El desarrollo de un cripto-algoritmo simétrico inteligente con aproximación a la secrecía perfecta de Shannon

2026 Daniel Alarcón Narváez

Universidad Autónoma de Querétaro

Facultad de Informática

El desarrollo de un cripto-algoritmo simétrico inteligente con aproximación a la secrecía perfecta de Shannon.

Tesis

Que como parte de los requisitos para obtener el Grado de

Doctor en Ciencias de la Computación

Presenta

MCC Daniel Alarcón Narváez

Dirigido por:

Dr. Fausto Abraham Jacques García

Co-dirigido por:

Dr. Luis Adrián Lizama Pérez

Dr. Fausto Abraham Jacques García
Firma aceptación avance a 100%

Querétaro, Qro. a 01 de mayo del 2026

Universidad Autónoma de Querétaro

Facultad de Informática

Doctorado en Ciencias de la Computación

El desarrollo de un cripto-algoritmo simétrico inteligente con aproximación a la secrecía perfecta de Shannon.

Tesis

Que como parte de los requisitos para obtener el Grado de

Doctor en Ciencias de la Computación

Presenta

MCC Daniel Alarcón Narváez

Dirigido por:

Dr. Fausto Abraham Jacques García

Co-dirigido por:

Dr. Luis Adrián Lizama Pérez

Dr. Fausto Abraham Jacques García
Presidente

Dr. Luis Adrián Lizama Peréz
Secretario

Dra. Sandra Luz Canchola
Vocal

Dr. Ricardo Chaparro
Vocal

Dr. Alfredo Acuña
Suplente

Centro Universitario, Querétaro, Qro.
Fecha de aprobación por el Consejo Universitario (mayo 2026)
México

Dedicatorias

A lo "Creador" por darme vida, salud y las oportunidades.

A mis padres y mi hermana, por su presencia constante incluso en la distancia,
y por enseñarme que la paciencia y la convicción
son formas silenciosas de amor.

A mi mismo, por no dejarme vencer, por creer en mi a pesar de la incertidumbre.

Agradecimientos

En primer lugar, agradezco a mi asesor, el Dr. Fausto, por aceptarme en esta línea de investigación, lo cual despertó en mí la pasión por un campo que desconocía, lo cual fue fundamental para el desarrollo de esta investigación. Del mismo modo, agradezco al Dr. Adrián Lizama, que, gracias a su vocación, me guió e impulsó a continuar en los momentos de nostalgia académica.

Agradezco también a los miembros del comité evaluador por el tiempo dedicado a la revisión de este documento, así como sus observaciones, las cuales contribuyeron significativamente a mejorar la calidad del mismo.

Extiendo mi reconocimiento a la UAQ que hizo posible la realización de este proyecto, así como a los espacios académicos que facilitaron su desarrollo.

Este trabajo no solo representa un resultado académico, sino también el reflejo de un proceso compartido.

Índice

Índice de Figuras

Índice de Tablas

Resumen

En el cifrado simétrico, el algoritmo y la clave secreta determinan el factor de seguridad. Este trabajo de investigación presenta la idea de crear un algoritmo de cifrado múltiple (7 veces) y separado por bloques. Para lograr esto, usaremos los métodos Hill Cipher y Gauss-Jacques. Sumado a lo anterior, nuestro aporte más significativo será la obtención de grandes claves secretas, que nos permitirán obtener como posible resultado una aproximación significativa a la secrecía perfecta de Shannon, así como la reducción de la complejidad computacional y la verificación de seguridad a través de mecanismos anti-bot como *code-breakers*.

Palabras Clave

AES, Block Cipher, Ciphertext, Clave secreta, Criptografía, Criptografía simétrica, Gauss-Jacques, Hill Cipher, Matriz inversa modular, Plaintext.

Abstract

In symmetric encryption, the algorithm and secret key determine the security factor. This research work presents the idea to create a multiple (7 times) and block-separated encryption algorithm. To achieve this, we will use the Hill Cipher and Gauss-Jacques methods. In addition to the above, our most significant contribution will be to obtain large secret keys, which will allow us to obtain as a possible result a meaningful approximation to the Shannon perfect secrecy, as well as the reduction of computational complexity and the verification of security through anti-bot mechanisms such as code breakers.

Keywords

AES, Block Cipher, Ciphertext, Cryptography, Gauss-Jacques, Hill Cipher, Modular Inverse Matrix, Plaintext, Secret Key, Symmetric Encryption.

Capítulo 1

Introducción

1.1. Antecedentes y estado del arte

La criptografía comprende el conjunto de métodos y mecanismos utilizados para proteger información durante su transmisión o almacenamiento, evitando que terceros no autorizadas puedan acceder o interpretar los datos. En la criptografía moderna, existen dos tipos principales de cifrado: criptografía simétrica y criptografía asimétrica Stallings (2017).

En los esquemas de cifrado simétrico, emisor y receptor comparten una misma clave criptográfica, la cual se emplea tanto en el proceso de cifrado como en el de descifrado de la información. La seguridad de este método depende de que la clave compartida sea conocida únicamente por el remitente y quien recibe. Ejemplos de algoritmos de criptografía simétrica son DES (Estándar de Cifrado de Datos), AES (Estándar de Cifrado Avanzado) y el algoritmo simétrico de Hill-Cipher Paar y Pelzl (2010).

A diferencia del cifrado simétrico, la criptografía asimétrica emplea dos claves relacionadas matemáticamente: una pública, destinada al intercambio de información, y otra privada, reservada únicamente para el propietario del sistema.La clave pública se comparte con cualquier persona que quiera enviar un mensaje al propietario de la clave privada. El propietario utiliza su clave privada para descifrar el mensaje. Ejemplos de algoritmos de criptografía asimétrica son RSA (Rivest-Shamir-Adleman), Diffie-Hellman y ECC (Criptografía de Curva Elíptica) Paar y Pelzl (2010).

Ambos tipos de cifrado tienen sus ventajas y desventajas y se utilizan en diferentes situaciones. La elección del método de cifrado apropiado depende

de la naturaleza de la información a cifrar y de las necesidades de seguridad del sistema.

Este trabajo de tesis se enfoca en el cifrado simétrico, que es una forma de criptosistema en el que el cifrado y el descifrado se realizan utilizando la misma clave. En otras palabras, los algoritmos que utilizan una clave compartida se conocen como algoritmos simétricos. La figura 1.1 ilustra el cifrado básico con llave simétrica. También se conoce como cifrado convencional. Los dos tipos de ataque a un algoritmo de cifrado son el criptoanálisis, basado en las propiedades del algoritmo de cifrado, y el de fuerza bruta, que implica probar todas las claves posibles. De manera general, un sistema de cifrado simétrico puede describirse mediante cinco componentes fundamentales:

1. **Plaintext**: Corresponde a la información original que será procesada por el algoritmo criptográfico antes de transformarse en texto cifrado.

2. **Algoritmo de cifrado**: Realiza substituciones o transformaciones sobre el *plaintext*.

3. **Llave secreta**: Es también una entrada del algoritmo. Dependiendo de la llave que se proporcione, será la salida que obtendremos.

4. **Ciphertext**: Este es el mensaje o dato codificado que se obtiene del *plaintext* y la llave secreta como salida.

5. **Algoritmo de descifrado**: Este es esencialmente el algoritmo de cifrado que se ejecuta a la inversa. Toma el *ciphertext* y la llave secreta y produce el *plaintext* original.

En el cifrado simétrico se usan dos técnicas: las de sustitución y/o transposición. Las técnicas de sustitución mapean los elementos del *plaintext* en elementos de *ciphertext*. Las técnicas de transposición transponen sistemáticamente las posiciones de los elementos del *plaintext*. Para la propuesta de este trabajo, nos enfocaremos en una técnica de sustitución llamada *Hill Cipher*. Debajo se encuentra el listado de las técnicas más comunes de la literatura William (2006).

- *Caesar Cipher*
- *Monoalphabetic Ciphers*

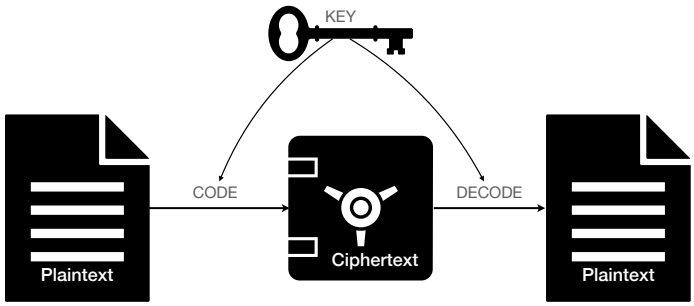

Figura 1.1. Encriptación de llave simétrica (Fuente: Diseño propio).

- *Playfair Cipher*
- *Hill Cipher*
- *Polyalphabetic Ciphers*
- *One-Time Pad*
- *Advanced Encryption Standard (AES)*

Actualmente, AES es considerado uno de los estándares de cifrado simétrico con mayor adopción en aplicaciones comerciales, industriales y gubernamentales. Mencionamos esto porque nuestro algoritmo propuesto pretende ser igual o más seguro, con la diferencia de que nuestro algoritmo requerirá un menor costo computacional. El Instituto Nacional de Estándares y Tecnología de EE. UU., en octubre de 2000, seleccionó el cifrado de bloques Rijndael como el Estándar de Cifrado Avanzado (AES) Daemen y Rijmen (2020).

