Comparing the Growth in Running Time of Different Factorization Algorithms

Andrew Chang[1], Evelyn Zhan[2]
[1]Henry M. Gunn High School, [2]Credo Semiconductor Inc.

Introduction

Cryptosystems are sets of algorithms that protect our digital data. One example is RSA, which uses an encryption scheme called public-key cryptography, in which a public key that everyone can access encrypts the data and a private key decrypts the data encrypted by the public key. In order to keep the private key secure, "RSA critically depends on the fact that a prime factorization of large numbers is not fast" (Estrada, Griffin & Sharma, 2011, p.6).

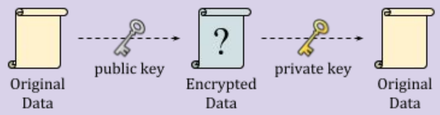

Figure 1. How RSA Applies the Complexity of Prime Factorization
The public key consists of a semiprime, or a product of two prime numbers, and the private key consists of its two prime factors.

However, this "fact" is not really a fact; it is conjecture that mathematicians may disprove at any moment. Once disproved, millions of system will be subject to huge risk. I aim to expand my knowledge of this crucial topic by comparing the growth in running time of different factorization algorithms.

Research Methodologies

Generate	Run	Record	Analyze
Generate semiprimes for test data. Ensure the difference between two prime factors decrease while the semiprime increases.	Run trial division, Fermat factorization, and Pollard's rho ten times for each semiprime.	Record the running times on spreadsheet and take the average of ten trials.	Create graphs and observe any correlations indicated in the graphs.

Algorithms

Let N be the factored number.

A. *Trial Division*
Divide N by x, where x is an integer iterating from 2 to N. If x divides N and x is a prime number, then x is a prime factor of N. For example if N = 247, divide 247 by each number from 2 to 247. Iterating from 2 to 247, notice that 13 and 19 divide 247, which means 13 and 19 are the factors of 247.

B. *Fermat Factorization*
For an arbitrary x, if y is an integer where $y = sqrt(x^2-N)$, then $x+y$ and $x-y$ are factors of N. For example, if N = 247 and arbitrary x = 16, then y = 9. Then, take the square root of 9, which is 3. Therefore, 16 + 3 = 19 and 16 - 3 = 13 are factors of 247.

C. *Pollard's Rho Algorithm*
Pollard's rho algorithm is a probabilistic method, meaning that a result is obtained based on chance, for factoring a composite number N by iterating a polynomial modulo N (Barnes, 2004, p.6).

Data and Findings

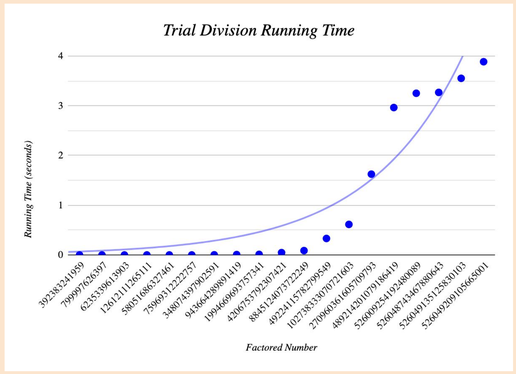

Figure 2. Trial Division
The factored numbers and running time displayed directly proportional exponential growth; as the factored number increased, running time increased.

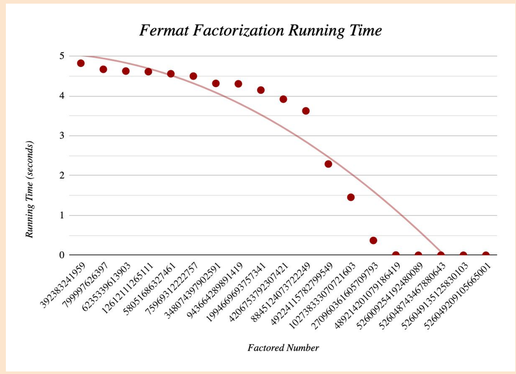

Figure 3. Fermat Factorization
The factored numbers and running time displayed inversely proportional relationship; as the factored number increased, running time decreased.

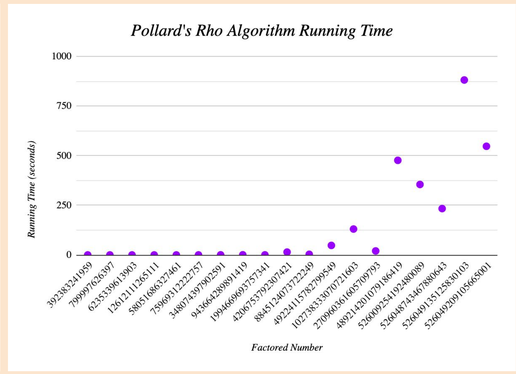

Figure 4. Pollard's Rho Algorithm
Generally, as the factored number increased, running time increased. However, the larger the factored number became, more sporadic the running times became. Thus, the trendline is not indicated.

Data Analysis

Trial division displayed a directly proportional relationship because it loops through all possible factors from smallest to largest, testing out each factor's divisibility. If the trend continues, RSA-768 (a number with 232 decimal digits) would take several years to factor. Fermat factorization displayed an inversely proportional relationship. Unlike trial division, while the algorithm loops through all possible factors testing them, it starts between the two factors and expands outward. Therefore, the running time of Fermat factorization will depend on how large the difference is between the two factors. Pollard's rho algorithm showed sporadic correlation because it is a probabilistic algorithm that finds the solution by chance. If N = factored number, the probability of finding the solution is $2/N$; thus, the larger the value of N is, the smaller the chance is of finding the solution more quickly.

Conclusion, Implication, and Next Steps

The data revealed that a larger factored number does not necessarily mean longer running time. Rather, the growth in running time depends on the algorithm.

This research was important because it led to a mathematical correlation which informed me that factoring a number with hundreds of digits would take several years. Understanding how the speed of different algorithms can be affected by various factors can contribute to broader perspectives on how quickly factorization could be achieved.

Next steps include analyzing more advanced algorithms such as number field sieving and elliptic curve algorithms to understand how factorization is further optimized. Despite the possibility of breakthrough and ongoing development of quantum computers, RSA is still considered a reliable cryptosystem because it has withstood a multitude of attacks. RSA keys have been extended repeatedly to prevent attackers from breaking through by waiting years to factor the keys.

Acknowledgements / References

Special thanks to Tarn Wilson and Evelyn Zhan for helping make this project possible.

References:

Barnes, C. (2004, December 7). Integer factorization algorithms. Retrieved April 20, 2019, from http://www.connellybarnes.com/documents/factoring.pdf

Blanda, S. (2014, March 30). RSA encryption – keeping the internet secure. Retrieved November 10, 2018, from American Mathematical Society website: https://blogs.ams.org/mathgradblog/2014/03/30/rsa/

Estrada, R., Griffin, J., & Sharma, V. (2011, May 12). *Prime factorization and its application to cryptography* [Lecture notes]. Retrieved December 7, 2018, from http://itcdland.csumb.edu/~restrada/files/Math170%20Project.pdf

Milanov, E. (2009, June 3). *The RSA algorithm*. Retrieved October 6, 2018, from https://sites.math.washington.edu/~mor row/336_09/papers/Yevgeny.pdf