	16	3	19	9	10	20
450	17	8	10	16	1	11	18	12	6	19	3	20	5	9	15	2	13	4	7	14
480	4	7	1	5	20	19	3	13	16	8	6	12	11	2	9	10	17	15	18	14
496	11	10	4	16	8	9	7	13	14	20	6	19	1	18	3	12	17	5	15	2
515	2	19	17	4	9	6	20	7	18	13	15	8	10	11	5	16	3	12	1	14
522	2	20	9	6	12	8	1	17	19	13	11	5	18	10	15	14	4	7	3	16
523	3	10	6	12	17	7	4	9	15	1	16	14	13	2	8	18	19	5	11	20
524	1	7	8	17	5	3	18	4	20	10	15	19	9	16	2	6	11	12	13	14

By comparing to the letters that are enciphered to themselves, we find

twenties letter	twenties wire	plugboard letter
B	1	
C	2	N
D	3	
F	4	
G	5	
H	6	
J	7	
K	8	F
L	9	
M	10	
N	11	S
P	12	
Q	13	
R	14	
S	15	L
T	16	
V	17	T
W	18	R
X	19	
Z	20	E

We cannot place the I, W, and Y just yet, because there are too many choices for each of them. To fill in the rest of this table, we need to locate each of the missing letters in the plaintext, check to see that we already know the mapping of the corresponding ciphertext letter, find the overall twenties permutation

at that point, and see on which line the plaintext letter must be. For example, at position 66 in the text, B is enciphered to F, which we know to be on wire 8. The overall twenties permutation at that point is

(18 19 11 13 4 9 8 15 5 3 1 2 7 10 12 16 20 6 17 14)

Since 8 is in the 7th place in the permutation, B must be on wire 7. We play the same game for the remaining twenties letters.

plaintext letter	ciphertext letter	position in text	overall permutation of twenties switches 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
B	F on 8	66	18	19	11	13	4	9	8	15	5	3	1	2	7	10	12	16	20	6	17	14
G	L on 15	307	18	9	13	12	1	19	11	5	7	17	20	2	8	4	10	15	14	16	6	3
H	R on 18	15	6	9	12	11	17	5	7	16	8	14	10	3	18	13	1	19	2	20	4	15
I	N on 2	47	5	11	13	2	19	12	18	4	10	20	8	15	9	3	17	6	14	7	1	16
J	none																					
P	F on 8	208	12	2	1	18	13	17	10	3	6	16	11	19	9	8	20	14	4	7	5	15
Q	none																					
U	N on 2	40	6	4	17	18	14	16	11	7	3	2	12	19	8	20	13	10	1	15	5	9
V	E on 20	96	3	11	8	5	19	10	16	1	9	18	2	6	13	7	12	17	14	4	20	15
W	R on 18	14	7	11	20	3	10	19	4	2	18	16	6	12	1	17	14	15	13	5	9	8
X	none																					
Y	L on 15	29	19	12	10	11	15	18	20	6	8	4	17	16	2	13	14	5	7	1	9	3
Z	none																					

We can now fill in more of the list. For example, B must be on wire 7, since that one is mapped to 8 at position 66.

twenties letter	twenties wire	plugboard letter
B	1	
C	2	N
D	3	
F	4	I
G	5	Y
H	6	
J	7	B
K	8	F
L	9	W
M	10	U
N	11	S
P	12	
Q	13	H
R	14	P
S	15	L
T	16	G

V	17	T
W	18	R
X	19	V
Z	20	E

The twenties plugboard setting is therefore _N_IY_BFWUS_HPLGTRVE, where the blanks are for letters that do not appear in the plaintext and are therefore irrelevant. We can find them, however, because they appear in the ciphertext.

plaintext letter	ciphertext letter	position in text	overall permutation of twenties switches																			
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
E on 20	J	367	18	17	12	2	5	19	16	8	14	4	6	10	7	11	1	13	15	20	9	3
E on 20	Q	16	12	17	7	5	6	13	11	9	19	18	2	20	16	14	10	8	4	3	15	1
E on 20	X	11	9	16	19	15	2	13	8	5	14	10	11	6	7	4	18	1	17	20	3	12
E on 20	Z	147	4	15	16	11	8	19	12	14	10	17	1	3	9	2	13	18	5	7	20	6

And now we can fill in the remaining spaces:

twenties letter	twenties wire	plugboard letter
B	1	Q
C	2	N
D	3	J
F	4	I
G	5	Y
H	6	Z
J	7	B
K	8	F
L	9	W
M	10	U
N	11	S
P	12	X
Q	13	H
R	14	P
S	15	L
T	16	G
V	17	T
W	18	R
X	19	V
Z	20	E

We now have the entire key: $S = 8$, $L = 21$, $M = 4$, $R = 16$, fast = M, medium = R, slow = L, sixes = DCMAKO, twenties = QNJIYZBFWUSXHPLGTRVE.

Reading and references

Barjol Lami, Gledis Kallco, Nicholas Guo, and Sean Shi; “Cryptanalysis of Purple, Japanese WWII Cipher Machine,” 2019, https://courses.csail.mit.edu/6.857/2019/project/24-Lami-Kallco-Guo-Shi.pdf

Programming tasks

1. Automate as much of the process as you like.

Exercises

1. Find as much of the key as you can from this pair of texts, taken from two parts of *The Purple Cloud* by M. P. Shiel.