Debido a su relevancia dentro de la criptografía moderna, el algoritmo Rijndael ha sido ampliamente estudiado en distintos escenarios relacionados con seguridad, rendimiento computacional e implementación en hardware. Diversos trabajos han evaluado su comportamiento frente a ataques criptográficos, así como las propiedades internas del algoritmo. Por ejemplo, Ferguson y cols. (2000) analizaron la seguridad de Rijndael mediante diferentes aproximaciones de ataque y estudiaron ciertas características no

esperadas del cifrado. Además, introdujeron una técnica denominada *partial sums*, orientada a reducir el esfuerzo computacional requerido durante el proceso de ataque.

De igual forma, se han realizado comparaciones experimentales entre Rijndael y otros algoritmos de cifrado por bloques. En este contexto, El Fishawy y Zaid (2007) evaluaron el desempeño de RC6, MRC6 y Rijndael mediante el cifrado de distintos tipos de imágenes digitales. Para el análisis consideraron métricas relacionadas con desviación máxima, correlación, diferencia de píxeles, tiempo de cifrado y rendimiento general del sistema, concluyendo que Rijndael presentó los mejores resultados entre los algoritmos evaluados.

Por otro lado, algunas investigaciones se han enfocado en la optimización del algoritmo para dispositivos móviles y sistemas embebidos. En ese sentido, Lee, Park, Kim, Lee, y Jun (2010) propusieron un esquema de cifrado y descifrado basado en Rijndael incorporado en terminales móviles, tales como teléfonos celulares, asistentes digitales personales y tarjetas inteligentes. La propuesta estuvo orientada a disminuir el tamaño del procesador criptográfico y reducir el tiempo necesario para las operaciones de cifrado y descifrado, manteniendo al mismo tiempo altos niveles de velocidad y seguridad.

Asimismo, existen trabajos orientados a implementaciones sobre hardware reconfigurable. Por ejemplo, Srinivas y Akramuddin (2016) desarrollaron una arquitectura basada en FPGA utilizando dispositivos Xilinx Virtex-7 para la implementación de AES Rijndael en sus variantes de 128, 192 y 256 bits. El diseño fue construido mediante tablas de búsqueda precalculadas (LUT), logrando una arquitectura menos compleja y con mejoras importantes en métricas relacionadas con latencia, rendimiento, velocidad de procesamiento, área y consumo de potencia.

El presente trabajo toma como base el método Hill Cipher, propuesto originalmente por Lester Hill en 1929, debido a su enfoque algebraico basado en operaciones matriciales. El punto central en el que se enfoca este método, es en la manipulación de matrices. Las fórmulas para obtener el *Ciphertext* y nuevamente el texto original a partir de la matriz Key inversa y el *Ciphertext*, se muestran a continuación.

$$C = (K * P) mod 26 \tag{1.1}$$

Donde: C = *ciphertext*, K = Matriz Key, P = *plaintext*

$$P = (K^{-1} * C) mod 26 \tag{1.2}$$

Donde:

$$K^{-1} = \frac{1}{|K|} adjK \tag{1.3}$$

Una de las principales limitaciones del método Hill Cipher radica en la necesidad de calcular matrices inversas modulares, proceso que puede volverse complejo o incluso inviable para determinadas configuraciones matriciales. También usa el determinante de una matriz, el cual, para los casos de matrices muy grandes, es muy difícil calcular el valor del mismo debido a su complejidad computacional $\mathcal{O}(n!)$ 1.3 Acharya, Panigrahy, Patra, y Panda (2009) Acharya, Rath, Patra, y Panigrahy (2007).

El cálculo de matrices inversas y de inversos modulares representa uno de los procesos matemáticos más utilizados dentro de distintos esquemas criptográficos. Debido a ello, múltiples investigaciones se han enfocado en desarrollar métodos que permitan optimizar su implementación y reducir el costo computacional asociado. Por ejemplo, Asad, Marouf, Al-Haija, y AlShuaibi (2019) desarrollaron una unidad modular inversa de 128 bits implementada sobre un dispositivo programable utilizando el algoritmo euclidiano extendido. La arquitectura fue descrita mediante VHDL, mientras que las etapas de simulación y síntesis se realizaron utilizando las herramientas ModelSim y Quartus II, respectivamente. Los resultados obtenidos mostraron mejoras relevantes en términos de desempeño para aplicaciones criptográficas que requieren operaciones de inversión modular.

De igual manera, Bos (2014) propusieron modificaciones al algoritmo de Kaliski con el objetivo de calcular tanto la inversión modular clásica como la inversión de Montgomery bajo un esquema de tiempo de ejecución constante. Este enfoque resulta especialmente importante como mecanismo de protección frente a ataques de análisis de energía en criptografía de llave pública. Los resultados mostraron que, sobre arquitecturas ARM, el método propuesto supera a varios enfoques basados en el teorema de Fermat cuando se trabaja con módulos primos genéricos.

Por otra parte, Phiamphu y Saha (2018) plantearon una versión mejorada del Algoritmo Euclidiano Extendido para el cálculo del inverso multiplicativo modular y del símbolo de Jacobi. La propuesta fue implementada mediante lenguaje HDL y posteriormente comparada con arquitecturas tradicionales basadas en el algoritmo euclidiano extendido convencional, obteniendo mejores resultados en distintos parámetros de desempeño.

Asimismo, Seysen (2005) presentaron un algoritmo alternativo para el cálculo de la inversión modular, reemplazando el algoritmo euclidiano exten-

dido por una variante basada en el algoritmo euclidiano estándar. La propuesta trabaja con enteros de longitud doble respecto al módulo utilizado, permitiendo incrementar considerablemente la velocidad de procesamiento en aplicaciones criptográficas.

Finalmente, Bajard, Meloni, y Plantard (2005) analizaron las propiedades del Sistema de Números Residuales (RNS) y sus aplicaciones dentro de la criptología. En dicho trabajo se estudió particularmente el comportamiento de la inversión modular sobre GF(P), proponiendo un algoritmo euclidiano extendido basado en RNS y apoyado en un módulo de aproximación de cociente. Los autores destacan que este tipo de aritmética presenta ventajas importantes en términos de resistencia frente a ciertos ataques criptográficos, especialmente en aplicaciones relacionadas con RSA.

Con el propósito de optimizar el cálculo de matrices inversas modulares, en esta investigación se implementaró el método Gauss-Jacques, debido a su capacidad para reducir significativamente el costo computacional asociado a este procedimiento, representada por la Ec. 1.4. El método Gauss-Jacques tiene por objetivo principal el obtener matrices inversas modulares en el subespacio *Zn,* de tamaño variable. Esto, con una complejidad computacional de tiempo potencial f(n) = ϕ^n. De esta forma, se reduce el tiempo computacional de los métodos de Leibniz y Murray, siendo factorial y polinomial respectivamente.

$$K_m^{-1} = \{\sum_{i=1}^{s}\sum_{j=1}^{s}[-k_{j(i+1)} + (k_{ij}e(k_{ii}, m) mod\ m)] mod\ m\} \tag{1.4}$$

El pseudocódigo del método se muestra en el listado siguiente: García (2018) Jacques-García, Uribe-Mejía, Macías-Bobadilla, y Chaparro-Sánchez (2022).

1. Se declara el tamaño de la matriz Key
2. Se crea la matriz Key con números aleatorios
3. Escoger un módulo primo
4. Escribir la matriz identidad junto a la matriz Key (lado derecho)
5. Buscar un número que, multiplicado por la posición 11 de la matriz y el módulo, te dé uno.

6. Después se aplica esa fórmula a todo el renglón

7. Se usa el pivote para reducir los elementos de su columna a cero y se aplica a todo la fila

8. Se realiza nuevamente desde el paso 5 para la posición 22 escalonando hasta tener la matriz identidad del lado izquierdo

9. La matriz del lado derecho es la matriz inversa

Con la fusión de estos dos métodos, el *Hill Cipher* y el Gauss-Jacques se espera lograr 3 puntos: un modelo de cifrado múltiple (7 veces), separados por bloques, y con una matriz key a lo más numerable. La Fig. 1.2 muestra un ejemplo de cifrado múltiple por bloque y la Fig. 1.3 muestra un ejemplo de descifrado múltiple por bloque. Por lo tanto, el objetivo general del trabajo será crear un nuevo modelo de cifrado simétrico capaz de lograr una aproximación a la secrecía perfecta.

