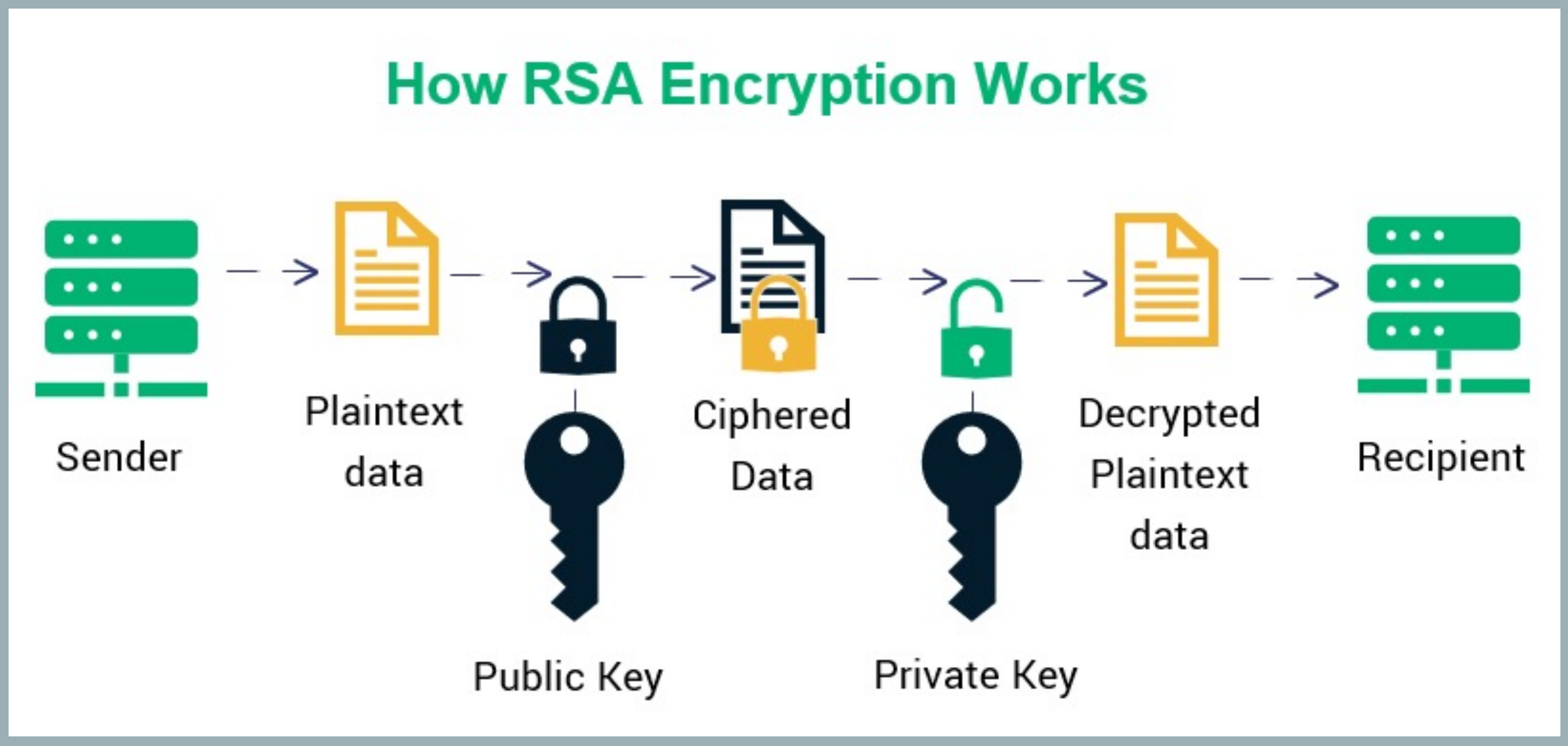
Breaking a Random Number Generator to Break a Cryptosystem

Jeffrey Zhang

College Park Scholars – Science & Global Change Program
Computer Science and Mathematics
jzhang45@umd.edu
College Park Scholars Academic Showcase, April 30, 2021

Introduction

Many of our current cryptosystems relies on the assumption that some problems have solutions that are easy to verify but hard to find (P vs NP). One of these problems include our RSA encryption system. We know that multiplying 2 prime numbers creates a composite prime number. However, it's much harder to determine the 2 prime numbers that make up composite prime numbers. It can take less than a millisecond the verify but almost 100 million years to solve.



