Práctica 3 de Criptografía y Computación

Carlos Núñez Molina
Alessandro Zito
Gabriela Antolinez

1 Análisis tiempos Factorización

En esta sección vamos a analizar como tarda el algoritmo de Factorizacion por números grandes. Utilizaremos los tres metodologías que tiene ese algoritmo: Fuerza Bruta, Fermat y Ro de Pollard. Hemos hecho los análisis de frecuencias con números al azar y con números productos de primos. Empezaremos comentando con los números al azar. Hemos factorizando los números completamente (no encontramos un solo factor sino todos los factores), y vamos analizando cómo varía la eficiencia de los algoritmos según aumenta el número de cifras (en formato decimal, no binario). Además, hemos hecho experimentos para dos tipos de números: aquellos elegidos al azar y aquellos que son producto de dos primos de forma que el número resultante tiene n cifras.

Empezando con el algoritmo de Fuerza Bruta, sabemos que este algoritmo va intentando hasta que no factoriza el numero. Podemos ver en la primera imagen, El algoritmo es capaz de factorizar números de hastas 28 cifras, a partir de los cuales ya tarda demasiado. Su eficiencia es exponencial en el número de cifras, llegando a tardar 10^2 segundos cuando el número tiene 28 cifras." de 10^2

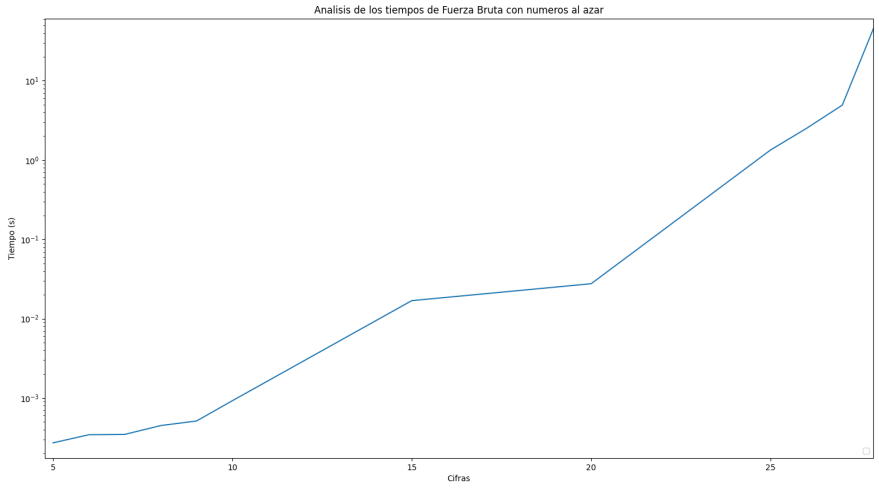

Figure 1: Análisis de los tiempos de Fuerza Bruta con números al azar

Después siguiendo con el algoritmo de Fermat, sabemos que este algoritmo va a resolver la ecuación $y = x^2 - n$ para factorizar el numero. Podemos ver en la imagen que este algoritmo es el peor de los tres, ya que solo es capaz de factorizar en un tiempo razonable números de hasta 10 cifras. El algoritmo es el peor de los tres por cómo factoriza los números: intenta factorizar el número de la forma $(x-n)*(x+n)$. Así, estos dos factores tienen que estar próximos entre sí (y de la raíz cuadrada del número a factorizar) para que el método funcione bien (si no lo están, tarda mucho). En los números elegidos al azar esto no pasa, ya que suelen tener divisores pequeños, y por eso funciona tan mal.