Para comprender el concepto de secrecía perfecta, resulta indispensable abordar previamente los fundamentos de la teoría de la información propuesta por Claude Shannon, particularmente aquellos relacionados con entropía y probabilidad condicional. Donde se describe cómo obtener el valor de la información, es decir, predecirla; saber cuál es la probabilidad de que suceda una variable. Esto lo mide mediante la fórmula de la entropía (Ec. 1.5). Esta entropía va de la mano con las proposiciones condicionales y la probabilidad condicional. La primera define que si se tiene una proposición p llamada hipótesis o antecedente y la proposición q llamada conclusión o consecuente, al ser p y q proposiciones, la proposición se define como *si* p *entonces* q ($p \rightarrow q$) Johnsonbaugh (2005). Aunque para este trabajo usaremos un encadenamiento hacia atrás, es decir, *si* q *entonces* p ($q \rightarrow p$). Dicho lo anterior, ahora necesitamos saber cuál es la probabilidad que ocurra p dado que ocurrió q. A esto se le llama probabilidad condicional y se representa por $P(p|q) = P(p)$ que se leería como la probabilidad de que ocurra p dado q Walpole, Myers, y Myers (1999). Traducido a la criptografía se tiene $P(Pt|Ct) = P(Pt)$, es decir, la probabilidad de que ocurra el *plaintext* dado que ocurrió el *ciphertext*
Donde:
Pt= *Plaintext*
Ct = *Ciphertext*
P( ) = Función de probabilidad.

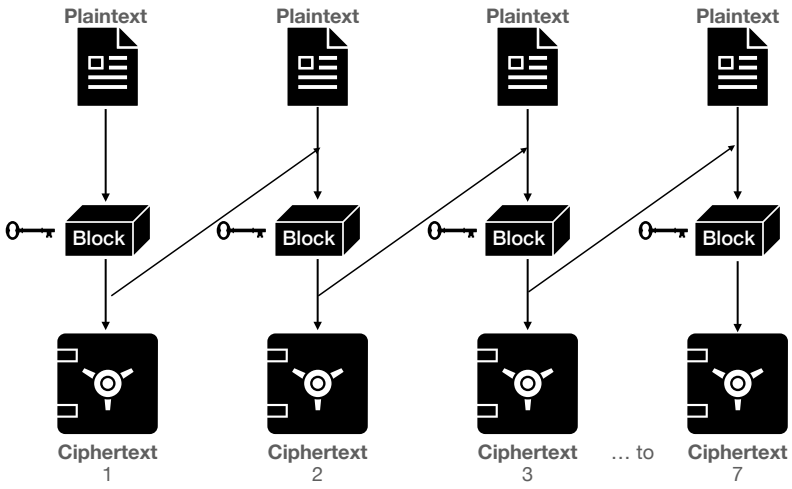

Figura 1.2. Ejemplo de cifrado múltiple por bloque (Fuente: Diseño propio).

Si se logra esto, entonces se logra la entropía. Dicha entropía es la base de la secrecía perfecta antes mencionada, pues conforme la entropía de una información disminuye, es más probable o fácil saber el resultado Shannon (1948). Por lo tanto, si se logra la entropía, se logra una aproximación a la secrecía perfecta.

$P(Pt|Ct) = P(Pt)$, en lenguaje natural "La probabilidad del *plaintext* dado el *ciphertext*, es igual a la probabilidad del *plaintext*".

$$H(x) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i) \tag{1.5}$$

Para completar el propósito general, se consideran los siguientes puntos principales:

1. Crear un método de cifrado de bloque para *Hill Cipher*.
2. Crear un método de cifrado múltiple 7 veces mayor.
3. Crear matrices a lo más numerables.

Diversas investigaciones recientes han buscado fortalecer y optimizar el método *Hill Cipher*, principalmente mediante la incorporación de técnicas

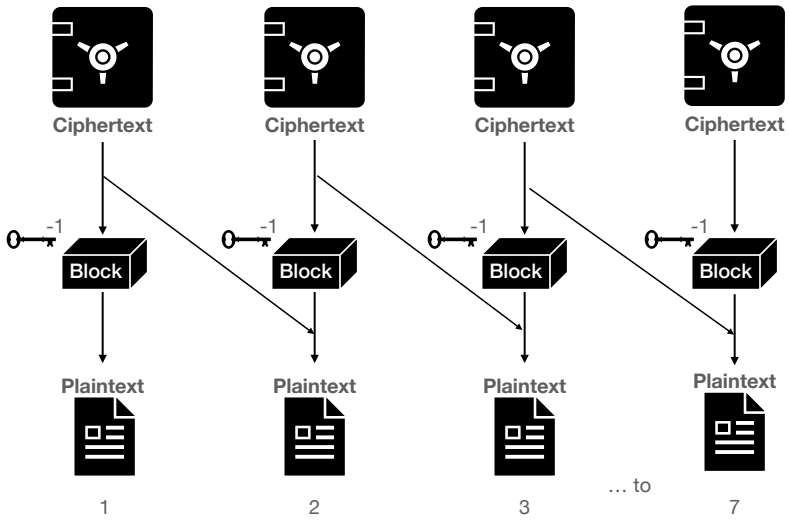

Figura 1.3. Ejemplo de descifrado múltiple por bloque (Fuente: Diseño propio).

complementarias orientadas a incrementar la seguridad, mejorar el rendimiento criptográfico y reducir vulnerabilidades conocidas. En este contexto, Paragas, Sison, y Medina (2019a) propusieron un esquema simplificado de cifrado por bloques basado en operaciones lógicas XOR, desplazamientos binarios y mecanismos de codificación Radix64. La propuesta incluyó procesos de generación de matrices llave, cifrado y descifrado implementados en C++. Los resultados experimentales mostraron mejoras en el *Avalanche Effect*, alcanzando un promedio cercano al 53 %, además de reducir redundancias presentes en los patrones del texto plano mediante el uso de Radix64.

De igual manera, Paragas, Sison, y Medina (2019b) desarrollaron una versión mejorada de *Hill Cipher* orientada a mitigar ataques de *known plaintext*. Para ello emplearon bloques de 128 bits, múltiples rondas de cifrado, encadenamiento de bloques y cuadros de sustitución hexadecimal. Asimismo, utilizaron SHA-256 para la generación de llaves criptográficas y operaciones XOR dentro del proceso de cifrado. Los autores reportaron un *Avalanche Effect* de 55.34 %, indicando una mejora significativa en la protección del texto cifrado respecto al esquema tradicional.

Por otra parte, Mahendran y Mani (2017) destacaron la importancia de seleccionar adecuadamente la matriz llave dentro de los sistemas basados en *Hill Cipher*, debido a que no todas las matrices poseen inversa modular. A diferencia de otros enfoques basados en generación aleatoria, los autores pro-

pusieron un método de construcción de llaves mediante procesos secuenciales y técnicas de permutación aplicadas sobre cifrados clásicos.

Asimismo, Khalaf, Abd El-karim, y Hamed (2015) presentaron una arquitectura triple de *Hill Cipher* implementada sobre FPGA para el cifrado de datos binarios, incluyendo imágenes, audio y video. La propuesta utilizó tres etapas consecutivas de cifrado con bloques de 128 bits y llaves de 256 bits, incrementando considerablemente la complejidad del sistema mediante múltiples rondas y el uso de diferentes llaves criptográficas.

En el ámbito del cifrado de imágenes, Putrie, Sari, Rachmawanto, y cols. (2018) combinaron el algoritmo *Hill Cipher* con técnicas de transposición de columnas para fortalecer la protección de imágenes RGB. Los autores evaluaron métricas como PSNR, MSE, SSIM y entropía, observando mejoras en uniformidad de histogramas y resistencia frente a diversos ataques criptográficos. La combinación de procesos de difusión y sustitución permitió obtener un esquema de superencriptación más robusto que los métodos convencionales.

Por otro lado, Saeednia (2000) analizaron las debilidades clásicas de *Hill Cipher*, particularmente aquellas derivadas de las transformaciones lineales utilizadas durante el proceso de cifrado. Como alternativa, propusieron mecanismos orientados a reducir el cálculo de matrices inversas mediante operaciones menos complejas, priorizando el fortalecimiento de la seguridad sobre aspectos relacionados con velocidad o eficiencia computacional.

Finalmente, Jangid, Mohmmad, Didel, y Taterh (2014) propusieron un esquema híbrido basado en *Hill Cipher* y criptografía de ADN para el cifrado de imágenes digitales. El método transforma las imágenes a escalas de grises y posteriormente a representaciones binarias, secuencias de ADN y aminoácidos. Los resultados mostraron incrementos en la entropía, disminución en la correlación y una distribución más uniforme en los histogramas, logrando además resistencia frente a ataques de tipo *chosen-ciphertext*, *known plaintext* y *chosen-plaintext*.

1.2. Justificación

El crecimiento acelerado de usuarios, dispositivos y servicios conectados a Internet ha incrementado considerablemente el volumen de información transmitida diariamente a través de redes digitales. La figura 1.4 DOMO (2020) muestra que actualmente se ha alcanzado el 59 % de la población

mundial. En la misma imagen podemos observar la cantidad de información que se carga o descarga en la red por minuto. Por poner algunos ejemplos, en whatsapp se comparten más de 46 millones de mensajes por minuto, más de 52 mil personas se conectan a Microsoft Teams por minuto, en VENMO se transfieren por minuto más de 200 mil dólares.

Si a la información anterior le añadimos la cantidad de dispositivos que estan interactuando con el internet. Entraríamos a un área llamada el internet de las cosas. Un modelo que facilita que los objetos físicos vean, escuchen, sientan y realicen trabajos al estar conectados con otros, intercambiando información. Un ejemplo de esto, es que hoy en día observamos televisores inteligentes, autos inteligentes, refrigeradores inteligentes, en fin; un sinnúmero de cosas que ya están interactuando con el mundo exterior por medio del Internet. Se espera que para el año 2022 al rededor de 212 billones de entidades cuenten con esta tecnología Birje, Kumbi, y Sutagundar (2017).

