Using Pollard’s Rho Algorithm For Factoring to Crack RSA Encryption

Charlie Lu
College Park Scholars – Science & Global Change Program
Computer Science
clue@terpmail.umd.edu

College Park Scholars Academic Showcase, April 30, 2021

Introduction

Encryption is key to keeping online interactions secure. The challenge comes with creating a method that makes it so that cracking the encryption is hard even with knowledge of how the decryption work. RSA (Rivest-Shamir-Adleman) is one such encryption mechanism that by relying on the fact that factoring numbers is hard to do.

Activities:

I researched and coded up trivial, randomized, and Pollard’s Rho algorithm for factoring numbers. My research group and I looked at compared their execution times. For Pollard’s Rho algorithm, we looked to see which numbers it factored quicker. We then tested to see what RSA numbers Pollard’s Rho algorithm could crack.

How does RSA work?

To create a key:

Given L, generate two primes of length L: p, q.

Given p, q find N = pq and R = (p−1)(q−1).

Given R, find an e relatively prime to R.

Given R, e find d such that ed ≡ 1 (mod R).

Broadcast (N, e) publicly. This is now the public key.

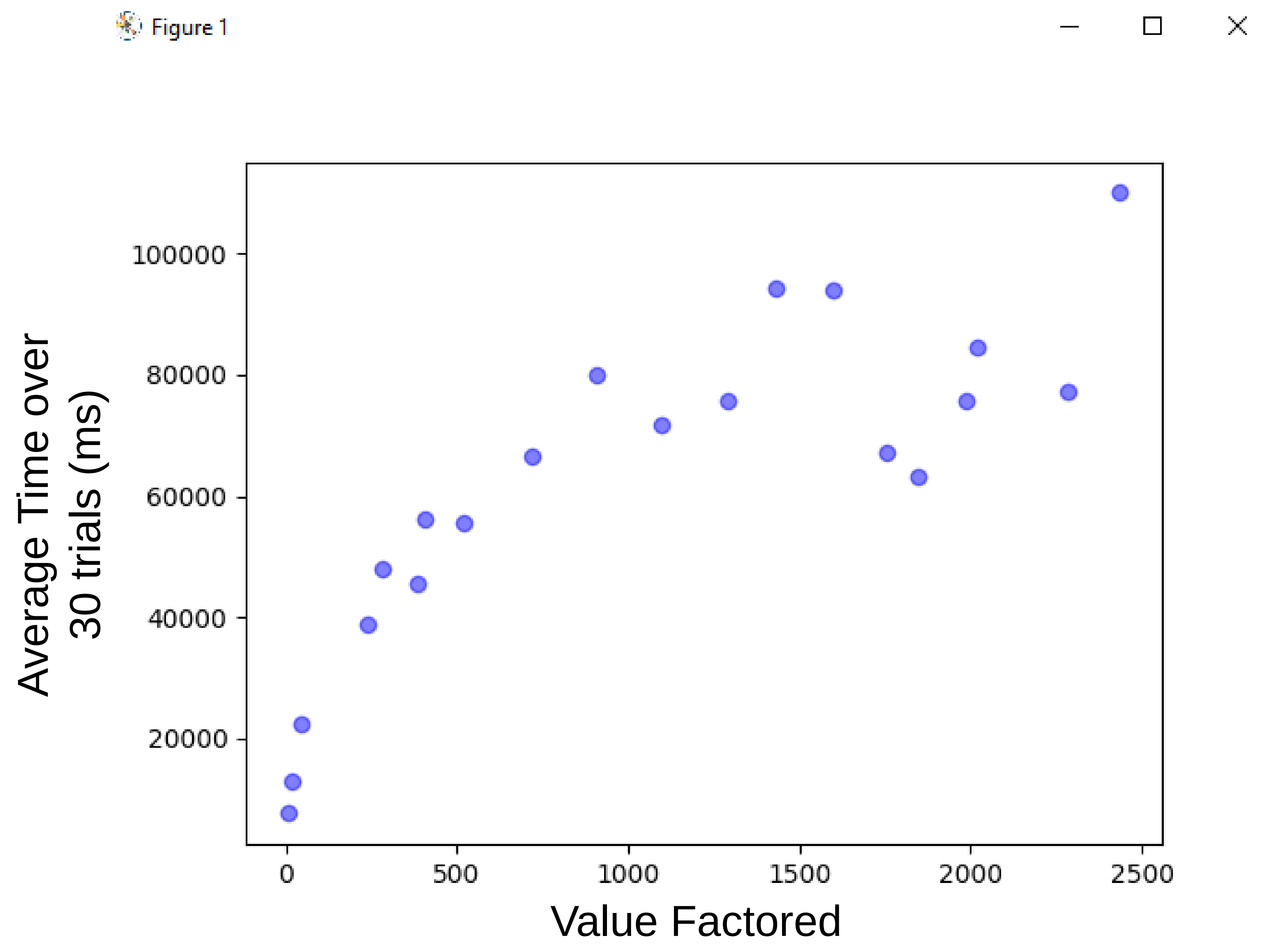
In order to send a message:

Given message m, compute m^e(mod N) = c, the coded message.

In order to decode the message:

Compute, c^d (mod N) = m

Time complexity of Factoring using P. Rho



Graph was created using code written in Python by me. The graph clearly shows that Pollard’s Rho algorithm the larger the number factored from it is. This shows that Pollard’s Rho algorithm is especially good for numbers with a lot of small prime factors.

Why is RSA hard to crack?

You need d in order to decode the message; however, d is generated from R, which is not in the public key!

So, to find R, one must be able to factor N. This is hard to do!

Much of the security depends on the fact that the fastest way to factor a very large number would still take an inordinately long time.

Site Information:

University of Maryland

Address: Online

Supervisor: William Gasarch

Research Group Acknowledgements:

My group originally consisted of Zongxia Li, Matthew Chan, and Jefferey Zhang. We split into two separate groups later to research different algorithms due to our varying levels of knowledge on mathematics and computer science. Jeffery and I comprised of the group aimed at researching Pollard’s Rho algorithm while Zongxia and Matthew Chan worked on another algorithm called quadratic sieve.

Future Work:

Future work would include further research into different methods of factoring and how they compare to Pollard’s Rho algorithm and its execution times. One such method is quadratic sieve, which would be the next step in terms of research for our research group on this topic.

Acknowledgments:

Professor William Gasarch

Dr. Holtz

Dr. Merck