

# Cryptanalysis of Typex

Kelly Chang\*   Richard M. Low†   Mark Stamp‡

## Abstract

Rotor cipher machines played a large role in World War II: Germany used Enigma; America created Sigaba; Britain developed Typex. The breaking of Enigma by Polish and (later) British cryptanalysts had an enormous impact on the war. However, despite being based on the commercial version of the Enigma, there is no documented successful attack on Typex during its time in service.

This paper covers the Typex machine. We consider the development of Typex, we discuss how Typex works, and we present and analyze two cryptanalytic attacks on the cipher. The first attack assumes the rotor wirings are known and uses Turing’s crib attack—originally developed for Enigma—to recover the settings of the stepping rotors. The second attack assumes that the rotor wirings are unknown. This ciphertext-only attack uses a hill-climb to determine the wirings of the stepping rotors. Finally, we briefly consider an attack developed by Polish cryptanalysts to recover the Enigma rotor wirings and we argue that Typex was significantly more resistant to this particular attack.

## 1 Introduction

After World War I, it was clear to the British government that they needed a more efficient, mechanized cipher system. Consequently, in 1926, the British established the Inter-Departmental Cipher Committee to explore possible cipher machines to replace their codebook systems. Almost a decade later, the committee decided on “Enigma type cipher machines improved through the use of ‘Type X’ attachments” or simply, Typex [13]. These improvements included the use of patented design elements for the commercial Enigma that were unused in the German military Enigma machines.

The Typex machine was such a close relative of the German Enigma that the British were able to use Typex machines in place of Enigma machines when trying to decipher Enigma messages. In addition, when German soldiers recovered a Typex

---

\*Department of Computer Science, San Jose State University

†Department of Mathematics, San Jose State University

‡Department of Computer Science, San Jose State University: stamp@cs.sjsu.edu

sans rotors, they successfully converted it into an Enigma [13]. Ironically, the similarity between Typex and Enigma discouraged German cryptanalysts from attempting to break Typex messages, since they believed Enigma to be unbreakable [15, p. 127].

In this paper, we implement and test two attacks on the Typex cipher machine, and briefly consider a third attack. The first attack, which is covered in Section 3, is based on Turing’s crib attack on the Enigma. For this attack, we assume the following:

- The attacker knows the rotor wirings but not the key used to encrypt the ciphertext.
- The attacker knows some amount of plaintext, that is, the attacker knows a crib [18, p. 34].

We apply this attack to recover the settings of the stepping rotors. Once we have determined these settings, recovering the static rotors is relatively easy. This attack highlights the similarities and differences between Typex and its German cousin, the Enigma.

The second attack, which is discussed in Section 4, assumes the rotor wirings are unknown, but the key is known. For this attack, we utilize a hill-climb technique to solve for the rotor wirings. The implementation of this attack is not specific to the Typex cipher machine—a similar approach could be applied to many other rotor cipher machines.

The remainder of this paper is organized as follows. In Section 2, we provide a reasonably complete description of the Typex cipher, and we analyze the size of the Typex keyspace. As outlined above, Sections 3 and 4 cover two cryptanalytic attacks in detail. In Section 5, we briefly consider the method used by Polish cryptanalysts to recovering the Enigma rotor wirings and show that it would be unlikely to succeed against Typex. Lastly, Section 6 contains our conclusions and some suggestions for future work.

## 2 The Typex Cipher Machine

The Typex went through several different designs during its years of service [7]; the model pictured in Figure 1 is a Mark III, the “portable” version [13]. Its close relative, the Mark VI, another hand-powered version of Typex, appears in Figure 2. The Typex Mark VI weighed in at 30 pounds, slightly more than the Enigma’s svelte 26.5 pounds [7]. The most widely produced and distributed versions during World War II were the Mark II and Mark IV, numbering about 8000 and 3000, respectively [7].

A typical Typex machine was equipped with a set of rotors, a reflector, a keyboard, and a printer [23]. Later models included a plugboard that was apparently used to set the reflector [22, paragraph 22]. The Mark III and IV models included a handcrank to power the device [13], while others, like the Mark II, required a 230-volt AC power supply, making them stationary units [16, p. 230]. Of the available rotors, the operator



Figure 1: Typex Cipher Machine Mark III [13]

would choose five to insert into the machine, and initialize each rotor, A through Z. Three of the rotors stepped as the operator typed, while two rotors remained static—the static rotors were known as stators. As with Sigaba, but unlike the Enigma, it was possible to reverse the orientation of the Typex rotors (by reversing the rotor insert), effectively doubling the number of available rotors [13].

The Typex stators serve a similar purpose as the Enigma stecker. That is, the stators permute the signal before (and after) it reaches the stepping rotors. However, unlike the Enigma stecker, the stators were not self-reciprocal [13]. The Enigma's reciprocal property simplifies its operation, since encryption and decryption keys are identical. However, the Typex stators offer simplicity and some slight cryptanalytic advantage, although both designs have the same fundamental weaknesses.

When a Typex operator typed a letter, the signal traveled from the keyboard through the two stators, followed by the three stepping rotors, then across the reflector, and back through the inverse of the stepping rotors and stators. The path of the signal is illustrated in Figure 3.

