Mantej Singh Rajpal Follow
Hacker. Ross Geller with a beard. NYU alum.
Nov 23, 2016 · 4 min read

The Best RSA

During the 1st weekend of November, I competed in Hack The Vote—a Jeopardy style competition modeled around hacking the general election. Please enjoy my elucidation of the 250-point cryptography challenge:

> *At his last rally, Trump made an interesting statement:*
>
> *I know RSA, I have the best RSA*
> *The more bits I have, the more secure my cyber, and my modulus is YUUUUUUUUUUUUGE*
>
> *We don't believe his cyber is as secure as he says it is. See if you can break it for us*

Ah, yet another can-you-break-RSA challenge; they never stale. Before we jump into Trump's fallacy surrounding the security of RSA, let's do some crypto 101. RSA, arguably the most well-known public-key cryptosystem, is used for ensuring the confidentiality of data in transit. Essentially, an RSA user (say, me) would generate two [large] prime numbers, p & q, and then publish a public key based on these two prime values. Anyone can use this public key to encrypt a message, and assuming the public key is large enough, only someone with knowledge of the two primes can decrypt the message [in a feasible amount of time]. Given modern computational power, 'large' suggests *at least* a 2048-bit public key. If someone were to successfully factor the public key and recover its prime factors, they would be able to decrypt *any* message.

> *p = 1024-bit prime number = kept private*
> *q = 1024-bit prime number = kept private*
> *N = p * q = 2048-bit number = public key*
> *If you can recover p and q by factoring N, you have successfully broken RSA.*

Side Note: N is actually a subset of the public key, and is referred to as the modulus. For the purpose of this write up, it's satisfactory to use 'public key' and 'modulus N' interchangeably.

Luckily for us, if Donald Trump generated N with the same regard in which he chose his cabinet members—it's going to be pretty crappy. In fact, a quick inspection yields something delightful:

> *N = 6423578730…[thousands of additional digits]…8388671875*

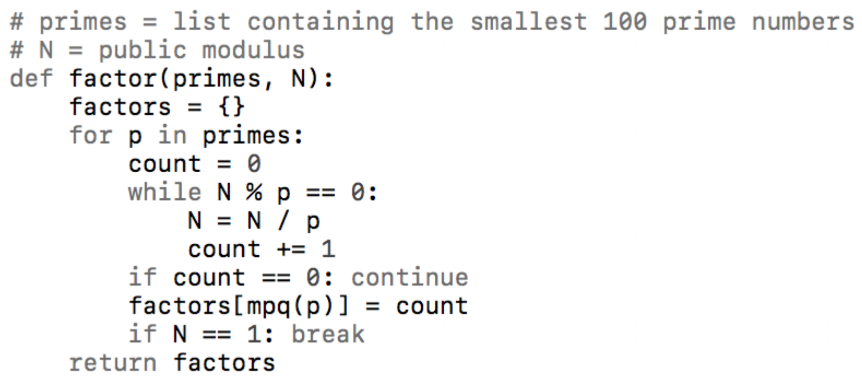
N is divisible by 5, which affirms that **5 is a prime factor**! Using 5 as one of the prime numbers when computing N is almost half as stupid as believing that climate change is a hoax perpetrated by the Chinese. Oh, Donald! Some prefatory division reveals that 3 and 7, in addition to 5, are also factors. I'm going to go out on a limb and say that N is a product of all the smallest prime integers (e.g. 3, 5, 7, etc). This is *exactly* the opposite of secure.

I want to take a 30 second detour and really shine some light around *small* primes. Hopefully I've driven home the point that generating an RSA public key using small prime numbers will result in complete failure of the cryptosystem. The security of RSA is contingent on not being able to determine the prime factors of the public key, and if the public key is generated using small prime numbers, they can be recovered by means of brute force. Now, yes, 3, 5, and 7 are obviously small—but let's take a look at the following prime number:

> *112,487,101,020,034,240,835,081,876,029,569,730,999,556,353,839,636,074,963,275,125,229,689,486,742,751*

For the purpose of generating secure RSA primes in the year 2017, the aforementioned prime integer is considered *very small*. Let that sink in for a minute. Now, laugh, because we're dealing with a public modulus N whose prime factors are 1, 2, and 3-digit primes.

Time to write some quick & dirty python. Five minutes later…



```python
# primes = list containing the smallest 100 prime numbers
# N = public modulus
def factor(primes, N):
    factors = {}
    for p in primes:
        count = 0
        while N % p == 0:
            N = N / p
            count += 1
        if count == 0: continue
        factors[mpq(p)] = count
        if N == 1: break
    return factors
```

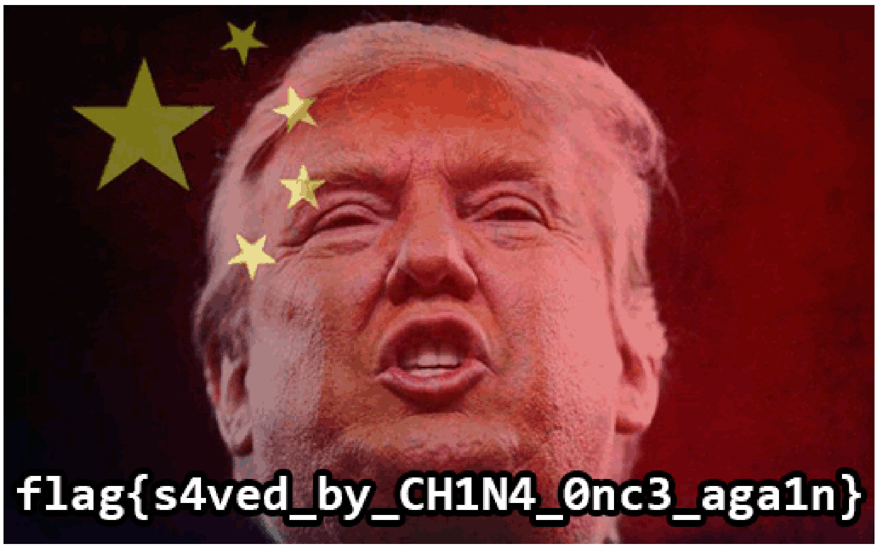
python function to recover N’s prime factors

> 3^{1545}
>
> 5^{1650}
>
> 7^{1581}
>
> …
>
> 241^{1564}
>
> 251^{1493}

Like I had thought, the prime factorization of N turned out to be every prime number from 3 to 251, inclusive. At this point, there are only a few things left to do:

1. Use Euler’s totient function to count the number of positive integers up to N that are relatively prime to N. This is referred to as φ(N).
2. Retreive the private decryption exponent by computing the modular multipliciate inverse of the public encryption exponent mod φ(N).
3. Decrypt the ciphertext.

Skimming past all the lower level details, I now attempt to decrypt the provided ciphertext. The resulting plaintext is a GIF file:



Voila! A flag is bestowed upon us. Submitting this flag awards us the state of Iowa, worth a whopping 250 points.