Con la cantidad de información que habrá cada vez más en la red, nos lleva a una frase dicha por el director del FBI, Robert Mueller, que dice: *"Sólo hay dos tipos de empresas: las que han sido hackeadas y las que lo serán"*. Por lo tanto, es de vital importancia tener un plan B de cuando ya han entrado a tu sistema, ese plan B es la criptología. Y en la criptología es donde nuestro trabajo se desarrolla pretendiendo aproximarse a la secrecía perfecta, es decir prácticamente indescifrable.

1.3. Descripción del problema

Esta investigación surge a partir de la necesidad de desarrollar mecanismos criptográficos que mantengan un equilibrio entre seguridad y eficiencia computacional, especialmente en entornos con recursos limitados, pasando de un costo $\mathcal{O}(n!)$ o $\mathcal{O}(n^a)$ a uno $\mathcal{O}(\phi^n)$. Con el desarrollo de este modelo se espera solventar este problema,mediante el equilibrio entre seguridad y bajo costo computacional. Permitiendo con estas métricas, una aplicación inmediata en el IoT. También el algoritmo sería adaptable para aplicaciones diversas, dependiendo a la tecnología HW que se quiera aplicar. Lo que haría al cripto-algoritmo inteligente, pues el cifrado por bloques y el cifrado múltiple derivado de las Ec. 2.1, 2.2, 2.3, 2.4 se aplicarán según el dispositivo HW.

Por ejemplo, las aproximaciones criptoalgorítmicas de llave pública tienen un alto costo computacional. Por lo que no cumplen con la eficiencia compu-

tacional y la seguridad deseada. Este alto costo, se debe también; a que los métodos que usa cada programa internamente, son complejos para su ejecución en la computadora. Haciendo complicado para el caso del método *Hill Cipher* el determinar vectores muy grandes, pues se vuelve un pActualmente, gracias a nuevas aportaciones como el método numérico Gauss-Jacques, se podrían fusionar estos dos últimos métodos, convirtiendo así las operaciones en más simples y amigables con el entorno computacional.n el entorno computacional.

El tener disponible este nuevo método numérico, el Gauss-Jacques; nos ayudaría a crear llaves a lo más numerables. Estas llaves de gran tamaño nos ayudarían a tener una aproximación significativa a la secrecía perfecta de Shannon. Debido a que si tenemos el espacio de mensaje, que son todas las posibles combinaciones de los elementos que se pueden obtener del dato a encriptar, y tenemos una llave (matriz) de ese mismo tamaño, que con el método Gauss-Jacques se podrían generar estas llaves; entonces, de este modo se alcanzaría la secrecía perfecta. Puesto que las probabilidades de representar un mensaje, son las mismas de los mismos mensajes antes de ser interceptados. Esta secrecía perfecta es posible, pero se requiere que el número de mensajes sea el mismo que las llaves posibles. Las cuales deben ser generadas en igual o mayor grado que el mensaje Shannon (1949).

Con la caracterización del criptoalgoritmo a desarrollar en este trabajo, que llevará el nombre tentativo de Hill-Alarcón-Jacques (HAJ). Se pretende solventar lo anterior. Definiendo como caracterización a la determinación de los rasgos característicos de nuestro modelo, como el uso de los dos métodos, la encriptación múltiple, la encriptación por bloque, entre otros. Así como representar en modelos computacionales los modelos matemáticos.

1.4. Planteamiento Teórico

1.4.1. Preguntas de Investigación

1. ¿Él HAJ resistirá los ataques de un mecanismo antibot de tipo *code breakers*?

2. ¿Él HAJ será tan ligero con respecto a la complejidad computacional y fuerte en seguridad?

3. ¿Él HAJ tendrá mejor rendimiento que el AES?

1.4.2. Hipótesis, Supuestos y/o proposiciones de investigación

Se plantea que la integración de los métodos *Hill Cipher* y Gauss-Jacques permitirá construir un esquema criptográfico con menor complejidad computacional y con características cercanas a la secrecía perfecta descrita por Shannon.

1.5. Objetivos

1.5.1. Objetivo General

Construir un nuevo método denominado HAJ, fusionando los métodos *Hill Cipher* y Gauss-Jacques para alcanzar una aproximación significativa a la secrecía perfecta.

1.5.2. Objetivos específicos

- Caracterizar el *Hill Cipher*.
- Caracterizar el Gauss-Jacques .
- Caracterizar la combinación del *Hill Cipher*y el Gauss-Jacques.
  - Adaptar el *Hill Cipher* para encriptarlo por bloques, donde se plantea usar una llave (matriz) significativamente grande.
  - Modificar el *Hill Cipher* para hacerlo múltiple a 7 veces.
- Validar el rendimiento del algoritmo HAJ mediante el costo computacional.
- Validar la fortaleza del método implementando un antibot de tipo *code breakers* que trate de descifrar un mensaje en el algoritmo HAJ.
- Extraer conclusiones de los experimentos realizados.

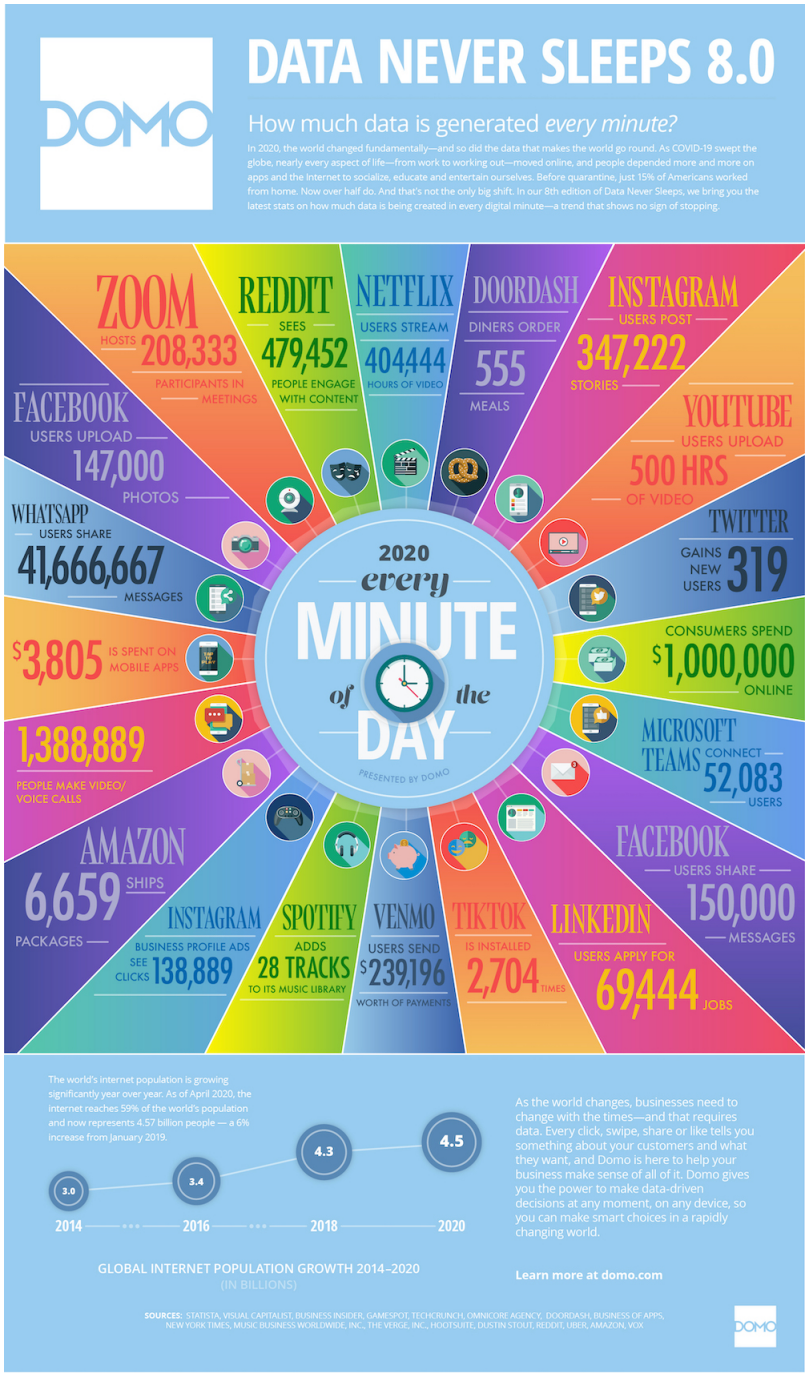

Figura 1.4. Datos generados por minuto en Internet (DOMO (2020)).

Capítulo 2

Marco Teórico

2.1. Metodología

En este trabajo se usarán dos metodologías: La Metodología de Investigación basada en el Diseño y la Metodología Experimental. La primera se utilizará debido a que su fundamento principal es el proponer un elemento nuevo para transformar una situación, además de que es un método relativamente nuevo y bastante utilizado en la literatura. La segunda es la base del cómo se llegará al conocimiento. Estos métodos, combinados con la metodología de Descartes y Bacon; permitirán llegar a la obtención del conocimiento. Reconociendo que por naturaleza Dios regala de modo innato el saber, pero al mismo tiempo; se mejora por medio de la experiencia, la observación y procurando eliminar todo prejuicio que pueda nublar el conocimiento GROBET y cols. (2004).