We use the following notation to discuss the various permutations in Typex:

$$\begin{aligned} S_r &= \text{right stator} \\ S_\ell &= \text{left stator} \\ R_r &= \text{right rotor} \\ R_m &= \text{middle rotor} \\ R_\ell &= \text{left rotor} \\ T &= \text{reflector.} \end{aligned} \tag{1}$$

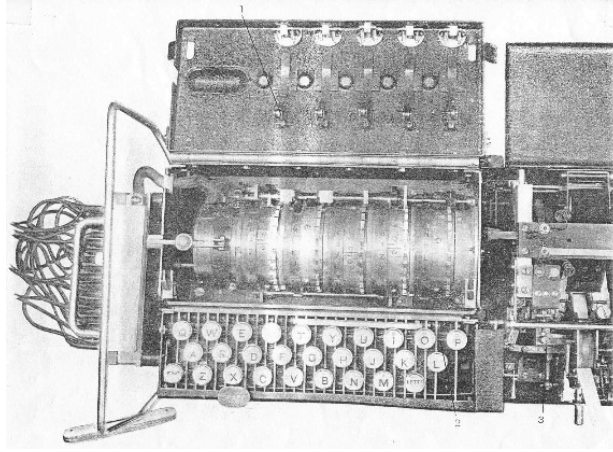


Figure 2: Typex Cipher Machine Mark VI [22]

With this notation, we can write the encryption of plaintext character  $x$  to ciphertext  $y$  as

$$y = S_r^{-1} S_\ell^{-1} R_r^{-1} R_m^{-1} R_\ell^{-1} T R_\ell R_m R_r S_\ell S_r(x). \quad (2)$$

Note that here we have ignored the dependence on the step, that is, the permutations induced by the non-static rotors varies with each character typed.

As with Enigma, the rightmost Typex rotor is its “fast” rotor (i.e., it steps once for each letter), while the “medium” rotor is in the middle, and the leftmost rotor is the “slow” rotor. However, unlike the Enigma, the middle and left rotors stepped more than once per revolution of their neighboring rotor. One of the major improvements made to the Typex over the Enigma was this multiple stepping of the rotors. The irony is that this multiple stepping came from existing unused Enigma patents [13].

The adjacent Typex rotor steps between four [5] and nine times per revolution—multi-notched rims attached to the rotors determined the number of steps [1, p. 149]. Unlike Enigma, where independent moveable rings determine the stepping of adjacent rotors [18, p. 29], the multi-notched rims of the Typex were the same for both the middle and left rotor. For example, if the middle rotor stepped when the right rotor was at, say, A, D, G, I, L, N, Q, U, and Y, then the left rotor stepped when the middle rotor was also at these same positions.

By adding multi-notched rims to the rotors, cryptanalysis of the Typex became somewhat more difficult than Enigma. This will become most apparent for the attack discussed in Section 5.

The multi-notched rims made the Typex stepping less predictable than the Enigma. For example, if we know that Typex rotors step nine times per revolution, but not the exact positions, there are  $\binom{26}{9} \approx 2^{21.6}$  combinations to consider. In contrast, for the single-notched Enigma, there are  $26^2 \approx 2^{9.4}$  combinations [18, p. 30].

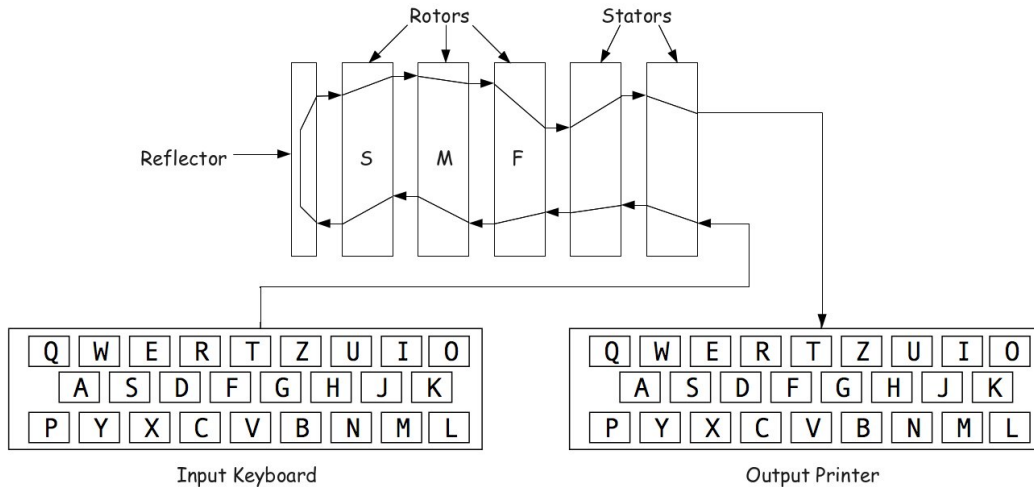


Figure 3: Typex Encryption

## 2.1 Typex Keyspace

To determine the theoretical size of the Typex keyspace, we must consider the following:

1. The choice of rotors and stators
2. The initial position of each rotor/stator
3. The choice of multi-notched rim
4. The choice of reflector

Each rotor implements a permutation of the 26 letters of the alphabet and each is initially set to one of 26 possible positions corresponding to A through Z. But, a different initial setting for a given rotor corresponds to some other possible rotor with its initial setting at, say, A. Therefore, we can neglect the initial setting when calculating the theoretical key size. Also, because the two stators remain in a fixed position, they are cryptographically equivalent to a single, fixed permutation. Consequently, there are a total of  $(26!)^4 \approx 2^{353.5}$  possible ways to select the rotors and set their initial positions.

