A Tutorial on RSA with Big Primes

```
(00:21) gp > p=nextprime(random(10^30)) \\ picks a random prime less
  than 10^30
%1 = 738873402423833494183027176953
(00:21) gp > q=nextprime(random(10^25))
%2 = 3787776806865662882378273
(00:22) gp > setrand(2) \\ if you want to change the starting point
  of the random number generator
%3 = 2
(00:22) gp > p=nextprime(random(10^30))
%4 = 420829752590518623220964128931
(00:22) gp > q=nextprime(random(10^25)) \\ for security, it is better
  to choose primes of slightly different orders of magnitude.
%5 = 118661898673302028252493
(00:22) gp > n=p*q
%6 = 499364574606068826039374862361500298954266678741749 83
(00:22) gp > e=random(n)
%7 = 279118888978939199244278248027466418657477555312642 98
(00:22) gp > phin=(p-1)*(q-1)
%8 = 499364574606068826039370654062787774781301448817935 60
(00:23) gp > while(gcd(e,phin)!=1, e=e+1) \\ to guarantee relatively prime
(00:23) gp > e   \\ see, we got one bigger
%9 = 279118888978939199244278248027466418657477555312642 99
(00:24) gp > d=lift(Mod(e,phin)^(-1))  \\ lift turns the input into an integer,
  NOT an integer mod e any longer
%10 = 240043169197099479989228714388643077807097669434176 72
(00:25) gp > (e*d)%phin  \\ always good to check these things
%11 = 1
(00:25) gp > log(n)/log(27) \\ gives the number of letters we can encode
%12 = 36.816928756619335544571365 34
(00:25) gp > m=5+27*21+27^2*12+27^3*5+27^4*18 \\ m=EULER
%13 = 9673673
(00:28) gp > E(x)=lift(Mod(x,n)^e) \\ Encodes using above data
(00:29) gp > D(x)=lift(Mod(x,n)^d) \\ Decodes using above data
(00:29) gp > secret=E(m)  \\ This is the ciphertext
%14 = 241502327001029804344400835387857063219330186883962 12
(00:29) gp > D(secret) \\ With any luck, this should give back "EULER"
%15 = 9673673
(00:29) gp > factor(n)
%18 =
[11866189867330202825249 3 1]
```

```
[420829752590518623220964128931 1]

(00:42) gp > ##
  ***   last result computed in 28,190 ms.
```