Classwork: Hashing and Salting Passwords

A hash function maps data, in our case a word, onto a number. The function, called a one-way function, is not one-to-one, so it cannot be reversed. For example, the function len maps a word to the number of letters it contains. So, len("Good") = 4, but so does len("word"). For this assignment you will make your own hash function.

1. Create a function which uses the values of the different letters, and perhaps their positions in the word along with 2 or 3 different mathematical operations (+,-,*,/,%) to map a string to a number. Describe the function here.

2. Find two short words which map to the same value using your function. Show your calculations for both words.

Letter	Value
A	1
B	2
C	3
D	4
E	5
F	6
G	7
H	8
I	9
J	10
K	11
L	12
M	13
N	14
O	15
P	16
Q	17
R	18
S	19
T	20
U	21
V	22
W	23
X	24
Y	25
Z	26

Salting the password

One problem with hashing password is that if the has algorithm is known and the has value is revealed, an attacker may be able to look up what password maps to that hash to determine the password. A list of precomputed hashes for dictionary words and commonly used passwords is called a Rainbow Table. To make this type of attack more difficult, we add a random value called a salt.

1. Append the letters AB to one of the words from step 2 on page 1 and recompute the hash for the new word.

2. Append the letters QU to the other word from step 2 on page 1 and recompute the hash for it.

Both the salt and the hash are stored instead of the password. To authenticate a password, we apply the salt to it and compute the hash. If the computed hash is the same as the stored hash, we accept the password as correct.