As discussed above, each Typex rotor rim can have four to nine notches. This provides

$$\binom{26}{4} + \binom{26}{5} + \binom{26}{6} + \binom{26}{7} + \binom{26}{8} + \binom{26}{9} \approx 2^{36.3}$$

combinations. The Typex reflector worked the same way as the Enigma reflector, which has approximately  $2^{42.8}$  different possible combinations [18, p. 31]. Therefore, in total, the theoretical size of the Typex keyspace is

$$2^{353.5} \cdot 2^{36.3} \cdot 2^{42.8} \approx 2^{432.6}$$

or more than 432 bits of key. In contrast, the 3-rotor Wehrmacht Enigma has a theoretical key space of about 366 bits, as discussed in [18, p. 31].

In the World War II era, the practical Typex key space was severely limited by the available hardware. In one common configuration, there were eight rotors to choose from and only one reflector [23]. However, a Typex Mark III was issued with 10 rotors [21, paragraph 16], while the Mark VI had 14 rotors [22, paragraph 22].

The Typex operator set the key by selecting the rotors, their order, orientation, and initial position. Assuming that the operator had eight rotors to choose from, the selection of the rotors and their order gives  $8!/3! \approx 2^{12.7}$  combinations. The orientation is a binary choice, giving  $2^5$  combinations. There are  $26^5$  choices for the initial positions of the five selected rotors. Therefore, for this particular configuration, the practical Typex key space was

$$2^{12.7} \cdot 2^5 \cdot 26^5 \approx 2^{41.2}$$

or about 41 bits. With 10 rotors, this number rises to about  $2^{47.7}$ , while for 14 rotors, the key space is of size  $2^{62.2}$ . In contrast, a typical 3-rotor Enigma configuration had a practical key space of about 77 bits, but a large percentage of this was due to the stecker, which is a cryptographically weak component. Ignoring the stecker, the Enigma had a practical key space of about 29 bits. Neglecting the stators, the practical key space for a Typex with eight available rotors is

$$8 \cdot 7 \cdot 6 \cdot 2^3 \cdot 26^3 \approx 2^{25.5}.$$

With 10 rotors to choose from, the corresponding number is  $2^{26.6}$  and for a Typex with 14 available rotors, the number is  $2^{28.2}$ .

In one well-known attack on Enigma, we can effectively ignore the stecker when determining the rotor settings [18, p. 34]. Neglecting the stecker, a 3-rotor Enigma has about  $2^{29}$  settings to consider. If the analogous attack applies to Typex, that is, if we can ignore the stators and recover the rotor settings, then it could be argued that Typex is slightly weaker than Enigma, since the number of settings to consider is less than  $2^{29}$ . We consider this attack in detail in Section 3.

Determining the relative security of these two closely related ciphers is not completely straightforward. For example, in the case where the rotor wirings are unknown, Typex appears to present a much greater challenge than Enigma; such attacks are considered in Sections 4 and 5.

## 2.2 Converting an Enigma Simulator to Typex

Because of the similarities between the Typex and Enigma, we started with an accurate Enigma simulator and converted it to a Typex simulator. We removed the stecker and replaced it with the two stators. However, this required a little more than a simple find and replace. For one thing, we need to take into account that, due to the stators, the Typex is not self-inverse.

Other modifications include the extra notches on the rotors and the ability to reverse the orientation of rotors. Based on our research, we concluded that the notches for each rotor were the same [13, 23]. This slightly simplified the implementation.

The most challenging part of the Typex simulator involved reversing the orientation of a rotor. With physical rotors, this is trivial—the operator simply flips the rotor over and inserts it into the machine. Figure 4 illustrates a Typex rotor stepping in its forward orientation, while Figure 5 illustrates the same rotor stepping in its reverse orientation.

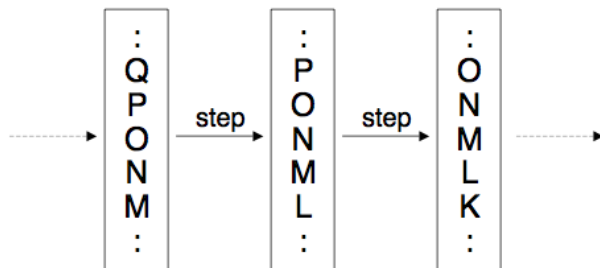


Figure 4: Typex Rotor in Forward Orientation [17]

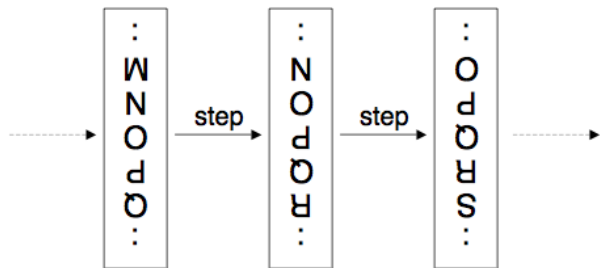


Figure 5: Typex Rotor in Reverse Orientation [17]

However, things are not so straightforward in software. For reversed rotors, we chose to simply generate a new, completely “rewired” rotor, to represent a reversed rotor. The pseudo-code for reversing the orientation of a rotor appears in Table 1.