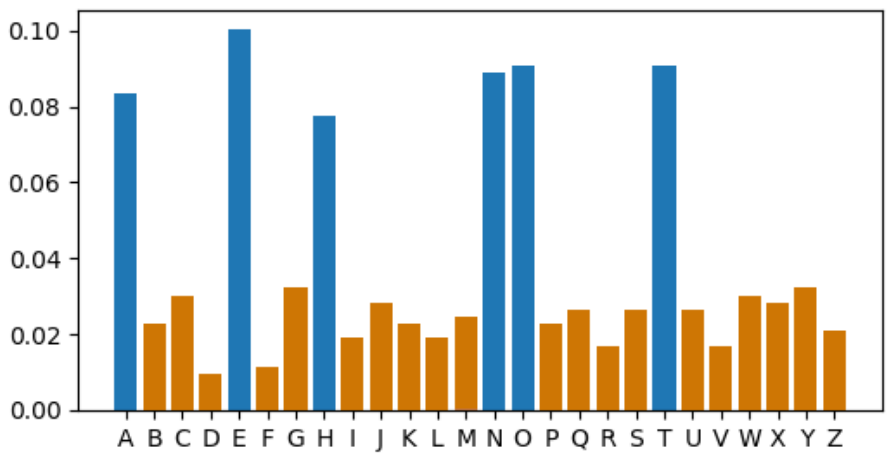
```
ALONETHATSAMEDAYIBEGANMYWAYSOUTHWARDANDFORFIVEDAYSMADEGOODPROGRE
SSONTHEEIGHTHDAYINOTICEDSTRETCHEDRIGHTACROSSTHESOUTHEASTERNHORIZ
ONAREGIONOFPURPLEVAPOURWHICHLURIDLYOBSCUREDTHEFACEOFTHESUNANDDAY
AFTERDAYISAWITSTEADILYBROODINGTHEREBUTWHATITCOULDBEIDIDNOTUNDERS
TANDITSNATUREITSORIGINREMAINSOFCOURSENOTHINGBUTMATTEROFCONJECTUR
EFORITLEAVESNOLIVINGTHINGBEHINDITNORGODKNOWSISTHATOFANYMOMENTNOW
TOUSWHOREMAINTHERUMOURTHATITISASSOCIATEDWITHANODOUROFALMONDSISDE
CLAREDONHIGHAUTHORITYTOBEIMPROBABLEBUTTHEMOROSEPURPLEOFITSIMPEND
INGGLOOMHASBEENATTESTEDBYTARDYFUGITIVESFROMTHEFACEOFITSROLLINGAN
DSMOKYMARCH

KUPCLBXBBVEPQNGYODCKZXWYYSIJYELAYJFSLRGKWJAOUALAYNINUKQOICOCYNZN
ZDIZJNHDYTSXUKVOYNWQWQZXVTKGBDZALRWAQQKAVONZHTZAWRQZSJLZVJHCOHMZ
YXLCNCWYHWDICEIBLFNMIGTPCWAXBCQWNQIYZHZEGJXBTCFZGXYEHXLXXZSCTKJP
XVBXGUVPOVCOMRBGNQSOUONRIPAWZUSUSVVVCHMDFAWFQWXNEEGOLIXSIXHEVURE
QZVLMNUHHKGEQWRVODIHMZGZWRPSCYATYCUFLVOSAYALHLDPHCDUJMLBWZNTSDEV
CGOEIQEVEXECSPJMHPZHGQYDZREAWGUMQNWZUWSDCIMSYKTRHFMFNJPIWOFUAZWM
RYKSWFWJFIEIGBVXEGOYHEDEBAMTPFLBZPVWAQUNYOLSSDOKWCSYNERPWRLRWDRN
AAQRGEWBLPJDCUJEMROLWCMXRWMYSPEFSXDQAVNLZPWUPFAPENPKJWFWJDOOOAEH
PTGXFPMPAQENNBKZVHTDBFVZMJKNHPHTRICYBEEVUYYFZBALLZWHYSJZMUAPXLFE
HNPWGYOSQTJ
```

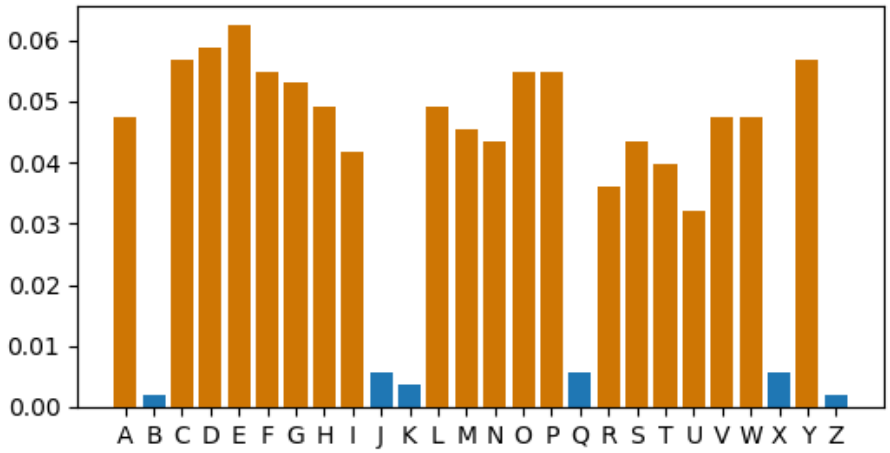
Unit 188
Attacking Purple with statistics and hill-climbing

In this unit we discuss how the use of statistics can help in analyzing a ciphertext that was encrypted with the Purple machine.

