Easy

How to find primes?

To determine whether n is a prime, test all divisors from 2 to $\sqrt{n}$

How to count prime divisors?

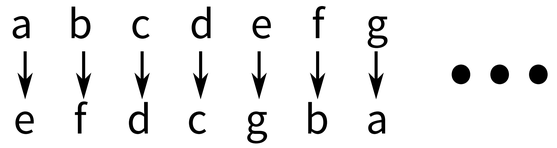
Let p be the password
Let $r \leftarrow p$
Let the result $m \leftarrow 0$
while $r \neq 1$:
  Let f be the next prime
  while f divides r:
    $r \leftarrow r / f$
    $m \leftarrow m + 1$

Now m is the # of factors. A password is valid if m is prime.

Medium

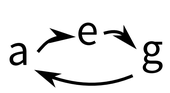
Obtain the square of the key

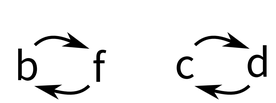
Create a map from plaintext chars to cyphertext chars.
Since the plaintext contains all characters, this map is complete.



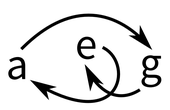
Compute the "square root"

Write the permutation as a list of cycles.

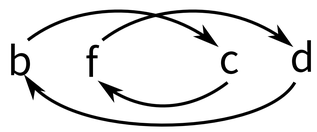




Reorder odd cycles: Arrows must point (length+1)/2 positions ahead



Interleave pairs of even cycles



Hard

Observe that the i-th letter of every text has been encrypted with the same letter of the key. Because of this, it becomes possible to launch frequency attacks. There are many possible solutions. Here's ours:

Preparation:
  Get a large body of text (e.g., text.in)
  Compute frequencies for substrings up to a certain length (say 3)

Stochastic optimization:
  Start with a random key
  Repeat:
    Replace a character in the key at random
    Decrypt, and compute a score:
      Each substring receives a score according to its frequency in normal text.
    If the score improved, keep the new key

Result after 20'000 iterations:

```
we_tan_factkvrthe_nuuzer_fiftien_r_th_quantum_himputerspbut
eulvr_wouldwtiobablyhbnjoy_thet_njn_his_theorerubecomespa_c
theqnice_thery_abouthheeyloq_ms_njn_cryptograpmzrs_can_triv
youqdont_wajxrto_buyhy_new_coqput_i_from_a_guyeqho_specyali
thehe_are_tssrtypes_wc_cryptokrapcp_the_one_thfn_works_qnd_
thehe_are_tssrtypes_wc_cyptogvaph_is_the_good_thes_and_ihe_
we_tan_see_plw_poinththere_thi_chdg_is_unhappyecf_a_wrocg_b
a_phivatekeudwncryptqln_schemi_stwkes_three_allirithms_uach
theqconcisewsoforddikqionary_shsisrdefnes_crypyi_as_thepart
remvmber_thewrlessonhthen_usirg_avjtream_ciphewunever_uhe_t
```


AdNovum
EPFL École polytechnique fédérale de Lausanne
hc²
Cryptography
open systems
PolyProg