The final modification to the simulator was to add the space character as an acceptable input. For Enigma, the only valid plaintext or ciphertext are the uppercase letters A to Z. With Typex, the operator can use the space bar, but it is connected to the X key on the keyboard and enciphers as such [13]. Therefore, our simulator checks for the space character and converts it to X prior to encrypting. Of course, when decrypting, the letter X will appear in place of spaces, as well as as for plaintext X.

Pseudo-code for the Typex simulator encryption (or decryption) and rotor step-

Table 1: Pseudo-Code to Reverse a Rotor

```

reversedRotor[26][26]
for  $i = 0$  to 25
  for  $j = 0$  to 25
    reversedRotor[ $i$ ][ $j$ ] = rotor[ $i$ ][(26 -  $j$ ) % 26]
  next  $j$ 
next  $i$ 

```

ping is given in Table 2, where we use the notation found in equations (1) and (2). The full Typex simulator code can be found in [3].

Table 2: Pseudo-Code for Typex Encryption and Stepping

```

while not EOF
  read inChar from file
  if inChar is not valid
    exit
  end if
  if inChar is a space
    inChar = X
  end if
  if  $R_m$  at a notch
    step  $R_m$  and  $R_\ell$ 
  else
    if  $R_r$  at a notch
      step  $R_m$ 
    end if
  end if
  step  $R_r$ 
  outChar =  $S_r^{-1}S_\ell^{-1}R_r^{-1}R_m^{-1}R_\ell^{-1}TR_\ell R_m R_r S_\ell S_r$ (inChar)
  print outChar
end while

```

### 3 Recovering the Key

In this section, we discuss an attack to recover the Typex key, assuming the rotors and their wirings are known. This attack is a slightly modified version of Turing’s crib attack for breaking the Enigma.

First, we recover the rotors and their initial settings, then we determine the stators. This is analogous to the Enigma attack, where we first recover the rotors and their initial settings, then we determine the stecker. This is a classic “divide and conquer” approach, in that we are able to split the key recovery into parts.

#### 3.1 Recovering the Rotors

The attack requires known plaintext, which in World War II parlance was known as a crib. Given a crib, we compare the plaintext against the ciphertext, looking for “cycles” between letters. With enough cycles, we can recover the rotor settings and, with a little extra work, the stator settings as well. This is similar to Turing’s attack on Enigma [18, p. 31], although the Typex stators are somewhat more challenging to recover than the Enigma stecker settings.

Let  $S(x)$  be the transformation of the letter  $x$  when it passes through the stators. Using the notation in equation (1), we have  $S = S_\ell S_r$ . Then,  $S^{-1}(x)$  is the transformation when  $x$  passes through the stators in the opposite direction. We define  $P_i$  as the overall permutation through the rotors and reflector—but not the stators—at step  $i$ , that is,

$$P_i = R_r^{-1} R_m^{-1} R_\ell^{-1} T R_\ell R_m R_r \tag{3}$$

where, again, we are using the notation in equation (1). Since we can decrypt, the inverse permutation  $P_i^{-1}$  exists.

Next, we look for a cycle in a given crib (or series of cribs). For example, in Table 3, column 6 yields the equation  $S^{-1}P_6S(\mathbf{T}) = \mathbf{I}$ , which we can rewrite as  $P_6S(\mathbf{T}) = S(\mathbf{I})$ . We generate analogous equations for columns 12, 2, 20, 23, and 15, respectively, as

$$\begin{aligned} P_{12}S(\mathbf{I}) &= S(\mathbf{S}) \\ P_2S(\mathbf{S}) &= S(\mathbf{M}) \\ P_{20}S(\mathbf{H}) &= S(\mathbf{M}) \\ P_{23}S(\mathbf{E}) &= S(\mathbf{H}) \\ P_{15}S(\mathbf{E}) &= S(\mathbf{T}) \end{aligned}$$

which, when combined with our equation from column 6 yields

$$S(\mathbf{T}) = P_{15}P_{23}^{-1}P_{20}^{-1}P_2P_{12}P_6S(\mathbf{T}). \tag{4}$$

We refer to equation (4) as a cycle, since the permutation  $P_{15}P_{23}^{-1}P_{20}^{-1}P_2P_{12}P_6$  has a fixed point at  $S(\mathbf{T})$ .

$i$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Plaintext	T	E	S	T	O	F	T	Y	P	E	X	C	I	P	H	E	R	M	A	C	H	I	N	E
Ciphertext	P	L	M	A	D	L	I	C	V	B	M	Y	S	U	D	T	L	W	P	R	M	V	V	H

Table 3: Typex Crib

Given a set of cycles involving  $S(\mathbf{T})$ , we then do an exhaustive search over the choice of rotors, their orientations, and initial positions. Note that for each such putative key, all  $P_i$  and  $P_i^{-1}$  are known.

Consider a given putative key. To make use of equation (4), we make guesses for  $S(\mathbf{T})$ . If we find that for some  $x = S(\mathbf{T})$ , equation (4) holds, then the putative key is consistent with the crib; if equation (4) never holds for any choice of  $S(\mathbf{T})$ , then we discard the putative key. However, there are 26 choices for  $x = S(\mathbf{T})$  and for each, there is a  $1/26$  chance that equation (4) holds at random. Consequently, we get no reduction, on average, in the number of putative keys when considering one cycle. But, if we have two cycles in  $S(\mathbf{T})$ , then there are still just 26 choices for  $S(\mathbf{T})$ , but there is only a  $1/26^2$  chance that both equations hold. Consequently, two cycles in the same letter provide a reduction in the number of keys by a factor of 26. Furthermore,  $n$  cycles give us a reduction by a factor of  $26^{n-1}$ .