The weakest part of the Purple cipher machine is the division of the letters into sixes and twenties, and the handling of the sixes is the weaker of the two. Now, because of the split, the letters assigned to the sixes share their frequencies among themselves, while the twenties share their frequencies together. As an extreme example, suppose that we have chosen the six most frequent letters in English (A, E, H, N, O, T) as the sixes. When we encipher a typical piece of text with over 500 letters, we see the following distribution in the ciphertext. The sixes are clearly distinguished.



If instead we choose infrequent letters for the sixes, such as B, J, K, Q, X, Z, the distribution of letters in the ciphertext looks like this:

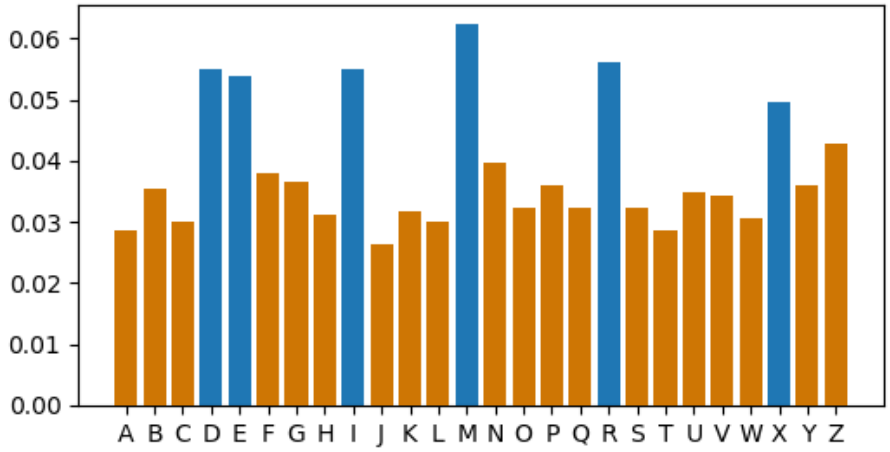


Here, the sixes are again easily distinguished by their low frequencies. Unfortunately, if the plugboard settings are chosen at random, the effect is washed out and it can be impossible to discriminate. But with a long enough text, it sometimes becomes possible again.

Here is a somewhat lengthy ciphertext with an almost randomly generated key (one—yes, just one—letter was changed in the selection of the sixes, and it makes enough of a difference). Let's see what we can do with it.

```
EFUZBKBWJVOVTVZWAQFRUFZBKXINIXIDOALXIZSUKRZHLMUFIIXWVNRVDGPNZEIK
MOXMKYUBRMRJWVYEFXRMSGTSEBKSOXAXVKEYEERLOOBEJCUEIMXNPMWFLEJBMRRQ
DPNSBZOXAGYZYQDHPBTDVWCORNMTHXCZMSOIPTDEPRAWFMMRJKNSZIKSMYARLFPD
DVRGCHNBODNUMUFPDDGCNTMMQWHGYQLUIFKRKTSYHMPMGPXQKDOGEYLXUUTGWPED
WOWVBBPOMTKCTMMUOGDEBLVPREPMISKPGCRLCWTGFPIPWZYIXMYIDJEYHEGYCJLG
EEGRYAQTQEDSCYJYNTHYEYOJZDDDYTSIUZWOQZENALNXCRHQPWKKHEIPPCNMRNEQ
IIEYKAQDPAEVRUKGNNMJYRMVSMBFQQPRYIRGWZLJOXKNAZGOTSXDZSMMCRIMPUNX
KZAAKCMMYVFPZUREMQRGFODSJCBLZRYHUGEADMFMZUDATEHSQXZRXZDDYMAEJMMN
TYICTIFDDAEHMHLEIOLDHXWKDZOUTPRBZVRRYFAKUCMODRDHFZLPMSTQMDNRREOJ
INOURWYKVCEOEXQMJBENTTSSQJCIOMMEPNPIVCMYRMSMQRREFSIFJURVVGFGEPCR
RNYZKLDAUMFILJTZDZDCQBYGTLVFXLNODYZYRHPADJZZUSFRKXZJBNCDFSYMZVJM
WIFTDUMJDFOFHKLAVRBAPXIIFRROBJQNQRTXWWSOIGNYEXXHGLNBNUZPYFVKJEUC
EZQOOZDUERVEVTQSIBVMXTUGIXAOFNEQEJPPXRPKDNNEDCRSMMBHGWHEMFDZISKJ
RTTXNIISXANQTLRTGKTWVULRNFVEKEIVSDDDUQMWXYILFLAKRIEXLVEINXPPCCOJ
YQCFJFUORLHEGKZQQAVOIFKNTZDXRKEBMAAXYNSNFGVHFIHLQXEFGBCDEIOUXPNI
EAIKEWBIYQMFOIOSRUPWLAIVASSUAMEXOMQHSASMPCNIXPZKVIXHRDXGGVIRWJGE
RNFRPAVDCUHTEHIGVBCJJLRJZKXIWLRVOGNLIEBZMNGNKMMWLZHBDEILWSZDGXBC
IGHSCMRCNFPYGFEXMEXREHLTGMGKUKEVYSDMMAILSOMBSPWFJRZVWBIZDDVESMST
ETMDPLAOIILAYUBFANTRYZNXIIHLZXERNCVRBXVCIRHFAUMIJUVQGRIEFDTVSINQ
ISUCOKDHYZWCDWHBEPXBURMHRFYMDYPBPMDDSMCVSINCQWUHXOFDMEYOCPJCVMQT
GXXDHIZQBXZVHWGQRJBAYDZMENEBDYNRGXIDBVQCZXMVFJLBFOSXUZLXRMHUPDII
YUWDGHDIMIRZLNDDFBHZMMDKXHZAXLTEJQGNWXUQQBKXUKSDLGDEEDXGOXRPFMWB
LBZEIATERIBYKVUORUHEIRZHYKTGEIXBVRIEQPZYWFBWRJHBXQEVAQYFXYNQTMMA
WMGMWWIZUZEPJVZZBQHDGGAMXZKXOPWFGGXZMXPXSPQLDIFIIBDURYFMOMUDXNCR
LNWQADQGWNHDMGTIRFKRIXDPIJKZXWNFJYWCVEUEAGYOSDCQFTXROZMURZNMXDXB
CLHBUDUGPKSWRBPEVNMRNIHMMDDSMHBJIBCX
```