La metodología de la investigación basada en el diseño es un tipo de investigación orientada a la innovación educativa cuya característica fundamental consiste en la introducción de un nuevo elemento para transformar una situación. Este tipo de investigación intenta dar respuesta a problemas detectados en la realidad educativa recurriendo a teorías científicas o modelos disponibles con el fin de proponer posibles soluciones a estos problemas de Benito Crosetti y Ibáñez (2016). La metodología experimental implica la observación, manipulación y registro de las variables que inciden en un objeto de estudio. McDermott (2002). La figura 2.1, muestra de manera general la ruta metodológica a seguir en este trabajo.

Las ecuaciones 2.1 y 2.2 muestran las fórmulas que se proponen para

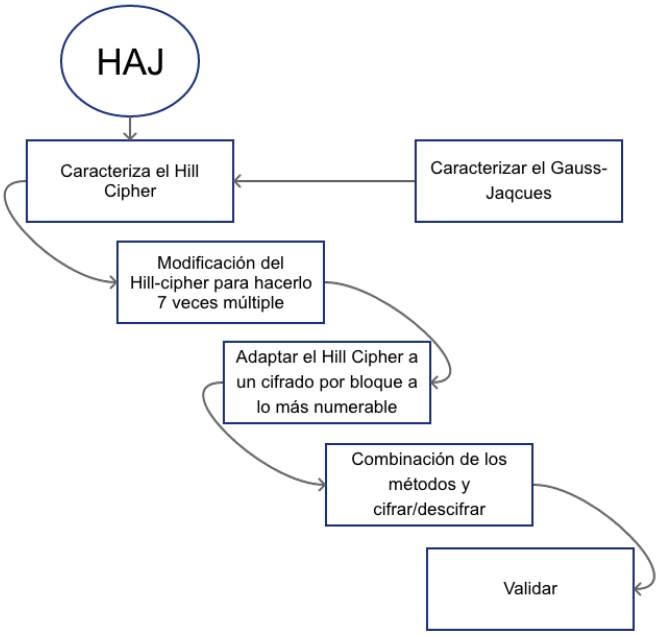

Figura 2.1. Ruta metodológica (Fuente: Diseño propio).

realizar el cifrado de bloques y obtener gradualmente un texto plano o un texto cifrado para cada bloque. Las ecuaciones 2.3 y 2.4 muestran las fórmulas para realizar múltiples cifrados y descifrados 7 veces. Parte de la propuesta de que se haga 7 veces es para observar el efecto en las variables. Esto parte de una analogía de refinación como la del oro que se realiza 7 veces. Es necesario mencionar que cuando se obtenga el resultado de las operaciones en los bloques, se realizará el "refinamiento" múltiple 7 veces. En otras palabras, se reemplazará el resultado de la Ec. 2.1 en la Ec. 2.3, así como el resultado de la Ec. 2.2 en la Ec. 2.4. Se espera que con la aplicación de encriptación múltiple y encriptación en bloque, así como con la matriz a lo más numerable, se logre un factor de seguridad alto.

2.1.1. Block

$$\bigcup_{i=1}^{n}\{(K_i P_i)\ mod\ m\} = C \tag{2.1}$$

$$\bigcup_{\mathrm{j}=1}^{n}\{(K_{m_i}^{-1}C_i)\ mod\ m\} = P \tag{2.2}$$

Cada fórmula en esta subsección puede ser defina por seis tuplas donde:

$\bigcup$ = Cada bloque creado de 1 a n
K = Matriz *Key*
P = *Plaintext*
C = *Ciphertext*
mod m = El modulo de un número primo
$K_{m_i}^{-1}$ = La matriz inversa

2.1.2. 7-HAJ (Múltiple)

$$\sum_{k=1}^{7}\{C_k\} = 7C \tag{2.3}$$

$$\sum_{\mathrm{l}=1}^{7}\{P_{\mathrm{l}}\} = 7P \tag{2.4}$$

Cada fórmula en esta subsección puede ser definida por tres tuplas donde:

$\sum$ = La suma del cifrado o descifrado múltiple de 1 a 7
P = *Plaintext*
C = *Ciphertext*

Nuestro algoritmo propuesto sigue los principios generales del cifrado de Hill. Implica dividir el texto plano en bloques, cada uno de los cuales se representa como una matriz. Estas matrices se multiplican por una matriz de clave secreta, lo que resulta en el texto cifrado. La innovación clave radica en el cálculo de la matriz inversa utilizando el método de Gauss-Jacques, que ofrece un enfoque más eficiente y modular en comparación con los métodos tradicionales. Al utilizar el cálculo de inversa modular, nuestro algoritmo asegura que el proceso de cifrado permanezca seguro incluso con un tamaño

de clave relativamente pequeño. Esta sección derivó en un articulo publicado en springer nature, ITNG 2021 (ver apéndice 7.9).

2.1.3. Cifrado de Hill

El algoritmo del cifrado de Hill es un algoritmo de cifrado de sustitución lineal que utiliza una matriz como clave para cifrar y descifrar el mensaje. Fue propuesto por Lester S. Hill en 1929 Hill (1929) y se basa en la matriz de clave, que es una matriz cuadrada invertible. El algoritmo del cifrado de Hill se utiliza en aplicaciones de seguridad de datos para proteger la información de ataques maliciosos. Este algoritmo se considera uno de los primeros algoritmos de cifrado simétrico y se ha utilizado en aplicaciones militares y gubernamentales durante décadas. Sin embargo, su uso ha disminuido debido a la llegada de algoritmos más seguros y eficientes Schneier (1996).

En el cifrado de Hill, el mensaje original se divide en bloques de longitud n, donde n es el tamaño de la matriz de clave. Cada bloque se multiplica por la matriz de clave y el resultado se reduce módulo un número primo para obtener el mensaje cifrado. El proceso de descifrado implica multiplicar el mensaje cifrado por la matriz inversa de la clave y reducirlo módulo el número primo.

A continuación, se presentan algunos experimentos relevantes y referencias relacionadas con el cifrado de Hill:

Experimento 1: Ataque de texto en claro conocido En este experimento, un atacante conoce algunos bloques de texto en claro y los bloques correspondientes de texto cifrado. Utilizando esta información, el atacante puede calcular la matriz de clave utilizando álgebra lineal y descifrar el resto del mensaje cifrado Singh y Srivastava (2014b).

Experimento 2: Ataque de texto cifrado conocido En este experimento, un atacante conoce algunos bloques de texto cifrado y los bloques correspondientes de texto en claro. Utilizando esta información, el atacante puede calcular la matriz inversa de la clave utilizando álgebra lineal y descifrar el resto del mensaje cifrado Agrawal y Gupta (2013).

Experimento 3: Análisis de frecuencia En este experimento, un atacante puede utilizar el análisis de frecuencia para descifrar el mensaje cifrado. El atacante puede utilizar la frecuencia de las letras en el mensaje cifrado para determinar las letras más comunes y, a partir de ahí, deducir la clave y descifrar el mensaje completo Singh y Srivastava (2014a).

Aunque el cifrado de Hill es un algoritmo sólido de cifrado, no es invulnerable a los ataques. Los experimentos mencionados anteriormente demuestran algunas de las vulnerabilidades del cifrado de Hill y la importancia de elegir una matriz de clave apropiada y asegurarse de que el número primo utilizado para la reducción módulo sea lo suficientemente grande. Además, es importante recordar que los algoritmos de cifrado son una de las muchas herramientas utilizadas en seguridad de datos, y la seguridad de los datos depende de una combinación de medidas de seguridad técnicas y procedimientos de gestión.

2.1.4. Método de Gauss-Jacques para el cálculo de inversa modular

El método de Gauss-Jacques desempeña un papel crucial en nuestro algoritmo simétrico de criptografía propuesto para el cálculo de la inversa modular. Este método nos permite calcular eficientemente la inversa modular de matrices de tamaño variable, lo que lo hace particularmente adecuado para aplicaciones criptográficas.

El método de Gauss-Jacques se implementa como una función separada, que toma dos entradas: la matriz 'K' y el número primo 'm'. Se asegura de que 'K' sea una matriz cuadrada y, si no lo es, genera un error. La función devuelve dos salidas: la matriz inversa modular 'InvMod' y la matriz identidad 'I'.

El algoritmo se desarrolla de la siguiente manera:

- La matriz original 'K' se amplía con la matriz identidad para formar una matriz aumentada.
- Para cada fila en 'K', se identifica un elemento pivote. Si el pivote es cero, se realiza una operación de intercambio de filas para asegurar un pivote distinto de cero.
- Se utiliza el algoritmo de Euclides extendido para calcular la inversa modular 'x' del elemento pivote con respecto a 'm'.
- La fila actual de 'K' se multiplica por 'x' módulo 'm' para obtener la fila actualizada.
- Para cada fila que no sea la fila actual, se realizan operaciones de fila para eliminar los elementos no cero debajo y encima del elemento pivote.

- Finalmente, la matriz resultante se divide en dos partes: 'InvMod', que contiene la matriz inversa modular, e 'I', que representa la matriz identidad.