Here, we restrict our attention to the case where eight Typex rotors are available. Recall from Section 2.1 that in this case there are about  $2^{25.5}$  ways to select and initialize the Typex rotors. Therefore, for this part of the attack we must consider each of these  $2^{25.5}$  putative (partial) keys. Since

$$26^5 < 2^{25.5} < 26^6$$

given six cycles in one letter (or various other combinations involving cycles on different letters), we expect to find a unique solution.

Note that for the crib in Table 3, we can find additional cycles in  $S(\mathbf{T})$ . For example

$$\begin{aligned} P_0 S(\mathbf{T}) &= S(\mathbf{P}) \\ P_8 S(\mathbf{P}) &= S(\mathbf{V}) \\ P_{21} S(\mathbf{I}) &= S(\mathbf{V}) \\ P_6 S(\mathbf{T}) &= S(\mathbf{I}) \end{aligned}$$

yields the cycle  $S(\mathbf{T}) = P_0^{-1} P_8^{-1} P_{21} P_6 S(\mathbf{T})$ .

For this part of the attack, we consider  $2^{25.5}$  settings, and for each setting we make 26 guesses for the cycle values (i.e., the stator output). This yields a work factor of about  $2^{30.2}$  to recover the rotor settings.

## 3.2 Recovering the Stators

Once the British recovered the correct rotor settings for the Enigma, they could easily determine any remaining unknown stecker cables by looking for swapped letters in the putative plaintext. For example, if the putative plaintext read LEIHLITHEK, then they could see that a stecker cable swapped the letters L and H [18, p. 55]. However, unlike the Enigma stecker, the Typex stators are not their own inverse. Consequently, for Typex, the output produced by the correct rotor settings (with incorrect stators) still looks like ciphertext.

Once the rotor settings have been determined, an exhaustive search to recover the stators has a work factor of

$$5 \cdot 4 \cdot 2^2 \cdot 26^2 \approx 2^{15.7}$$

since we must choose two of the remaining five rotors, put them in the correct order, and for both stators, set the orientation and initial position. The work factor here is much less than that needed to recover the rotors, so an exhaustive search is a reasonable approach. In our experiments, we found that trial decryption of 200 characters was more than sufficient for this exhaustive search. Therefore, the total work to recover the stators is less than  $2^{23.3}$ .

In the previous section, we showed that the work to recover the rotors is about  $2^{30.2}$ . It follows that the overall work factor for this attack is dominated by the work needed to recover the rotors.

We also experimented with a hill-climb attack [12, p. 157] to recover the stators, where we begin with an arbitrary solution and incrementally improve the solution through a series of minor transformations, retaining the modification if the resulting putative solution is better than the previous putative solution. More details on this hill-climb approach can be found in [3]

The bottom line here is that, as with the Enigma, a divide and conquer attack on Typex is feasible. And, as with the Enigma, this attack significantly reduces the work factor as compared to an exhaustive search to recover the key. For Typex, we need to consider  $2^{25.5}$  rotor settings, while the comparable number for Enigma is about  $2^{29.4}$ . However, the number  $2^{29.4}$  includes a setting for the moveable rings on the Enigma, while we are assuming that the Typex notches are known and fixed. From this perspective, Typex appears to offer somewhat less security than Enigma.

If the Germans had chosen to pursue an attack on Typex, they would have faced a significant practical difficulty in intercepting enough ciphertext and determining cribs. Two additional factors that affected the Nazi's ability to cryptanalyze the Typex were their excessive confidence in the security of Enigma and a chronic lack of resources [8, p. 138].

## 4 Hill-Climb for Rotor Wirings

Hill-climbing is a heuristic search technique. The two necessary ingredients of any hill-climb algorithm are a method to iteratively modify a putative solution and a means for calculating how “good” or “close” a putative solution is to the actual solution. In practice, we measure the “goodness” of a solution using a numeric score—a solution with a score closer to that of the actual solution is considered a better solution. As the solutions continue to improve, the scores continue to climb, thereby giving the technique its name. However, a hill-climbing technique will, in general, only find a local maximum, since once a local maximum has been reached, we cannot climb further. As a result, hill-climb algorithms generally exhibit a sensitive dependence on the initial putative solution.

Hill-climb attacks have been successfully applied to classic machine ciphers, such as the Japanese Purple [9]. However, it is worth noting that well-designed modern ciphers are not susceptible to such attacks. For modern ciphers, an incorrect key that is close to the correct key should reveal no more about the plaintext than a randomly chosen key. In such a case, there are no hills to climb—a scoring function is essentially a delta function with a spike at the correct key, and all remaining keys live in a “flat” space, with random variations in score.

In this section we consider the effectiveness of a hill-climb to recover unknown Typex rotor wirings. We assume that the key (i.e., orientation and initial settings of the rotors) is known, but this is not an essential aspect of the attack—if the key is not known, we could recover an equivalent rotor wiring, assuming some arbitrary initial setting (e.g., forward orientation, initially set to A).

### 4.1 Scoring the Wiring

For our hill-climbing algorithm, we require some measure of how close we are to the correct solution. To define such a score, we first assume that the plaintext is English. Then given a putative wiring, we decrypt the ciphertext and score the resulting putative plaintext using English digram frequencies, that is, the frequency of pairs of letters in English. The fast simple substitution attack in [11] uses only digram frequencies, so digrams should be sufficient for our purposes.

