Very Small Primes - Alan Yan

We use a segmented sieve to compile all primes from up to 10^9. Then, we try each one to see which one divides N. After finding the prime factorization of N, we can decode the message to get the number $896521733040669876562418477899743481 32$. Turning this into hexadecimal and then into ASCII, we get the flag "Craig Smells Bad." Code is down below.

Note: Segmented Sieve implementation taken from here.

```
import numpy
def modInverse(a, m):
	m0 = m
	y = 0
	x = 1
	if (m == 1):
	return 0
	while (a > 1):
	q = a // m
	t = m
	m = a % m
	a = t
	t = y
	y = x - q * y
	x = t
	if (x < 0):
	x = x + m0
	return x
def primesfrom3to(n):
	""" Returns a array of primes, 3 <= p < n """
	sieve = numpy.ones(n//2, dtype=numpy.bool)
	for i in range(3,int(n**0.5)+1,2):
	if sieve[i//2]:
		sieve[i*i//2::i] = False
	return 2*numpy.nonzero(sieve)[0][1::]+1
listPrimes = primesfrom3to(10**9)
N =
635579409671819260353775570338848482799357572403522160711496608486971009202
044622860862202607464723
e = 65537
M =
267692158982108203054528739047762713926141076955817170010967989855854594142
794334871913606841759263
```

```
smallPrime = 0
for i in listPrimes:
      if(N % i == 0):
      smallPrime = i
      break
largePrime = N//smallPrime
newMod = (smallPrime - 1) * (largePrime - 1)
privateKey = modInverse(e, newMod)
print(pow(M, int(privateKey), N))
```