2.1.5. Algoritmo extendido de Euclides

El algoritmo extendido de Euclides es una herramienta fundamental para encontrar el máximo común divisor (gcd) de dos números enteros en un espacio euclidiano. Se utiliza dentro del método de Gauss-Jacques para calcular la inversa modular del elemento pivote.

La función del algoritmo extendido de Euclides toma dos enteros positivos, 'a' y 'b', como entradas. Devuelve tres enteros, 'x', 'y' y 'd', tales que 'd' representa el gcd de 'a' y 'b', y 'ax + by = d'.

El algoritmo se desarrolla de la siguiente manera:

- Si 'b' es igual a cero, la función establece 'x' en 1, 'y' en 0 y 'd' en 'a' antes de devolver los resultados.
- De lo contrario, la función se llama recursivamente a sí misma con 'b' y 'mod(a,b)' como las nuevas entradas, almacenando los resultados en 'x1', 'y1' y 'd1'.
- 'x' se establece en 'y1', 'y' se establece en 'x1 - floor(a/b)*y1' y 'd' se establece en 'd1' antes de devolver los resultados.

El algoritmo extendido de Euclides es una extensión del algoritmo de Euclides básico y se utiliza ampliamente en criptografía, especialmente en aplicaciones que involucran aritmética modular, como el cálculo de inversas modulares.

2.1.6. Una ronda del HAJ

En lo descrito anteriormente se describe la generalización para crear y entender como funcionaría el algoritmo HAJ, es decir la sumatoria de los bloques y las 7 veces que se realizará. Ahora en los siguientes párrafos describiremos como pretendemos generar cada bloque, los bits que utilizará, el tamaño del mismo y como se generaran las llaves primarias y secundarias.

2.1.6.1. Obtener tamaño de cadena

Existen diferentes funciones para saber el tamaño de una cadena, donde cualquiera de ellas será utilizada para encontrar ese valor. Lo interesante en este apartado es que buscamos que dicho valor sea múltiplo de tres. Por lo que utilizaremos el módulo 3 de la longitud de la cadena para verificarlo, de ser necesario se agregarán 1 o 2 caracteres al *string* para convertirlo en múltiplo de tres.

Ejemplo del código:

$if s \neq 0$;
    a=3-s;
for i=1:a
    p(lp+i) = 'Q';
end
end

2.1.6.2. Sistema de soporte de decisión

El sistema de soporte de decisión (DSS por sus siglas en ingles), es un programa computarizado utilizado para respaldar resoluciones, juicios y cursos de acción en una organización o negocio. Un DSS filtra y analiza cantidades masivas de datos, compilando información completa que puede usarse para resolver problemas y en la toma de decisiones.

Nosotros utilizaremos como ya se mencionó el resultado del tamaño de la cadena, en conjunto con los requerimientos del equipo donde se ejecutará nuestro algoritmo. Esto con el objetivo de lograr la máxima seguridad y el máximo rendimiento para ese equipo. La Ec. 2.5 es representativa, mas no demostrativa.

$$block\_size = max(string.size(), computer.reqs()) \tag{2.5}$$

2.1.6.3. Generación de llaves

La llave primaria se generará mediante el método Gauss-Jacques, donde el tamaño de la matriz será determinada por el tamaño del bloque resultante

en los pasos anteriores. Sin embargo, para cada bloque se generará una llave primaria y para cada ronda de las 7 se generarán llaves secundarias o sub-keys para ir mejorando la seguridad en cada iteración.

Para las llaves secundarias utilizaremos el *left circular shift*. Un cambio circular en este caso a la izquierda es la operación de reorganizar las entradas en una cadena, ya sea moviendo la entrada final a la primera posición, mientras se desplazan todas las demás entradas a la siguiente posición, o realizando la operación inversa. Un desplazamiento circular es un tipo especial de permutación cíclica, que a su vez es un tipo especial de permutación.

Esta operación se utiliza en diversos ámbitos y diferentes algoritmos, un ejemplo de esto en la criptografía es en el algoritmo *Data Encryption Standard*, donde a partir del orden de la llave primaria se van recorriendo posiciones con respecto a una lista de corrimiento determinada con anterioridad. La tabla 2.1.6.3 es representativa mas no demostrativa.

Iteración	Desplazamiento
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	3
13	1
14	1
15	2
16	3

2.1.7. Proyección

A continuación se enumera la proyección de este trabajo, donde las partes inicial, intermedia y final son la prioridad para cumplir con la tarea.

1. Estudiar el estado del arte relacionado con *Symmetric Encryption, Advanced Encryption Standard*, Hill Cipher y toda la información relacionada con la criptografía que nos sea útil para lograr nuestro objetivo.
2. Realizar una publicación con la propuesta del nuevo algoritmo a realizar.
3. Desarrollar el método con las modificaciones propuestas.
4. Probar la funcionalidad del método desarrollado.
5. Realizar una publicación con los resultados encontrados hasta el momento.
6. Obtener el algoritmo final propuesto
7. Evaluar nuestro método con un algoritmo que verifique el factor de seguridad.
8. Realizar una publicación con los resultados finales obtenidos.

Capítulo 3

Caracterización de los métodos

Para demostrar la funcionalidad de nuestro algoritmo criptográfico simétrico propuesto, implementamos los procesos de cifrado y descifrado utilizando el código proporcionado. El algoritmo utiliza el cifrado de Hill con cálculo de la inversa modular mediante el método de Gauss-Jacques.

Para el cifrado, el algoritmo requiere la selección de un número primo, denotado como 'm', mayor que el valor ASCII más alto previsto para el cifrado. Además, se deben definir el número de matrices, 'num_matrices´, y el tamaño del bloque, 'block_size'. Las matrices se generan de forma aleatoria, asegurando que el determinante sea distinto de cero módulo 'm'.

El texto plano se obtiene del usuario y se procesa en bloques. Si la longitud del texto plano no es un múltiplo de tres, se aplica relleno para que sea un múltiplo. El proceso de cifrado implica multiplicar el bloque de texto plano con la matriz de clave correspondiente, asegurando una aritmética módulo 'm'.

El descifrado sigue un proceso similar, pero utiliza la inversa de la matriz de la llave calculada mediante el método de Gauss-Jacques registrado en MATLAB Cantón (2023). El texto cifrado se procesa en bloques y el descifrado se realiza multiplicando el bloque de texto cifrado con la matriz inversa módulo 'm'.

El texto plano descifrado resultante se muestra como la salida final del algoritmo.

3.1. Una ronda del HAJ

Algorithm 1 Criptografía simétrica con Hill Cipher

Entrada: Archivo de entrada, m
Salida : Archivo de salida

1 `Inicio()` Limpiar la pantalla Iniciar el cronómetro `LeerArchivo(`*Archivo, data*`)` p ← convertir datos binarios en cadena de caracteres lonp ← longitud de p a ← lonp mod 3 **if** $a \neq 0$ **then**
2 rm ← 3 - a **for** $i \leftarrow 1$ **hasta** *rm* **do**
3 Agregar el carácter 'h' al final de p
4 **end**
5 **end**
6 lonpL ← longitud de p Generar una matriz aleatoria k de tamaño lonpL-por-lonpL con valores entre 1 y m-1 Calcular el valor cifrado c utilizando la operación (k * p') mod m lonc ← longitud de c p ← cadena vacía Calcular la matriz inversa InvMod y la matriz identidad I utilizando el método de Gauss-Jacques con las matrices k y m Calcular el valor descifrado p utilizando la operación (InvMod * c) mod m `EscribirArchivo(`*ArchivoSalida, p*`)` Detener el cronómetro y calcular el tiempo de ejecución Mostrar el mensaje .[El] texto plano descifrado se ha guardado en el archivo: ArchivoSalida" Mostrar el texto plano descifrado Mostrar el tiempo de ejecución
7 `LeerArchivo(`*Archivo, data*`)` Leer el archivo de texto en formato binario y almacenar los datos en la variable data
8 `EscribirArchivo(`*ArchivoSalida, p*`)` Especificar el nombre del archivo de salida Abrir el archivo en modo de escritura en formato UTF-8 Escribir el texto plano descifrado en el archivo Cerrar el archivo
9 `Fin()` Fin del programa