We modify the standard digram matrix to improve the efficiency of the attack considered here. First, we account for the fact that Typex automatically converts a space to an X. Second, we account for the case where the rotor wiring is correct, but the initial position is not, that is, we pre-compute shifts of the digram matrix. With these modifications in mind, we generated a digram matrix using *Alice’s Adventures in Wonderland*, *Common Sense*, and *Adventures of Huckleberry Finn* [2, 14, 20].

## 4.2 Swapping

In addition to a scoring function, we also need a systematic way to modify a putative rotor. Again, we mimic simple substitution attacks, such as that in [11], and simply swap elements.

After considerable experimenting, we settled on the swapping algorithm discussed below. The paper [3] contains a discussion of other swapping variations that were tested.

We start with a swapping algorithms described in [6]. In this method, we swap the wirings in a series of rounds: Given an  $N$ -letter rotor with putative wirings  $W_1$  to  $W_N$ , in the first round we swap  $W_1$  with  $W_2$ , then  $W_2$  with  $W_3$ , and so forth. In the second round, we swap  $W_1$  and  $W_3$ , then  $W_2$  and  $W_4$ , etc.. We continue to swap in this manner until we reach round  $N - 1$ , where we only swap  $W_1$  with  $W_N$ . Note that these  $N - 1$  rounds give us a total of  $\binom{N}{2}$  swaps.

We then iterate by repeating this sequence of  $\binom{N}{2}$  swaps until we go through one entire iteration without any improvement in the score. Pseudo-code for our swapping method is given in Table 4.

## 4.3 Hill-Climbing is Hard Work

As outlined in the previous section, we do a total of  $k\binom{N}{2}$  swaps, for some  $k$ . In practice, we find that to recover the wirings of one unknown rotor,  $k \approx 4$ . Consequently, we consider about  $2^{10.4}$  swaps. This number is small compared to an exhaustive search, which has an expected work factor of  $26!/2 \approx 2^{87}$ . However, the work for our attack is still significant, since we have to use the Typex simulator to decrypt the ciphertext for each swap. Therefore, the total amount of work depends on the amount of ciphertext required. Our tests in Section 4.4 show that with one unknown rotor, for a high probability of success, we need about  $2^{15.6}$  ciphertext characters. When we combine this number with the number of swaps required, this implies a work factor of about  $2^{26}$  operations to recover the wiring of one unknown rotor. In contrast, for a brute force attack, the work is  $2^{87}M_1$ , where  $M_1$  is the amount of text that must be tested. Regardless of the size of  $M_1$ , our approach is a major shortcut.

With each additional unknown rotor the work increases exponentially. In the case of two unknown rotors, we have implemented a nested hill-climb attack. For this attack, we need about  $2^{20}$  characters, on average. In addition, two unknown rotors average about  $2^{21}$  swaps to recover both wirings. In total, this gives a work factor of  $2^{21} \cdot 2^{20} \approx 2^{41}$ . The exhaustive search work to simultaneously recover two rotors is  $2^{87} \cdot 2^{87}M_2 = 2^{174}M_2$ , where  $M_2$  is the amount of data required. Our work factor is also much better than would be expected from running a nested version of our single-rotor hill-climb, which would have a work factor of  $2^{26} \cdot 2^{26} = 2^{52}$ .

Next, we consider the one-rotor case in more detail. Then we provide more details on the two-rotor attack.

Table 4: Pseudo-Code for Swapping Method

```

—MAIN—
while previous wiring  $\neq$  current wiring
  previous wiring = current wiring
  while round  $< N$ 
    SWAP(round,position)
    if newscore better
      keep wiring
    else
      swap back
    end if
    increment position
    if position+round  $> N-1$ 
      reset position
      increment round
    end if
  end while
end while

—SWAP(round,position)—
temp = wire[position]
wire[position] = wire[position+round]
wire[position+round] = temp

```

#### 4.4 One Rotor at a Time

As mentioned above, we assume that the initial position, order, and orientation of each rotor is known. In this section, we assume the wirings of the rightmost rotor are unknown, but all other wirings are known.

The unknown rotor wirings are initialized to randomly selected values and we then use the hill-climb attack discussed in Section 4, varying the amount of available ciphertext. Our results are summarized in Table 5. With 100,000 characters of ciphertext, our success rate was  $1770/2000 = 0.885$ , where success is defined as the recovered rotor having no more than four incorrect wirings.<sup>1</sup> Also, for this case, on average, our program only took about 17 seconds to complete the attack, with an average of 1273 swaps. One interesting thing to note is that as the amount of data increases the number of swaps slightly decreases. This can be attributed to the fact

---

<sup>1</sup>In all of our experiments, we found that the recovered rotor had four or fewer incorrect wirings, or it had essentially no correct wirings.

that with more data, the scoring improves, so we can expect to climb more quickly to a local maximum.