Indeed, we can pick out the sixes:



The letters in the sixes have a wide range of frequencies in English, from seldom-used X to most-used E. This variation may help us in the next stage—finding the setting of the sixes switch.

The sixes switch advances by one for every letter that is enciphered, so let us bin the sixes letters in the ciphertext into twenty-five buckets, each for one position of the switch.

position in text (modulo 25)	ciphertext letters
0	DDDEEEEEEMRRXXXXXXXX
1	DDDDDDDEEEIIIIIIIIIIIRRRX
2	DEEEEEEIIMMMMRRRRRRRRR
3	DDDIIIIIIIIIIIMMMMMX
4	EEEEEMMMMMMMMRRRRRRXXXX
5	DDDDDEIIMMMMMMRRRRRRRRRRR
6	DDDDDDDDEEIIIIMMXXXXXXXXXX
7	EEEEEEEEEEEIIIIRRRRXXXX
8	DDDDDDDDDDDEEEMMMMRXX
9	DDDDEIIIIIMMMXXX
10	DEEIIIIIMMMMMRXXXXXXX
11	DEEEEEEEEEEEEIIIRRXX
12	DDIIIIIIIIMMMRRRRRRRRRX
13	DEEEEIIIIIMMMMXXXXX
14	EIIMMMMMMMMMMRRXXXXXXXXXXX
15	DDDEEEEEMMMRRRRRX
16	DDDDDDDDDDDIIIMMRRRRRRXX
17	DDDDEIIIIIIIIIIIMMMMRR
18	EEEEIIMMMMMMMRRXXXX
19	DDDDDEEEEIIMMMMRRRRRR
20	DDDDDEEIMMMMMMMMXX
21	DDEEEEIRRRRRRRRRXXX
22	DDDDDDDDDDDEEMRRRRXXXXX
23	EEEEEEEEEIIIMMMMMMMRXXX
24	DDIIIIIMMMMMMMMMMMRRRRRRXX

Notice that some letters are missing, such as I in bin 0. It is likely that I is the encipherment of X by the sixes switch for that bin, and since X is very infrequent, I is missing in that bin. But we can be more general than that, and consider all the letters of the sixes. Convert the above lists of letters into frequencies in each bin:

position in text (modulo 25)	frequencies					
	D	E	I	M	R	X
0	0.15	0.3	0	0.05	0.1	0.4
1	0.28	0.12	0.44	0	0.12	0.04
2	0.0455	0.2727	0.0909	0.1818	0.4091	0
3	0.15	0	0.55	0.25	0	0.05

4	0	0.2174	0	0.3478	0.2609	0.1739
5	0.2	0.04	0.08	0.24	0.44	0
6	0.3077	0.0769	0.1538	0.0769	0	0.3846
7	0	0.4783	0.1739	0	0.1739	0.1739
8	0.5238	0.1429	0	0.1905	0.0476	0.0952
9	0.25	0.0625	0.3125	0.1875	0	0.1875
10	0.0476	0.0952	0.2381	0.2381	0.0476	0.3333
11	0.05	0.6	0.15	0	0.1	0.1
12	0.0833	0	0.3333	0.125	0.4167	0.0417
13	0.0526	0.2105	0.2632	0.2105	0	0.2632
14	0	0.0385	0.0769	0.3846	0.0769	0.4231
15	0.1765	0.2941	0	0.1765	0.2941	0.0588
16	0.4583	0	0.125	0.0833	0.25	0.0833
17	0.1818	0.0455	0.5	0.1818	0.0909	0
18	0	0.2105	0.1053	0.3684	0.1053	0.2105
19	0.2381	0.1905	0.0952	0.1905	0.2857	0
20	0.2778	0.1111	0.0556	0.4444	0	0.1111
21	0.1053	0.2105	0.0526	0	0.4737	0.1579
22	0.4783	0.0870	0	0.0435	0.1739	0.2174
23	0	0.3913	0.1304	0.3043	0.0435	0.1304
24	0.0769	0	0.1923	0.4231	0.2308	0.0769

