The Enigma Machine
Joshua I Gold
2002

THE MACHINE

The basic Enigma machine used three rotors and a plugboard ("stecker"). The current went from the keyboard, through the plugboard then the three rotors, bounced of the reflector, went back through the three rotors and plugboard, then lit the a bulb. Of course, after each keypress, the right (or "fast" rotor) turns by 1/26 of a rotation; at a certain point of this rotation, the next ("middle") rotor turns 1/26; same with the third (to be technically correct, this rotation occurs immediately after the keypress but before the current flows).

There were originally only 3 rotors. By 1938, the Germans were using 5. By 1940 or 41, they had 8. For each rotor, part of its function was set by the user: 26 "ring settings", corresponding to the letters of the alphabet, represented shifts in the input/output mapping (e.g., a setting of "A" meant to shift the output by one character; "B" = two characters; etc). A particular ring setting was referred to as a "base position." Also, two properties of the rotors were fixed: the wiring that created the input/output mapping, and the position on the rotor at which point the leftward neighbor would rotate. This second point is key to Banburismus, as described below.

ENCRYPTION

To use the Enigma machine, the German operator would establish settings for the day, as specified by a key that was sent to all the operators. This key would specify: rotor order (which three of the eight rotors to use and in which order; 8x7x6=336 permutations), ring settings (26x26x26=17,576 permutations), and plug connections (10 pairs connected = 152,418,964,472,775 permutations).

Then for each message one final step of setting the machine was necessary -- to set the "starting point"; that is, where the rings actually start at the beginning of the message. This setting was chosen by the operator and was sent as part of the message. They were sent using ring positions defined by the "Grund," which was part of the daily key. The air force used a stupid protocol for sending this information -- it sent the three-letter key (the "trigram" corresponding to the initial letter settings of the three rotors) twice in a row; this was readily broken. The naval Enigma operators used a more complicated method that was part of the reason why their codes were so much more difficult to break. They paired the trigram key with another trigram, encoded the pairs of letters using a "bigram table", then sent the encoded bigrams. The table was changed roughly once a year.

DECRYPTION

The main part of decrypting the messages was done by the Bombes (I've read two different explanations for the name, which came from the earlier Polish version -- it ticked like a bomb, or it made a really loud sound when a piece fell off at a particular point in the computation). The basic idea of the bombes is that given a "crib" -- a guess at the contents of a part of a message, like the words "weather report" or something like that -- they would try to arrive at the settings of the Enigma by process of elimination; that is, by ruling out contradictions. Basically, you assumed a rotor order & starting point, then it could test lots of plugboard settings at once. If it kept finding contradictions, you'd try new starting points (which were relative and could be used to deduce ring settings)... then new rotor orders... then new cribs. Remember that the plugboard connections, ring settings, & rotor order was fixed for the day for all messages, so finding those were a huge step towards getting all of the day's message.

So here's why Banburismus was useful -- it determined the identity of the "fast" (and sometimes middle) rotors. This would greatly reduce the number of assumptions they'd have to make about the day's Enigma settings & allowed them to use the bombes to find self-consistent solutions within a reasonable amount of time.

And now, the moment you've all been waiting for: how Banburismus worked. The basic idea was that it exploited a weakness of the design of the first 5 rotors (#'s 6-8 did not have this weakness): the letter (determined by the ring setting) at which it caused the leftward neighbor to rotate was different for the different rotors. This was their signature used to identify which was being used. The details...

- Your starting point: the current bigram table (they were figured out partly by deduction, partly by being "pinched" or captured), plus a day's worth of messages. You know the starting points of the messages, so with the table you can pull out the (encoded) trigram keys.

- Because the encoded trigram keys were all encoded using the same Enigma settings, those that differ only in the last letter correspond to initial rotor settings that are exactly the same except for the final ("fast") rotor. Likewise, those that differ in the second letter correspond to initial rotor settings that are different in the second rotor. Assume you have a bunch of pairs of messages that have trigram keys like this.

- So let's say we have messages with (encrypted) trigram keys "ALX" and "ALE". We therefore infer that the Enigmas generating these messages are identical except for an initial offset of the fast rotor (note that we don't know the actual value of this offset because we don't know the letters that "X" and "E" correspond to when decrypted).

- Thus, if we offset the messages, there should be a point at which the two (different) messages are being encrypted by machines in exactly the same state.

- This is where knowledge of letter frequencies in German comes in. We want to find the magic offset. So at each offset, we test which is the more likely hypothesis: H0, that the two different messages were encoded using two different Enigma settings, versus H1, that the two different messages were encoded using the same Enigma settings.

- The likelihood of getting two letters (i.e., one from each message) that match given H0 is just 1/26 -- the machine creates a uniform distribution of letters. However, the likelihood of getting a match given H1 is larger (about 2/26 for German.. it's because some letters are more common than others). From this it is straightforward to calculate the logLR of H1 vs H0 for matches and non-matches. From this, you can figure out the number of characters you need to expect a logLR of around 2 bans (100 to 1 odds) -- it's about 400 characters.

- Banburismus was just a way of counting up the numbers of matches in these pairs of messages (i.e., a quick way of estimating logLR). Each message was printed on a special sheet (in Banbury, of course) that has the message across the top, then rows of letters, with holes punched at the position where that letter was found in the message. This was just a way of visualizing the number of matches in pairs of messages -- you'd slide the sheets over (sliding one position corresponded to an offset of one of the fast rotor position), then count the holes that aligned in both sheets. Based on the formalism of the logLR calculations, they knew to expect (for certain length messages) a maximum number of matches (the "depth") that would correspond to a weight of evidence for H1. So if they found a clear maximum, they took it as evidence that they found the offset for which the two different messages were encoded using the same settings.

- Now they'd have this info for a bunch of message pairs, with which they could build a network of offsets. That is, they could get a series of encrypted letters that they knew corresponded to rotor positions that were offset by specified amounts; e.g.,

R -- 2 -- F -- 5 -- O -- 3 -- J -- 8 -- N

in other words, "R" was the encrypted version of some letter that was 2 positions away from the letter that was encrypted by "F", etc...

- Finally, they could line this set of offsets up with the alphabet, and find an alignment with no contradictions (they knew that the Enigma would not encrypt a letter with itself, and encryptions were symmetric so E -> K implied K -> E). This was called "scritching."

- Now the one final part: this last technique works only for changes in the given rotor (the fast one, in this case). That is, the small offsets cannot correspond to rotations that also cause the leftward neighbor to rotate (which would totally change the encryption). Thus, the set of offsets could not align with a place in the alphabet that contained the particular character at which point that rotor would cause the next one to rotate (the "turnover"). So the conclusion -- because the turnover was different for the first five rotors, when they got to this point they could often identify which rotor must have been used to generate the set of offsets.