INF247 Mandatory Assignment 1: G-Shcreiber Known Plaintext Attack

Eirik D. Skjerve

2024

1: Known plaintext attack against its internal settings

1. Find cabling and 0-1 distribution on first five wheels:

   - Find positions in ciphertext where ciphertext 5-bit group is "00000" or "11111". Then we know input to the relay box will also be "00000" or "11111" respectively. Since we know plaintext at those positions, we can retrieve the first 5-bit group by xor-ing the relay input ("00000" or "11111") with the plaintext code at that time.
   - Next we find the correct cabling from the 5-bit group to the wheels. For each position in the 5-bit group (1,2,3,4,5), we find a pair of 5-bit groups, and check if the bit is different when their times are congruent modulo period of a wheel. If the bit is different for a period p, then we eliminate the possibility of a cable between that bit position and the wheel with period p. We do this for every wheel/period and every bit position. In the end, if plaintext/ciphertext is long enough, we see that there is only one possible cabling between each bit position and a wheel.
   - When we know the cabling for the first 5-bit group, it is easy to reconstruct the 0-1 distribution on the relevant wheels, by going through the plaintext/ciphertext, and inserting into the wheels the correct bit on the correct position on the wheel.
   - At this point in my implementation, not all bits for the first five positions were known. To solve this, I used the fact that if there is one unknown bit in the first 5-bit group at the same time there is a weigth one or weight four 5-bit group in the relay input/output, the missing bit is possible to find. For example: plugboard at position i is (0 0 x 1 0) and plaintext encoding is (1 0 1 1 0) the result from xor'ing them together is (1 0 x 0 0). But since we know that this 5-bit group should be of weight one, the unknown in the plugboard must be 1 such that the unknown in the xor result is 0. By doing this for every such situation I was able to retrieve all remaining bits in the first five wheels.

2. Find control bits for relay box, cabling and 0-1 distribution on remaining wheels.

   - We locate times in the ciphertext where the code is a 5-bit group of weight one or weight four (e.g. "10000", "01111"). Since we know the bits in the first 5-bit group for those times, we also know the input to the relay box at that time.
   - By constructing a table of possible control-bits for specific 1/4-weight input/output pairs, we can decide some control bits for those moments. For example, a specific input/output pair can yield control bits "101xx". Then we can place the bits "1","0" and "1" with certainty, while the remaining bits ("x") are still unknonw.
   - Next we test periodicity on these bits aswell, like in the previous step. Then we should know the cabling from the last 5-bit group to the remaining wheels.
   - In my implementation a lot of bits for the last wheels were still missing at this point. To find more bits, I did the following two things:
     - Again, find positions where input/output to the relay box is of weight one or four that were retrieved through periodicity instead of directly from the table. By comparing the current values on the plugboard to the value in the table we can infer the value of a missing bit on the plugboard. For example: if the 5-bit group (x 0 1 1 0) on the plugboard at time i, we check the weights of the encoded ciphertext at time i. If the weight is one or four, we know that these control bits have an entry in the table from the previous steps. We find the entry, and compare. If, for example, the entry from the table is (0 0 1 x 0), we can update the plugboard at time i such that $x \to 0$ By doing this for all such cases, I was able to reconstruct a good portion of the remaining missing bits: 31/60 missing bits for wheel 9, 20/21 missing bits for wheel 1, 30/34 missing bits for wheel 5, 30/35 missing bits for wheel 8 and 42/44 missing bits for wheel 3. At this point, I was able to encrypt 85% of the plaintext correctly.
     - Another approach to retrieve more of the missing bits is the following: for those positions where encryption failed, if there is only one unknown control bit for that moment, I simply check which of "0" and "1" for the unknown bit yields the correct encryption, and update the plugboard and wheels accordingly. In total, this method retrieves 24 more unknown bits, and 26 missing bits remain. At this point, about 94% of the encryption is successful.

2: Attack in practice