1. Drew Sadler, Cori Diaz, Abhinav Kumar
2. [asadler1@hopper3 asadler1]$ ./crack 4 4 abccBcrPOxnLU

Password: pass

1. [asadler1@hopper3 asadler1]$ ./crack 5 5 xyxMB6gxuiBZg

Password: ferry

1. [asadler1@hopper3 asadler1]$ time ./crack 5 5 nomatchingpass

Password not found

real 0m19.745s

user 1m26.356s

sys 0m0.011s

nomatchingpass hashed with salt ‘ab’ = ab.eoMC8IXGx2

1. [asadler1@hopper3 asadler1]$ time ./crack 2 5 nomatchingpass

Password not found

real 0m43.756s

user 1m25.207s

sys 0m0.023s

[asadler1@hopper3 asadler1]$ time ./crack 4 5 nomatchingpass

Password not found

real 0m26.087s

user 1m25.550s

sys 0m0.007s

[asadler1@hopper3 asadler1]$ time ./crack 8 5 nomatchingpass

Password not found

real 0m16.310s

user 1m26.135s

sys 0m0.013s

[asadler1@hopper3 asadler1]$ time ./crack 13 5 nomatchingpass

Password not found

real 0m7.094s

user 1m26.524s

sys 0m0.137s

[asadler1@hopper3 asadler1]$ time ./crack 26 5 nomatchingpass

Password not found

real 0m5.677s

user 1m33.201s

sys 0m0.118s

1. We ran through the entire combination space from aaaaa to zzzzz. Giving us 26^6 possible combinations. Time per thread comes out to be -

2 threads - (26^5)/44.984 = 264124.49 hashes/s

4 threads - (26^5)/27.906 = 425764.2 hashes/s

8 threads - (26^5)/17.239 = 689214.9196 hashes/s  
13 threads - (26^5)/6.799 = 1747418.16 hashes/s

26 threads - (26^5)/5.327 = 2230406.61 hashes/s

1. Our program is able to nearly half the time taken to compute when doubling the amount of threads from the previous iterations from 2-13, But when reaching from 13 to 26 threads it’s only able to reduce a second or so. Showing an exponential decrease in time, until the reduction starts to flatten out around 13 threads, due to hopper only containing 20 allocated threads to divide the work upon, meaning it can only improve so much after that amount. Since any thread passing the amount only contains 1 letter, we can’t divide the extra work any more, making the threads essentially useless. Also maybe doesn’t split to work exactly evenly causing the reduction to be not as much if evenly distributed
2. 2230406.61 hashes/s
3. 26^8 = 208,827,064,576 hashes
4. 93627.3519096 seconds ~= 26 hours

The benefit is that the more keys one uses, the longer it takes to brute force hack the password, and therefore makes the password more secure.

1. 1.5114 x 10^12 seconds ~= 47926.103 years
2. There are no known bugs in the program.

13. Extra-Credit:

Five-letter hashed passwords:

* abA.g8pU2Iffo = salts
* cdfnIXMyMCpPg = salts
* efgC/gw8PDKhs = password was not found
* ghLneTdBMxJP = password was not found
* ijSsXTgIC7QRU = salts
* klr7dT7cAODsk = donot
* mn4iWfK0m76t6 = skipp
* opjPsgpXaahxM = anyyy
* qr6ncfvfqecME = salts

Why salts are useful:

Salts help to hash similar/identical passwords in multiple different ways, so that way (as seen above) the same password chosen by multiple users will each have a unique hash. This is much more secure because this prevents one person’s password being hacked from compromising all other persons with the same or similar password.

\*\* ASCII

33 to 126 is the letters and symbols