Data Used	1000	2000	5000	10,000	25,000	50,000	75,000	100,000
Swaps	1365	1380	1419	1420	1390	1342	1318	1273
Seconds	1.439	1.867	2.809	4.107	6.091	11.964	14.394	17.236
Successes	1	15	260	506	918	1472	1559	1770
Test Cases	2000	2000	2000	2000	2000	2000	2000	2000

Table 5: One Rotor Hill-Climb Results

## 4.5 Two Unknown Rotors

After verifying that hill-climbing for one rotor is possible, we considered two unknown rotors. For the experiments presented here, the right and middle rotors are unknown; additional test cases are given in [3]. As previously mentioned, the amount of time it takes to recover two unknown wirings is exponentially greater than that for one rotor, since the number of swaps increases exponentially. Additionally, the amount of data necessary to recover the wirings increases.

The results for this test case are summarized in Table 6. For example, using 1,000,000 ciphertext characters, we had a success rate of 38.7%, where success is defined as recovering both rotor wirings, with each having no more than four incorrect wirings. For this case, we needed an average of more than 2,000,000 swaps and the program took, on average, about 41 minutes to complete.

Data Used	50,000	100,000	250,000	500,000	750,000	1,000,000
Swaps	2,047,712	2,056,453	2,065,281	2,054,810	2,051,334	2,042,556
Minutes	5.3	8.8	14.1	27.1	36.2	40.8
Successes	7	75	144	203	291	387
Test Cases	1000	1000	1000	1000	1000	1000

Table 6: Two Rotor Hill-Climb Results

The average number of swaps for the two-rotor experiments is roughly the square of the number needed in the one-rotor case. And, as expected, the time also increases based on the number of times the simulator needs to execute.

In principle, this attack could be extended to all three unknown rotors. However, the time and data requirements would make a three-rotor attack extremely challenging, even with modern computing power.

## 5 The Polish Attack to Recover the Wirings

Marian Rejewski was a brilliant Polish mathematician and cryptanalyst who, in 1932, recovered the internal wirings for the Enigma rotors. His attack exploited the use of an encrypted message key that appeared at the beginning of each ciphertext message. By carefully analyzing the cycle structure derived from these keys, Rejewski was able to recover the wirings of the Enigma rotors [4].

Each day, Enigma operators had a new key to use. However to avoid having a large number of characters enciphered with the daily key, a three-letter message key was selected, which indicated the initial settings of the three rotors for the upcoming message. This three-letter message key was repeated, then encrypted with the daily key. That is, only six letters per message used the daily key and these six letters were the message key, which consisted of a three letter sequence repeated twice. The repetition was used to avoid mistakes from operator error and signal interference. However, this repetition was the crucial mistake that allowed Rejewski to recover the unknown rotor wirings [19, p. 50].

An excellent description of Rejewski's work can be found in [4] and we will not repeat the details of the attack here. Instead, we simply want to consider whether the same attack could succeed against Typex. That is, assuming that Typex operators had used the same process to transmit the daily key, would Rejewski's attack be feasible?

Under an equivalent set of assumptions, Rejewski's attack would succeed on Typex. However, there are at least two significant problems related to these assumptions. For one thing, the attack relies on knowledge of the Enigma stecker settings for the given day. For Typex, this corresponds to having complete knowledge of the stators. Since the stators are themselves rotors, assuming the rotor wirings are unknown, this information would not be available. While we might hope to have information on which rotors are used as stators, and their settings, this would not be of any use unless the wirings for those particular rotors were known. So, the attack appears to be limited to rare cases, such as when the rotor wirings are known, except for a new rotor that has been added to the mix.

Another crucial assumption in Rejewski's attack is that only the fast rotor steps during the encryption of the 6-character message keys. For Enigma, this is generally a safe assumption, since the middle rotor steps once per 26 steps of the fast rotor. Assuming the fast rotor initial position is set at random, for the Enigma the chance that only the fast rotor steps during the encryption of the first six characters is  $20/26 \approx 0.77$ . This implies that for a given daily key, there is about a 77% chance that the attack is viable.

In contrast, for the multi-notched Typex rotors, it is unlikely that we can encrypt six characters without having the middle rotor step. And even if we are fortunate enough to obtain the rotor wirings, determining the number and positions of the notches would be nontrivial. This illustrates a strength of the multi-notched Typex rotors as compared to single-notched Enigma rotors.

## 6 Conclusion and Future Work

When it became clear that codebook ciphers and other methods used in World War I were no longer sufficient, the British began to search for other ways to encrypt sensitive information. One result of this search was the Typex cipher machine, a surprisingly close cousin of the commercial Enigma machine.

The Typex incorporated two stators, and not Stecker, as in Enigma. In addition, Typex used a printer for output, as opposed to the Enigma light board. Most significantly, the Typex rotors step more frequently than the Enigma rotors. Whereas the Enigma rotors step in an odometer-like fashion (with an occasional double-stepping [10]), the Typex rotors step anywhere from four to nine times per rotation of the adjacent rotor.

While the more frequent stepping may have been an improvement over the Enigma, it certainly does not make Typex immune to attacks that break the Enigma. The Enigma attack that was developed by Alan Turing during World War II [18, p. 34] also works on Typex. The stators, which lack the reciprocal property of the stecker, make the Typex attack marginally more difficult than that for the Enigma. In either case, a classic divide and conquer approach is successful, with a comparable work factor and data requirement for both ciphers.

Turing’s crib attack assumes that the rotor wirings are known; if not, the wirings must be recovered first. We showed that a hill-climbing technique can be used to recover the Typex wirings of one unknown rotor, with a high probability of success. However, the data requirements would likely have made such an approach impractical during World War II. The same attack can be extended to recover two rotors simultaneously but the work and data requirements grow substantially, while the success rate declines significantly.