The frequencies of the sixes letters in American English are

D	0.03961
E	0.12497
I	0.07303
M	0.02541
R	0.06136
X	0.00198

The action of the cipher for each bin is a simple substitution on six letters. That substitution is determined by the portion of the plugboard that sets the sixes and by the position of the sixes switch. If the initial position of the sixes switch is s, and the permutation by the plugboard is π, then the true substitution for bin n is

$$S(n) \;=\; P^{-1}(\pi)\, S_6(s+n)\, P(\pi) \qquad (188.1)$$

where $s+n$ is evaluated modulo 25. There are 6! = 720 possible permutations and 25 possible initial positions of the sixes switch, so there are 18,000 configurations that we have to check for the portion of the Purple machine that handles the sixes. Because we have computers, this is not a large number. We can score a pair (π, s) for bin n by shuffling the English frequencies in a way determined by Equation 188.1 and comparing the result to the frequencies observed in that bin. The comparison can be made by looking at the cosine of the angle between the sets of frequencies in a six-dimensional vector space (see Unit 9). For example, let's score the permutation π = (2 1 3 5 6 4) and switch setting 7. When a D enters the machine, it is mapped to the second line to the sixes switch. From Table 186.1, we see that

for setting 7 it emerges on line 4, which is mapped by π^{-1} to `R`. Likewise, `E` is enciphered to `X`, `I` to `M`, `M` to `E`, `R` to `I`, and `X` to `D`. The vector of frequencies is shuffled to

$$(0.00198, 0.02541, 0.06136, 0.07303, 0.03961, 0.12497)$$

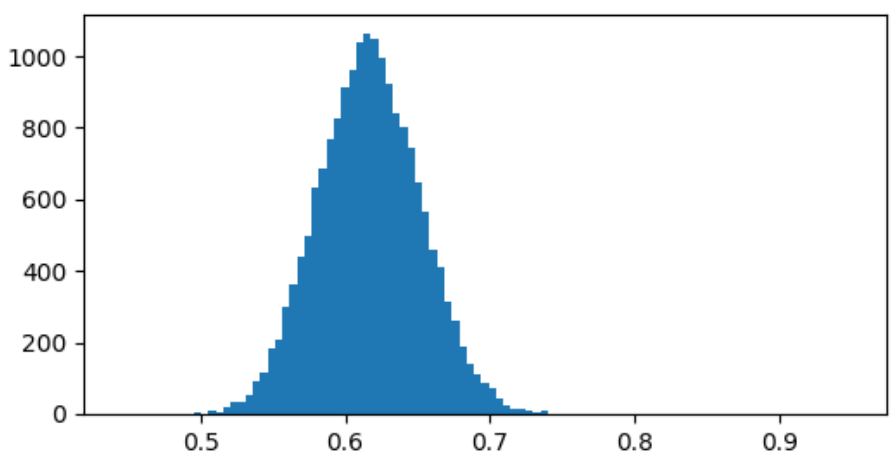
In terms of permutations,

$$\begin{aligned}(2\ 1\ 3\ 5\ 6\ 4)^{-1}\,(5\ 4\ 6\ 3\ 2\ 1)\,(2\ 1\ 3\ 5\ 6\ 4) &= (2\ 1\ 3\ 6\ 4\ 5)\,(5\ 4\ 6\ 3\ 2\ 1)\,(2\ 1\ 3\ 5\ 6\ 4)\\ &= (6\ 4\ 5\ 3\ 1\ 2)\end{aligned}$$

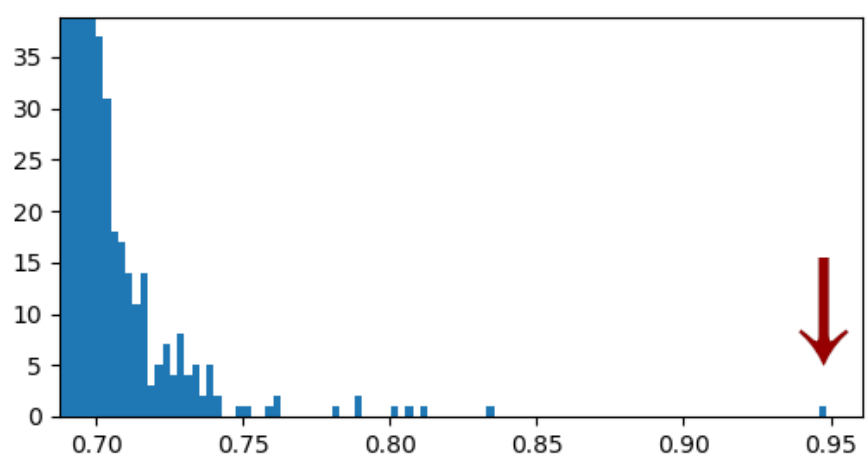
so the first number in the vector is was originally last in the list of English frequencies, the second was originally fourth, etc. Take the scalar product of that vector with this vector from bin 0 in the table:

$$(0.15, 0.3, 0, 0.05, 0.1, 0.4)$$

Divide the result by the product of the lengths of the two vectors to get the score: 0.7478. The score for a pair (π, s) can be taken as the average of the scores of the twenty-five bins. A score close to 1 is considered a good match to the data extracted from the ciphertext, and may mean that we have found the true configuration of the machine so far as the sixes are concerned. We generated a score for every (π, s), and here is an histogram of the results:



It is difficult to see the winning score, so here is an enlargement of the tail of the distribution:



The winner is π = `MRIXDE` and s = 3, with a score of 0.9484. The average is 0.6168, and the standard deviation is 0.0358, which puts our winner more than nine standard deviations above the mean. This gives us a high confidence in the correctness of the result.

At this point, if we want to, we can decrypt all the sixes in the ciphertext. We might even see the hints of words in there.

```
I..................I.....ER.ERMI...ED....R...E..IDE...D.I....ER.E.
DI....RRI....I.EDR....R....R.M..E.IDE....E...ERDI..I...I..EDI.I...
...E......R....E....E.E..R..I..I..ED.I...RED.....R..D..E...ER.I...
...E..E...ER....ER........D..E.....R.I..E..E..D..I......ID........
E....ME...MI....EM.RE.....I.......I....ERI.IE.D..R......RI.M.....I
D.........E....IDE...R......I....E.I.......ED....EE.E.ERE....E..D.
E.....I..ED..D.....I.ED......E........IX..DE.REE...I......EI......
ERE.R...I......E....E.RI.M..D..I...I.IE.DI.I.E.DI...D..E.RI.M.E..R
E..I.D..I.....M...ER......E.IDE.....E...ER.ERE..R...E.....E.RI.M..
D........E.IDE...D..E.RI.M.ERE..R...E.....E..RI.....D..E.R....I.E.
........R...E...I...D......E.I.....I...E...R.M..E.I.D........E...E
RM.DE......E.I....E...ERE..............E..I.....M.DE.I....E..ME...
ER....E.I...RE..E..ED.R.MI.....EE.E.E...D..E.RI.M.....E........E..
.M.ER..DER..E.I.D.....ERED..ER.I...............D..E........I.....ED
I.D.R..E.........I...MI....ERE..E..ED.R.M..E..E..I..I.....I......E
ED.E.....E...ER....EE.EMI...MI...EI..E...I....E.I.......E...ER..D.
....RE..E....ME.....ERE....E.E..I....EI.......RDEREDI....D....I...
ERE.R...I......E....E.RI.M.E..R.ED....RD.........E...ERM...EEM...E
.I..ED....RD.....ERE.R...I..I.....E.....I...E.I..ED.I..ER....ERE.R
...I......I..RED.......I...ERE.R...I......E....E.RI.M.E..R.EDD....
.RD........E...ERM...EEM...E..RRIED...ER....ERE.R...I..I.....E....
.I...E..RRIED..ME..I.....ER..ERE......I..RED......ERE..REI........
E...E.I.....I....ME..R.M..E...E........E...ER..R......E.RI.M....EE
.ED.E.I..I.E.IR..M.....E.....ER..RE..ERRE.R...I........E.I.....I..
..ME..R.M..ERED......D......E..E..EI.M.RERE.R...I..E
```

Now things get harder. The sixes switch had only one choice for its configuration, and the portion of the plugboard relevant to the sixes was small. But the twenties switches have $3! \cdot 25^3 =$ 93,750 possible configurations. And there are 20! (a big number) possible ways to configure that part of the plugboard. So we are going to resort to hill-climbing. The algorithm is almost identical to the one in Unit 36, and we will use tetragram fitness as the function to maximize. The differences are that we are only varying twenty letters of the key, that we have to try climbing as many as 93,750 hills, and that 1% of the time we are going to allow the routine to take the step even if the fitness is as much as 5% lower (we introduced this idea in Unit 90 when attacking a transposition cipher). The implementation is very fast, but checking all 93,750 configurations is slow, but we eventually find success when the configuration is

sixes = 3, left = 24, middle = 7, right = 12
fast = middle, medium = left, slow = right

The hill is climbed and at the peak is the twenties alphabet: `AQJYLZCKSPOHGBVNUWTF`. The full key for our example is

sixes = 3, left = 24, middle = 7, right = 12
fast = middle, medium = left, slow = right
plugboard: `MRIXDE AQJYLZCKSPOHGBVNUWTF`

The plaintext is from *Opticks, or, a Treatise of the Reflections, Refractions, Inflections, and Colours of Light* by Isaac Newton:

> I took a black oblong stiff Paper terminated by Parallel Sides, and with a Perpendicular right Line drawn cross from one Side to the other, distinguished it into two equal Parts. One of these parts I painted with a red colour and the other with a blue. The Paper was very black, and the Colours intense and thickly laid on, that the Phænomenon might be more conspicuous. This Paper I view'd through a Prism of solid Glass, whose two Sides through which the Light passed to the Eye were plane and well polished, and contained an Angle of about sixty degrees; which Angle I call the refracting Angle of the Prism. And whilst I view'd it, I held it and the Prism before a Window in such manner that the Sides of the Paper were parallel to the Prism, and both those Sides and the Prism were parallel to the Horizon, and the cross Line was also parallel to it: and that the Light which fell from the Window upon the Paper made an Angle with the Paper, equal to that Angle which was made with the same Paper by the Light reflected from it to the Eye. Beyond the Prism was the Wall of the Chamber under the Window covered over with black Cloth, and the Cloth was involved in Darkness that no Light might be reflected from thence, which in passing by the Edges of the Paper to the Eye, might mingle itself with the Light of the Paper, and obscure the Phænomenon thereof. These things being thus ordered, I found that if the refracting Angle of the Prism be turned upwards, so that the Paper may seem to be lifted upwards by the Refraction, its blue half will be lifted higher by the Refraction than its red half. But if the refracting Angle of the Prism be turned downward, so that the Paper may seem to be carried lower by the Refraction, its blue half will be carried something lower thereby than its red half. Wherefore in both Cases the Light which comes from the blue half of the Paper through the Prism to the Eye, does in like Circumstances suffer a greater Refraction than the Light which comes from the red half, and by consequence is more refrangible.

The last thing we should mention is that if we cannot pick out the sixes at the beginning of our analysis, then we need to hill-climb the entire plugboard. This does not add much to the effort.

Reading and references

Wes Freeman, Geoff Sullivan, and Frode Weierud, "Purple Revealed: Simulation and Computer-Aided Cryptanalysis of Angooki Taipu B," *Cryptologia* 27:1 (2003) 1-43, DOI: 10.1080/0161-110391891739

Programming tasks

1. Write a script or program that tests the 18,000 possible configurations for the sixes and finds that portion of the plugboard and the initial setting of the sixes switch. This assumes that we have been able to identify the sixes.

2. Write a script or program to perform the hill-climbing attack on the portion of the plugboard relevant to the twenties. Use tetragram fitness as the function to be maximized. Remember that this attack is performed for each configuration of the twenties switches (initial settings and speeds).

3. Write a script or program to perform the hill-climbing attack on the entire plugboard, for the cases in which we are unable to clearly identify the sixes.

Exercises

1. Give it a try.

```
WJQOITWTVPHJCNTYVTLZFHJQBTTQUWYFTHEQPXQIYWWCYWESSUEQSKLROUOCNONY
XFEADSNFJSLHTPYTTGYNICGQCXJRTLIAXSANSKQNJRUGYHGPECHMDXDUTRSLECHA
HNDRYYCHGJKYITMNQCAQRXCJWQLYXJAXCECSWNRCNOGVCSYDQLSKWXWAZAILWYMP
PRJALPTYOLUORCDLDUOEXRRCCPFCARHQPOCQIPXDEPSGYJZDADNONSDDNCSPXHDW
VWCCUIEJWWIODYJVTXCJASTIVPAPAHCRHSJJODJGRUSWXUSNJEHVNSCWTWLJTEUG
NXXUWREQBYHCPNPOICTLCAURQJXIATLSZUSAFJNWDBWLOEEPXYPHUYEJESDXPXWP
LICTZRNCRQNESVAPOHVXTIIXTQWLGYQLIUDZIKDJGCFYYCOYNADRXYELSSARWGNI
NNBTDPNTIIETGCYHIHIRPPYEDAUATHOPMSPMIJRLIRTDCHEIVXWPHATTVYXNJHDQ
EQHIPCOHGDNSNJXDBUKLTPNMHYLWOSOUOCPRGWOQHIRHCLCUDDETTPWLVNPHPJSD
HOJAGQRHRMOLCBPDONQAAHNUUTCWDWJMRCPWEJVJCJDRCHLGIQQKGRJDLLTDNVYT
WXEAWFLDNIPDWYKWYSEPVHJSGSYTPRWPEROYRTWLXDRIOJSIJGSASRCXGEXYCWIA
HJFYXGLLGYWOJQGOVQQAJAXSOIGNAGRZIAENFVWJNZTAENYPYCLCXXPIAPLQHMOB
NRWPPKTXMAKGROETCGIIDCBTEJHEOGUFRHTGSMWYEOLPYTUMPEAEFEYRGWPTUYAL
XBLOQLFARPHUYIGIHGTJAWGONJGLIJNATOMSQWSSRXOCWASEJRTXFNKFXIIXUDDN
XITBDXPPOGYYHJXPHUXJWWAUUGTMCREQHEUZIINQJCDHPNUZAXTXXIUPGVXPNDAE
DUUENGWJDGUGTAOCJHGMSECAANXTIECACCSDYPISASEDGILUOPEWGGCYJQFDHEJI
FCDSIALCOQISSUCPIDXTHINLIXYULWZQQYRERITANWNGOWTWNTRSNSTCPQLIOVHH
CZQUORJKXLDDSYNLINPYQJCDWDEHPTPAWSEANLAJYCKKOINBAUZEXHWOFCEYLLJX
SQQDULBRSDPCNWLUTAOXRWIHHTDUSHLISWFXPPQHENDCPPGUTRLEVTLAZZHQONTA
ASAGINYQYALHTPQIDLOSCOGPAVLIFWIQYWLDCSXRQXAXOGRSOJHGCINKEOSYYHUC
TSRCNXERESNXTUAPALGPEECWJXLWKPICAMDQQNPCHTAILVNLVSAWGUCRPOLSYSZG
LWUYUWLMIIHNNUSCJLXAUWJWPNALIMIIUAQRDFQOHYOSULECUPBNAGJAYNODSEXQ
PPTRPMGJYZJESSQRTQPQUOJSNQTCYCYGOINVIPAIPITPYPRGAZPANLAPQEPSCHDE
UNFQRSXIACHPOTWLQPQHPWANIDSHAAXUWJUCAAUTPEDDBADNHPKHWIGIWENXDOSO
IEJLIRIRIEAEEQQTRHIUFWHRICHESDSUTDHXVKOGDGCRNHUJWCVAQLOITCYGWJAP
AICDVRYTOTURAPNGHAGDRWYJVPABDEJHAHYOCITNRGQGQIQRRLNTDVRUAYEWELKZ
OJUXOQPPNXSXWGJFICMUXLYVDCPSXHTYOYGOHMPNXEDORYGGIX
```