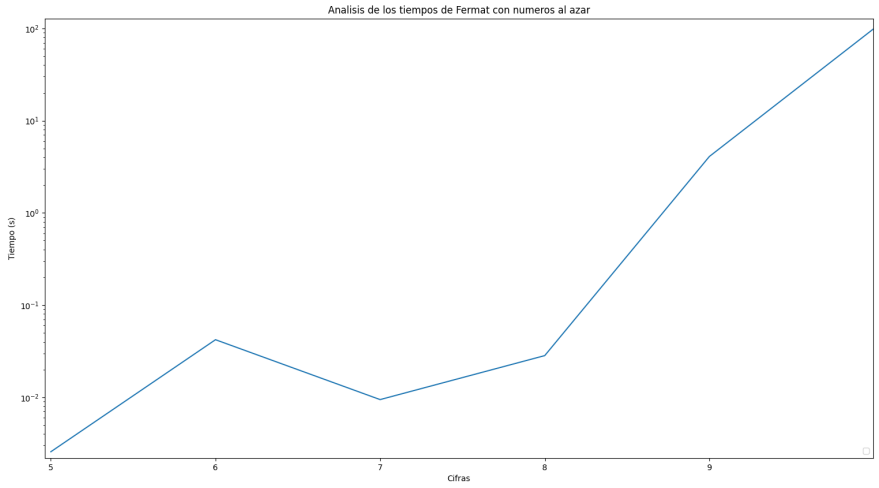

Figure 2: Análisis de los tiempos de Fermat con números al azar

El ultimo de los tres el algoritmo de Ro de Pollard, que tiene el nombre de el matemático que lo inventó. Se trata de construir una sucesión $x_1, x_2, ..., x_n$ y encontrar dos términos de la sucesión x_i, x_j tales que $mcd(x_i - x_j; n) \neq 1$. Podemos ver que este algoritmo es el mas eficiente de los 3, porqué se tardará "solo" 100 segundos con numerosa de 42 cifras!

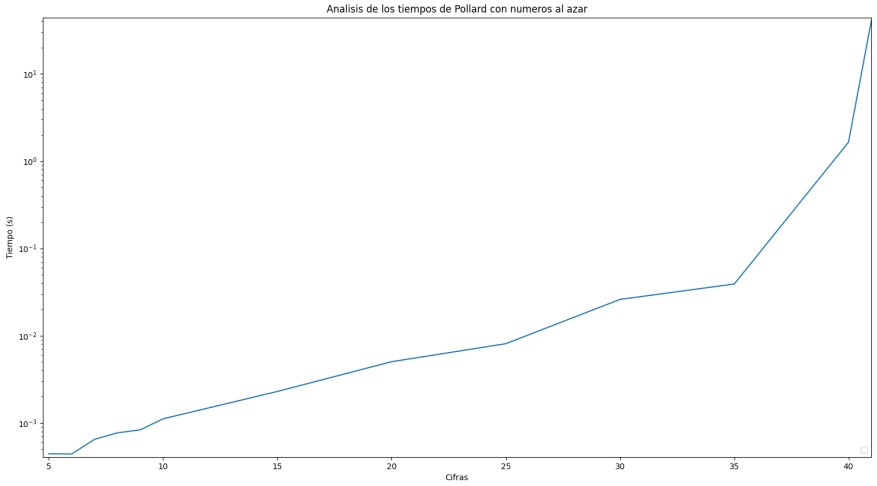

Figure 3: Análisis de los tiempos de Pollard con números al azar

Haciendo los mismos test con números productos de dos primos, podemos ver como Fuerza bruta y pollard tardan más porque los números son más difíciles de factorizar: tienen solo dos factores y son más grandes que cuando los números se eligen al azar. Sin embargo, fermat tarda menos con este tipo de números porque ambos factores están más cerca entre sí y de la raíz cuadrada del número. El mejor sigue siendo pollard que factoriza hasta las 28 cifras, pero que ahora fuerza bruta y fermat son más o menos igual de eficientes, aunque fuerza bruta sigue siendo mejor que fermat factorizando 20 cifrad y fermat 18 cifras

