Report

1) Found every possible C letter word given an alphabet, ex: aa ab ac...52, 53, 54, 55. For every word, I found its subset sum encryption given the table T and compared it against the encrypted password, if it matched I printed the decrypted word. To find the word, I treated the 5 byte password as a number and added 1 to it, like a C-base adder. The complexity for this solution is O(N*2^N), where N is the number of subsets. It is O(N*2^N) because there are 2^N subsets and to check each subset we have to sum N elements.

4-char : < 1s.
5-char: 40s.
6-char:31 mins.

2) Computed all possible floor(C/2) and C - floor(C/2) letter words concurrently, put them in two arrays. Merge them, use the encrypted merged pass as a key and the actual word as the value of a table with O(XlogX) lookup speed. The complexity of the algorithm for this solution is O(N*2^(C- floor(N/2)), where N is the number of subsets. The exponent is reduced to C - floor(N/2), because the possible passwords lengths are split, and the greatest split is C - floor(N/2).

5-char:
6-char:
8-char:
10-char:

3) I generated unique tables by creating a linearly independent matrix of the form:

00001
00010
00100
01000
10000

For the appropiate C, and then shuffled the rows so they are not always the same. Other tables enable the possibility of a carry happening with the sum and that causes the same sum to be possible with different passwords.