Capítulo 4

Cifrado con llave de tamaño del plaintext

4.1. Algoritmo HAJ: Con llave == Plaintext

```
clc;
clear all;
close all
%feature('DefaultCharacterSet', 'UTF8');
m = 257;

% Iniciar el cronómetro
tic;
%p = input('ingresa el plaintext:', 's');
% Leer el archivo de texto en formato binario
Archivo = 'Himnos_Nacionales/Himno_Mexico.txt';
fid = fopen(Archivo, 'rb');
data = fread(fid);
fclose(fid);

% Convertir los datos binarios a una cadena de caracteres
p = char(data');

lonp = length(p);
```

```
% Asegurar que el texto plano sea una cadena múltiplo de 3 caracteres
a = mod(lonp, 3);
if a ~= 0
    rm = 3 - a;
    for i = 1:rm
        p(lonp+i) = 'h';
    end
end
%p

lonpL = length(p)


% Generar matriz aleatoria de tamaño n-por-m
k = randi([1, m-1], lonpL, lonpL);
%k
length(k)

% Cifrado
%c = mod(k * p', m);
c = mod(k * double(p)',m);

% Descifrado
lonc = length(c);
p = '';
[InvMod, I] = gauss_jacques(k, m);
p = mod(InvMod * c, m);

% Especifica el nombre del archivo de salida
ArchivoSalida = 'Texto_Plano_Descifrado.txt';

% Abre el archivo en modo de escritura en formato UTF-8
fid = fopen(ArchivoSalida, 'w', 'n', 'UTF-8');

% Escribe el texto plano descifrado en el archivo
fwrite(fid, p, 'char');

% Cierra el archivo
```

```
fclose(fid);

% Muestra un mensaje de confirmación
fprintf('El texto plano descifrado se ha guardado en el archivo: %s\n',
ArchivoSalida);
%
% % Detener el cronómetro y calcular el tiempo de ejecución
tiempo_ejecucion = toc;

% Mostrar el resultado del descifrado
fprintf('Texto plano descifrado: %s\n', char(p));

% Mostrar el tiempo de ejecución
fprintf('Tiempo de ejecución: %.2f segundos\n', tiempo_ejecucion);
% Medir uso de memoria
%mem_info = memory;
%mem_used = mem_info.MemUsedMATLAB; % Tamaño de la memoria utilizada por MATLAB
%en bytes
%fprintf('Uso de memoria: %.2f MB\n', mem_used / 1024^2);
```

4.2. Procedimiento experimental

Se realiza la experimentación con diez himnos nacionales los cuales se enlistan debajo:

1. Australia
2. Canadá
3. Gran Bretaña
4. India
5. Irlanda
6. Jamaica
7. Nueva Zelanda
8. Singapur

9. Sudáfrica
10. USA

Estos himnos se utilizaron para la primera parte de la experimentación, en este capítulo en específico la experimentación consiste en una sola ronda del algoritmo HAJ. Es decir solo se cifra y se descifra una vez. Con la particularidad de que se genera una matriz del tamaño del *plaintex*. La table 4.2.0.1 muestra la longitud que tiene cada himno nacional. Por lo que la llave, una matriz k sería del tamaño nxn donde n es igual a la longitud de cada himno.

Himno	**Longitud**
Australia	620
Canadá	318
Gran Bretaña	496
India	336
Irlanda	800
Jamaica	591
Nueva Zelanda	1190
Singapur	206
Sudáfrica	226
USA	1705

Tabla 4.2.0.1. Longitud de cada Himno.

Por ejemplo el himno nacional de USA cuenta con 1705 caracteres con todas las estrofas y coros actuales, sin hacerlo pasar por el algoritmo y hacerlo múltiplo. Por lo tanto se genera una matriz aleatoria K de 1705x1705 elementos que van del 0 al 256, valor que se multiplica por el *plaintex* para general el *ciphertext*, para este ejemplo particular aquí descrito; sería el himno de USA.

4.3. Resultados

Se realizaron treinta ejecuciones con cada uno de los himnos nacionales descritos en la sección 4.2. La tabla 4.3.0.1 es un ejemplo de cada una de las ejecuciones realizadas, las demás tablas se pueden encontrar en el apartado número 7.1 del Apéndice A.

Las tablas 4.3.0.2, 4.3.0.3 y 4.3.0.4. Muestran el promedio, la desviación estándar, el valor máximo y mínimo del tiempo de ejecución expresados en segundos, así como el uso de memoria representado en Mb. De las treinta ejecuciones que se realizaron a cada himno.

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	14.3504	5.9011
2	33	14.2019	5.9248
3	88	14.3617	5.9291
4	64	21.8156	5.9295
5	55	15.8586	5.9299
6	51	16.8142	5.9303
7	74	15.2743	5.9307
8	56	13.9836	5.9311
9	25	14.1974	5.9315
10	57	14.5327	5.9319
11	21	13.3327	5.9323
12	16	13.7034	5.9326
13	7	15.9926	5.9330
14	83	12.6617	5.9334
15	52	11.8213	5.9338
16	18	11.7951	5.9342
17	39	11.7608	5.9346
18	72	11.7796	5.9350
19	95	11.7669	5.9354
20	93	11.7723	5.9358
21	42	11.7540	5.9362
22	78	11.8916	5.9366
23	40	11.7728	5.9370
24	90	11.8036	5.9374
25	89	11.7625	5.9378
26	2	11.7553	5.9382
27	10	11.8032	5.9386
28	20	11.7897	5.9390
29	91	12.2509	5.9394
30	1	11.7482	5.9398

Tabla 4.3.0.1. Ejecuciones Australia

Himno	Parámetros	TiempoEjecucion(s)	UsoMemoria(MB)
Australia	Average	13.3370	5.9330
	SD	2.2320	0.0070
	Max	21.8156	5.9398
	Min	11.7482	5.9011
Canadá	Average	2.2672	1.6134
	SD	0.1914	0.0070
	Max	2.7470	1.6202
	Min	2.0245	1.5815
Gran Bretaña	Average	7.9106	3.8298
	SD	1.6691	0.0070
	Max	15.6925	3.8366
	Min	6.6204	3.7979
India	Average	2.5626	1.7630
	SD	0.1623	0.0068
	Max	2.9756	1.7697
	Min	2.3115	1.7323
Irlanda	Average	27.7280	9.8431
	SD	3.9408	0.0070
	Max	39.7620	9.8499
	Min	24.6128	9.8112
Jamaica	Average	11.6885	5.3762
	SD	1.3711	0.0068
	Max	15.1518	5.3830
	Min	10.1736	5.3455
Nueva Zelanda	Average	92.7189	21.7070
	SD	18.0516	0.0070
	Max	149.5322	21.7138
	Min	82.6005	21.6751
Singapur	Average	0.7071	0.6921
	SD	0.1679	0.0070
	Max	1.3809	0.6989
	Min	0.5553	0.6602
Sudáfrica	Average	0.7658	0.8320
	SD	0.0520	0.0070
	Max	0.9386	0.8388
	Min	0.7127	0.8001
USA	Average	280.4113	44.5373
	SD	59.3591	0.0070
	Max	576.2381	44.5441
	Min	255.1275	44.5054

Tabla 4.3.0.2. Resumen Ejecuciones Mac

Himno	Parámetros	TiempoEjecucion(s)	UsoMemoria(MB)
Australia	Average	8.2658	5.9330
	SD	0.1805	0.0070
	Max	8.5023	5.9398
	Min	7.9580	5.9011
Canadá	Average	1.0878	1.5841
	SD	0.0289	0.0070
	Max	1.1596	1.5909
	Min	1.0596	1.5522
Gran Bretaña	Average	3.9785	3.8298
	SD	0.0473	0.0070
	Max	4.1067	3.8366
	Min	3.9137	3.7979
India	Average	1.2708	1.7630
	SD	0.0165	0.0068
	Max	1.3227	1.7697
	Min	1.2483	1.7323
Irlanda	Average	17.6643	9.8431
	SD	0.3715	0.0070
	Max	18.3467	9.8499
	Min	17.0091	9.8112
Jamaica	Average	7.0676	5.3762
	SD	0.1952	0.0068
	Max	7.5738	5.3830
	Min	6.7943	5.3455
Nueva Zelanda	Average	61.2749	21.7070
	SD	2.0358	0.0070
	Max	64.1252	21.7138
	Min	57.9498	21.6751
Singapur	Average	0.3318	0.6921
	SD	0.0617	0.0070
	Max	0.6527	0.6989
	Min	0.3075	0.6602
Sudáfrica	Average	0.4134	0.8321
	SD	0.0107	0.0070
	Max	0.4581	0.8388
	Min	0.4010	0.8001
USA	Average	198.1801	44.5373
	SD	11.1756	0.0070
	Max	213.1193	44.5441
	Min	176.1961	44.5054

Tabla 4.3.0.3. Resumen Ejecuciones Windows

Himno	Parámetros	TiempoEjecucion(s)	UsoMemoria(MB)
Australia	Average	5.4928	5.9330
	SD	0.0870	0.0070
	Max	5.7460	5.9398
	Min	5.4067	5.9011
Canadá	Average	0.9267	1.6134
	SD	0.0317	0.0070
	Max	1.0079	1.6202
	Min	0.8720	1.5815
Gran Bretaña	Average	3.0410	3.8298
	SD	0.0336	0.0070
	Max	3.1122	3.8366
	Min	2.9755	3.7979
India	Average	1.0303	1.7630
	SD	0.0250	0.0068
	Max	1.0771	1.7697
	Min	0.9749	1.7323
Irlanda	Average	11.6492	9.8431
	SD	0.2980	0.0070
	Max	12.6112	9.8499
	Min	11.3622	9.8112
Jamaica	Average	5.0064	5.3762
	SD	0.0669	0.0068
	Max	5.2021	5.3830
	Min	4.8887	5.3455
Nueva Zelanda	Average	56.5294	29.3796
	SD	1.1452	0.0070
	Max	59.4291	29.3864
	Min	54.8559	29.3477
Singapur	Average	0.2881	0.6921
	SD	0.0098	0.0070
	Max	0.3312	0.6989
	Min	0.2803	0.6602
Sudáfrica	Average	0.3697	0.8321
	SD	0.0113	0.0070
	Max	0.4188	0.8388
	Min	0.3588	0.8001
USA	Average	103.6944	44.5373
	SD	3.7643	0.0070
	Max	110.1514	44.5441
	Min	98.1536	44.5054