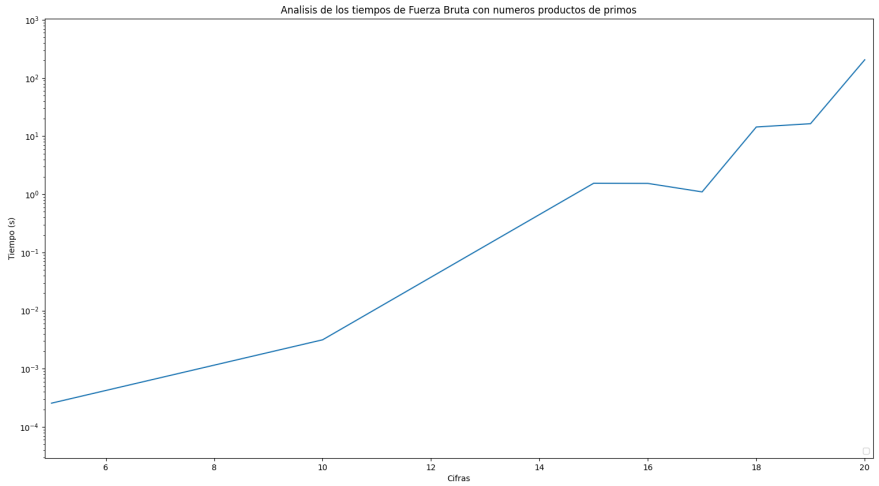

Figure 4: Análisis de los tiempos de Fuerza Bruta con números productos de dos primos

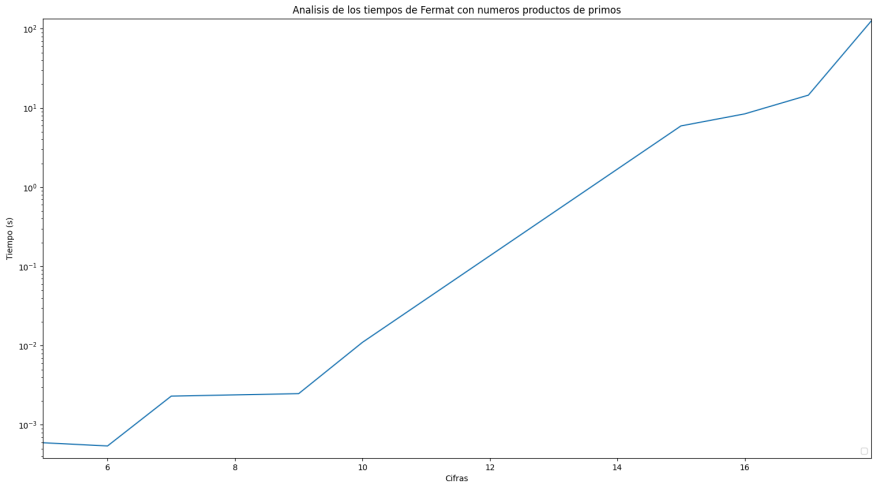

Figure 5: Análisis de los tiempos de Fermat con números productos de dos primos

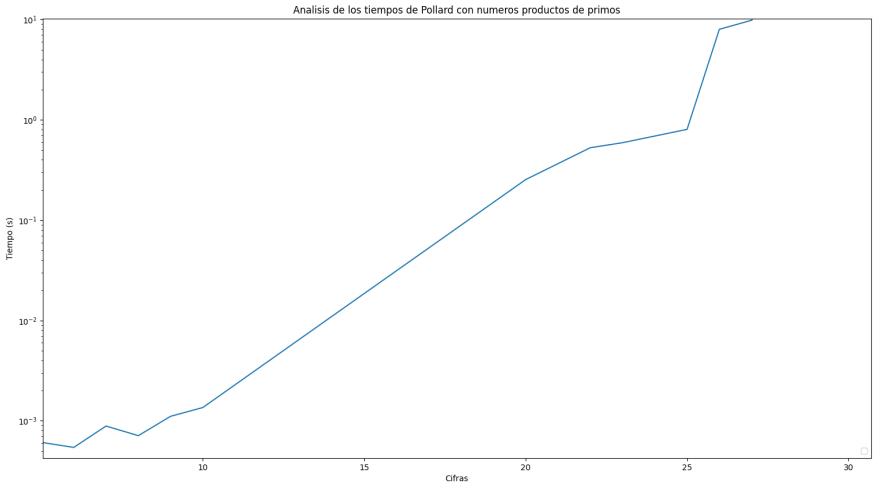

Figure 6: Análisis de los tiempos de Pollard con números productos de dos primos

2 Conclusiones

Creemos que el peor de los tres métodos es fermat: funciona peor que fuerza bruta para números al azar y cuando son producto de primos funciona mejor pero sigue siendo peor que fuerza bruta. El mejor de los tres métodos es pollard, ya que es el mejor sin importar si los números son escogidos al azar o producto de primos