2. Now another:

```
AAVHHBJWBYVYYJPWYYQSKAYALUDDTFBCHNLROWGHYYTTBMJIGHMXFHAOMRLWEYPP
LEJWAGBKAPDPWRLYHERVHIIYNRZTYSBBHAWNPNITVAAAZYYLNIEIHFFIMHAMBLJQ
VTNHVBXFSRVXWCJQHQUSUSTHORXRUPHTNUKCBOFCHRDCHLTGLTIQEOLQRQWENHYF
JUINBAGBNWMFSFITHSOAFYJLBPRFNWATVQVKUCWBKCOQIIRREMDVWJMFVEREHMVU
```

```
DHGQCTTJLBUQNOZAHUCZSWLTFCRWAYKCQHZNHOMVTTZXTEFOTWNJAVAVTXAMCZAO
DGJMHAUYWJWNYBZNYZKHGQUYTNZTSQFNVODWTJQSHNBYRLRKJUAYLVIRIVAALCIW
HQOFNHGAPRDJRNMTTOHTWSTTGUWTTEVHIWHJRTTQXJWSWWRARIIPOOVZAWRHBWGH
AMAHMXKAMGJNVNVNSAWHFUIHGSTANZWJLRDTSVWUNHWQNAMBAOGUPFBYMGNWPRHH
BCAHKYDCQATAYXXPMCKRTVDJODTUMRTBHMNYNLWNXHHCPKAPZTITRNZFYIUZNBNP
WFEVNMWEXGTTUBWGYREGRNSVTFUHAJCASVYEJVACWSLVVFJQICWNTUXWPIOLNREH
HYSBXWVBHXMIAWSHUWRXQTAXJPFTHWVPMLPDFBMIMDRXUAENSYUHTTUVASXWYNHV
OAAMDFWDRJNZRQSEWXDWNLLTFQLBMHVWXEXRZOMHRFWWNXJWTDEWBYPFFBSVPOVD
HTNJTADNMURDIKQLIPTEPNAWUZRNQBANWOYNANHLITCBLBRFARRIAURJXQWDNGQH
JSTNRRIRVWGEOLHFHVYSXPIRLFIAHYC
```

3. And this one:

```
CEJUQDBFZBCRZGPMACSVGOAWTKOTTEPVAQBJYBCNFCGEAADCJJROVJMYUHQOUMID
WXWKDQRJXVOOKOVZEUGFIUNERKLYJPIZJTKRKPVHVKACCRENEHYPWKRVYQBJUARP
HEZOPKPKGFRSQFFHWVCBGJHQXYNAYYPPYDSULKVFSDUHDOJGECDIXTBNRVFDLGYJ
BTCOGHPKVCMRMZXRJBHXRISBPMMRXSKLPEISUEWABJAWPWSNBJBWIYKTXJIJNBFZ
NQNTYFDBLYSHVFJBLSFKOZJHFWDJRQRUBCDHOHWDWIMTQWZMJRUKFXYOHQNIFHEW
UDEDCBZFIRANKLYGZADRLOZESVKBQDSIQJQRUFZEDQKFPQXMSRIYBFIGFCNKIUFZ
GQGXDYGIOYZRJXFJYBAOPCSOPRIMMIKSCGGRMMKTAEOKYMZRYNRULPQNTWCJUGRD
LZHDFRAPUKSMRKNKNFHBVCAONJEUZWQKW
```

There is a challenge at MysteryTwister (https://mysterytwister.org) that is similar to these examples. If you would like to try it, you can find it at https://mysterytwister.org/media/challenges/pdf/mtc3-stamp-08-purple1-en.pdf.