One method of caption: A picture of how RSA works

Site Information:

Name of Site: Virtual

Address: Virtual

Your supervisor: William Gasarch

The site mission: Cryptography Research

The particular goals of the site you were at: Ensure students had a good understanding of cryptography and cryptographic methods

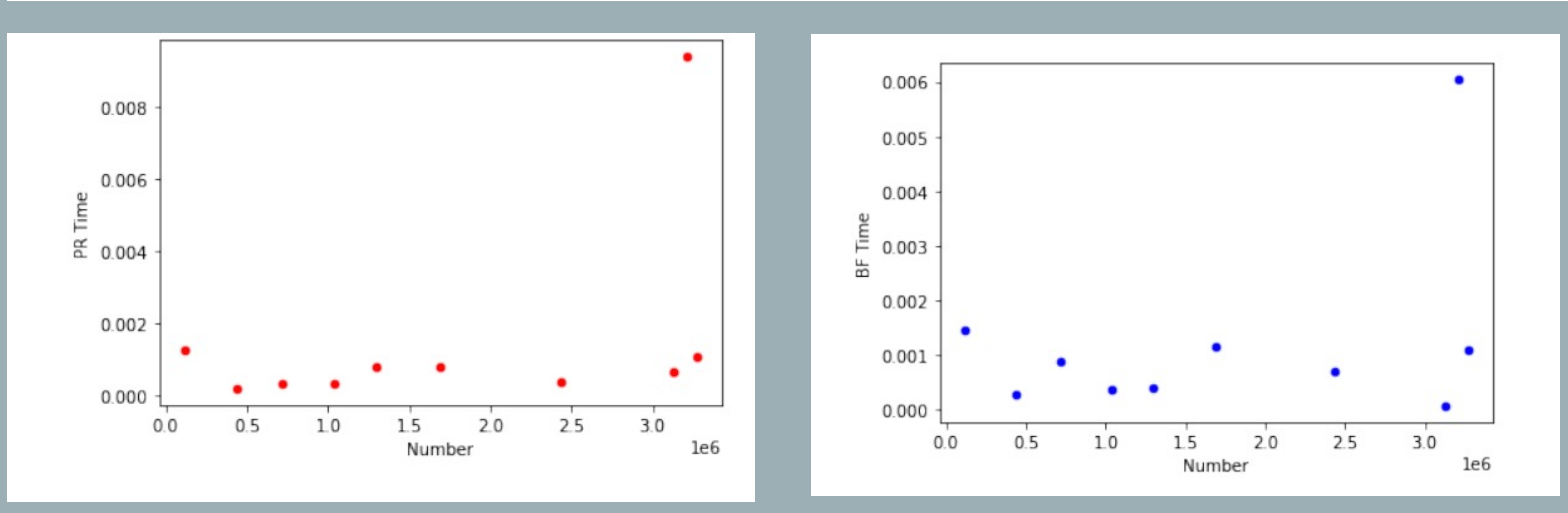
Issues Confronting Site:

I had trouble seeing eye to eye with the people in charge of implementing Quadratic Sieve. Some of them wanted to use the algorithm to determine 100-bit primes whereas my goal was to implement an RSA decryption algorithm. Hence there was a lot of confusion when we shared code.

Activities:

We first created an RSA algorithm that ensures that 2 parties can can communicate cryptically. We used several factoring algorithms to see how long it'll take to crack our RSA encryption. The 2 most notable ones were Pollard Rho and the classic "check every prime number until we find the right one" algorithm. Other algorithms attempted were Quadratic Sieve and Number Field Sieve. Had to learn unfamiliar topics to implement them.

Impact: We were able to see which algorithms cracked the encryption fastest at different circumstances. Thus, we can determine when it is appropriate to use which algorithm. For numbers below 4x10^6, there isn't much of a significant difference between any algorithm. However, the Brute Force algorithm was slightly faster than Pollard Rho. The encryptions were cracked in a few milliseconds.



All graphs were developed using Matplotlib from Python. The left shows the average speed (seconds) of the Pollard Rho Algorithm and the right it shows the average speed (seconds) of the Brute Force algorithm for cracking an RSA encryption (finding the 2 prime factors of a composite number)

Discussion:

Theoretically, Number Field Sieve should be the fastest factoring algorithm for any number greater than a google., followed by Quadratic Sieve. Both are slower for small numbers but the threshold of which it is slower than other algorithms is not exactly known from our experiment. If P = NP, then everything done in this research would be obsolete and our RSA encryption system would not be safe.

Future Work:

Might do research on the new O(nlogn) multiplication algorithm as it might improve the size of our RSA numbers and make our encryption system more secure.

Acknowledgments:

Drs. Holtz & Merck, Dr. William Gasarch

Scan the QR code to see full project