Finally, we considered Rejewski’s attack to recover the Enigma rotor wirings and showed that it can succeed against Typex. However, for the attack to be successful, it requires that the same assumptions as for Enigma also hold true for Typex. In particular, the medium and slow rotors cannot step during the encryption of the message key. However, due to the Typex multi-notched rotors, this stepping condition would rarely be met.

There are several open problems related to the attacks presented in this paper. We have shown that it is possible to use a nested hill-climbing algorithm to solve for one or two unknown rotors. For each swap, we decrypt the ciphertext and compute statistics on the putative plaintext. This hill-climb attack would be much more practical if a faster scoring algorithm was available. For fast attacks on a simple substitution cipher [11] or a more general homophonic substitution cipher [6], only one putative decrypt is necessary—all remaining score calculations are determined by manipulations of a scoring matrix. Consequently, the work factor for such attacks are essentially independent of the amount of ciphertext used. If a comparable scoring technique were applicable to the hill-climb attack considered in this paper, it would then be well within the practical range of World War II-era technology.

Second, we showed that Rejewski's cycle structure method can, in principle, be applied to Typex. However, given the multi-notched rims on the Typex rotors, it is unlikely that neither the middle nor left rotor steps when six consecutive letters are encrypted. It might be interesting to attempt to generalize Rejewski's attack to the case where one notch is encountered and the middle rotor steps once.

## Acknowledgment

The authors thank an anonymous referee for providing several fascinating and relevant references to the archival literature.

## References

- [1] Bauer, F. L. (2007). *Decrypted Secrets: Methods & Maxims of Cryptology*. Springer.
- [2] Carroll, L. (n.d.). *Alice's Adventures in Wonderland*. Retrieved from [accessed January 6, 2013]  
<http://www.gutenberg.org/cache/epub/11/pg11.txt>
- [3] Chang, K. (2012). Cryptanalysis of Typex. *Master's Projects*, paper 235. Retrieved from [accessed January 6, 2013]  
[http://scholarworks.sjsu.edu/etd\\_projects/235/](http://scholarworks.sjsu.edu/etd_projects/235/)
- [4] Christensen, C. (2007). Polish Mathematicians Finding Patterns in Enigma Messages. *Mathematics Magazine*, 80(4), 247–273. Retrieved from [accessed January 6, 2013]  
<http://mathdl.maa.org/mathDL/22/?pa=content&sa=viewDocument&nodeId=3131>
- [5] Cryptographic Description, Type X Machine (n.d.). National Archives and Records Administration of the United Kingdom, HCC Nr. 3588
- [6] Dhavare, A., Low, R. M. & Stamp, M. (2012). Efficient cryptanalysis of homophonic substitution ciphers. Retrieved from [accessed January 6, 2013]  
<http://www.cs.sjsu.edu/faculty/stamp/RUA/homophonic.pdf>
- [7] Erskine, R. (1997). The Development of Typex. *The Enigma Bulletin*, 2, 69–85.
- [8] Ferris, J. R. (2005). *Intelligence and Strategy: Selected Essays*. Routledge.
- [9] Freeman, W., Sullivan, G. & Weierud, F. (2003). Purple revealed: Simulation and computer-aided cryptanalysis of Angooki Taipu B, *Cryptologia*, 27(1), 1–43.
- [10] Hamer, D. (1997). Enigma: Actions involved in the 'double-stepping' of the middle rotor, *Cryptologia*, 21(1), 47–50.
- [11] Jakobsen, T. (1995). A Fast Method for the Cryptanalysis of Substitution Ciphers. *Cryptologia*, 19(3), 265–274.

- [12] Kreher D. & Stinson D. (1999). *Combinatorial Algorithms*. CRC Press.
- [13] Kruh, L. & Deavours, C. A. (1983). The Typex Cryptograph. *Cryptologia*, 7(2), 145–166.
- [14] Paine, T. (n.d.). *Common Sense*. Retrieved from [accessed January 6, 2013]  
<http://www.gutenberg.org/cache/epub/147/pg147.txt>
- [15] Ratcliff, R. A. (2006). *Delusions of Intelligence: Enigma, Ultra, and the End of Secure Ciphers*. Cambridge University Press.
- [16] Rusbridger, J. & Nave, E. (1991). *Betrayal at Pearl Harbor: How Churchill Lured Roosevelt into World War II*. Summit Books.
- [17] Stamp, M. & Chan, W. O. (2007). SIGABA: Cryptanalysis of the Full Keyspace. *Cryptologia*, 31(3), 201–222.
- [18] Stamp, M. & Low, R. M. (2007). *Applied Cryptanalysis: Breaking Ciphers in the Real World*. John Wiley & Sons, Inc.
- [19] Trappe, W. & Washington, L. C. (2005). *Introduction to Cryptography with Coding Theory*, second edition, Prentice Hall.
- [20] Twain, M. (n.d.). *Adventures of Huckleberry Finn*. Retrieved from [accessed January 6, 2013]  
<http://www.gutenberg.org/cache/epub/76/pg76.txt>
- [21] Type X Machine, Mark III, Volume 1 (1941). National Archives and Records Administration of the United Kingdom, Public Record Office, AIR 20-1531.
- [22] Type X Machine, Mk VI (n.d.). National Archives and Records Administration of the United Kingdom, Public Record Office, AIR 20-1531.
- [23] Wescombe, P. (24 September 2011). Personal correspondence.