Tabla 4.3.0.4. Resumen Ejecuciones Online

Capítulo 5

Cifrado por bloque y múltiple

5.1. Algoritmo Criptográfico

```
\begin{verbatim}
clc;
clear all;
close all;
% Iniciar el cronómetro
tic;
% Definir m como el valor primo ASCII más grande que deseas cifrar
m = 257;

% Definir la cantidad de matrices y el tamaño de bloque
num_matrices = 7;
block_size = 3;

% Generar las matrices de manera aleatoria
k_matrices = cell(1, num_matrices);
for i = 1:num_matrices
    while true
        k = randi([1, m-1], block_size);
        % if gcd(mod(round(det(k)), m), m) == 1
        %     % Verificar que la matriz generada no haya sido utilizada antes
        %     if ~ismember(k, cell2mat(k_matrices(1:i-1)))
        %         k_matrices{i} = k;
```

```
        %         break;
        %     end
        % end
    end
end

% Solicitar el texto plano
p = input('Ingresa el plaintext:', 's');
%p = 'Universidad Autonoma de Queretaro';
%disp(p);


% Leer el archivo de texto
%Archivo = 'donQuijote.txt';
%p = fileread(Archivo);


% Asegurar que el texto plano sea una cadena múltiplo de 3 caracteres
lonp = length(p);
a = mod(lonp, block_size);
if a ~= 0
    rm = block_size - a;
    for i = 1:rm
        p(lonp+i) = 'h';
    end
end

% Crear archivos para almacenar los resultados
salida_cifrado = 'ciphertext.txt';
salida_descifrado = 'desciphertext.txt';
cifrado_c_u = 'ronda_cifrado.txt';
descifrado_c_u = 'ronda_descifrado.txt';

% Abrir los archivos en modo de escritura
end_ciphertext = fopen(salida_cifrado, 'w');
end_desciphertext = fopen(salida_descifrado, 'w');
each_ciphertext = fopen(cifrado_c_u, 'w');
each_desciphertext = fopen(descifrado_c_u, 'w');
```

```
% Cifrado
lonp = length(p);
ciphertext = '';
for r = 1:num_matrices
    c = '';
    for i = 1:block_size:lonp
        s = double(p(i:i+block_size-1))';
        k = k_matrices{r}; % Utilizar una matriz de cifrado diferente en
        %cada ronda
        c_block = mod(k * s, m);
        c = [c char(c_block)];
    end
    ciphertext = [ciphertext c];
    fprintf(each_ciphertext, 'Ciphertext (Ronda %d): %s\n', r, c); %Guardar
    %el resultado de la ronda de cifrado en el archivo
end

fprintf(end_ciphertext, 'Ciphertext: %s\n', ciphertext);  %Guardar el
%cifrado final en el archivo

% Descifrado
lonc = length(ciphertext);
desciphertext = '';
for r = num_matrices:-1:1
    p = '';
    for i = 1:block_size:lonc
        r_block = double(ciphertext(i:i+block_size-1))';
        k = k_matrices{r}; % Utilizar la misma matriz de cifrado en cada ronda
        %de descifrado, pero en sentido inverso
        [InvMod, I] = gauss_jacques(k, m);
        p_block = mod(InvMod * r_block, m);
        p = [p char(p_block)];
    end
   desciphertext = p;
   fprintf(each_desciphertext,
   'Desciphertext (Ronda %d): %s\n', num_matrices - r + 1, p);  % Guardar
   %el resultado de la ronda de descifrado en el archivo
```

```
end
fprintf(end_desciphertext, 'Desciphertext: %s\n', desciphertext);  %Guardar
%el descifrado final en el archivo

% Detener el cronómetro y calcular el tiempo de ejecución
tiempo_ejecucion = toc;

% Cerrar los archivos
fclose(end_ciphertext);
fclose(end_desciphertext);
fclose(each_ciphertext);
fclose(each_desciphertext);

% Mostrar el resultado del descifrado
fprintf('Texto plano descifrado: %s\n', desciphertext(1:lonp));

% Mostrar el tiempo de ejecución
fprintf('Tiempo de ejecución: %.2f segundos\n', tiempo_ejecucion);
```

5.2. Procedimiento experimental

Se realizó la experimentación con los mismos diez himnos nacionales mencionados en la sección 4.2.

Para motivos de la experimentación y practicidad la longitud de los himnos se modificó de manera que fueran múltiplo de trescientos esto con el fin de que el algoritmo funcione por bloques. En este caso la experimentación fue con trescientos bloques, treinta bloques y tres bloques. La longitud de cada himbo se muestran en la tabla 5.2.0.1.

Además de los bloques se realizó el cifrado y descifrado con siete rondas para cada bloque, esto con el fin de prevenir un ataque por medio de la fuerza bruta. Por lo que se generó una llave para cifrar/descifrar cada bloque y para cada ronda. Por ejemplo; si el plaintext se dividió en tres bloques, se generaban veintiún llaves. Una para cada bloque (tres llaves) y una para cada ronda (siete), haciendo un total de veintiún llaves ($7x3 = 21$).

Himno	Longitud
Australia	900
Canadá	600
Gran Bretaña	600
India	600
Irlanda	900
Jamaica	600
Nueva Zelanda	1200
Singapur	300
Sudáfrica	300
USA	1800

Tabla 5.2.0.1. Longitud de cada Himno en bloques.

5.3. Resultados

En esta sección se realizaron del mismo modo pruebas en tres diferentes secciones: Tres, treinta y trescientos bloques. Para más detalles de los resultados y todas las tablas obtenidas revisar el apartado número

5.3.1. Tres bloques

5.3.2. Treinta bloques

5.3.3. Trescientos bloques

Himno	Parámetros	TiempoEjecucion(s)	UsoMemoria(MB)
Australia	Average	51.7260	1443.6046
	SD	20.4781	0.0088
	Max	90.8278	1443.6117
	Min	36.8167	1443.5615
Canadá	Average	12.0455	641.7468
	SD	0.6378	0.0089
	Max	14.6893	641.7539
	Min	11.3357	641.7037
	Average	12.0719	641.7482
Gran	SD	0.5583	0.0088
Bretaña	Max	13.2173	641.7553
	Min	11.4135	641.7051
India	Average	12.0266	641.7469
	SD	0.4814	0.0089
	Max	12.9854	641.7541
	Min	11.4279	641.7039
Irlanda	Average	37.6622	1443.6059
	SD	1.2757	0.0089
	Max	40.4618	1443.6131
	Min	35.7892	1443.5629
Jamaica	Average	12.0122	641.7489
	SD	0.4890	0.0089
	Max	12.9713	641.7560
	Min	11.4587	641.7058
	Average	117.3482	2566.2041
Nueva	SD	2.8110	0.0089
Zelanda	Max	123.3332	2566.2113
	Min	113.7427	2566.1611
Singapur	Average	1.9650	160.6302
	SD	0.1603	0.0088
	Max	2.5828	160.6373
	Min	1.7717	160.5872
Sudáfrica	Average	2.0192	160.6304
	SD	0.1072	0.0089
	Max	2.2497	160.6375
	Min	1.8010	160.5873
USA	Average	399.4707	5773.6177
	SD	22.1698	0.0089
	Max	503.1189	5773.6249
	Min	381.8468	5773.5747

Tabla 5.3.1.1. Ejecuciones por bloque (tres) Mac

Himno	Parámetros	TiempoEjecucion(s)	UsoMemoria(MB)
Australia	Average	23.1433	1443.6046
	SD	0.1935	0.0088
	Max	23.6496	1443.6117
	Min	22.8555	1443.5615
Canadá	Average	7.5362	641.7468
	SD	0.0647	0.0089
	Max	7.6959	641.7539
	Min	7.4622	641.7037
Gran Bretaña	Average	7.4452	641.7482
	SD	0.0785	0.0088
	Max	7.6601	641.7553
	Min	7.3580	641.7051
India	Average	7.4440	641.7469
	SD	0.0690	0.0089
	Max	7.6496	641.7541
	Min	7.3474	641.7039
Irlanda	Average	23.0346	1443.6059
	SD	0.1658	0.0089
	Max	23.5341	1443.6131
	Min	22.8294	1443.5629
Jamaica	Average	7.4370	641.7489
	SD	0.0896	0.0089
	Max	7.6798	641.7560
	Min	7.3379	641.7058
Nueva Zelanda	Average	52.2878	2566.2041
	SD	1.0151	0.0089
	Max	56.3990	2566.2113
	Min	51.0499	2566.1611
Singapur	Average	1.0940	160.6302
	SD	0.0226	0.0088
	Max	1.1929	160.6373
	Min	1.0766	160.5872
Sudáfrica	Average	1.0962	160.6304
	SD	0.0229	0.0089
	Max	1.1788	160.6375
	Min	1.0763	160.5873
USA	Average	207.0969	5773.6177
	SD	3.9941	0.0089
	Max	217.9005	5773.6249
	Min	193.5934	5773.5747

Tabla 5.3.1.2. Ejecuciones por bloque (tres) WIndows

Himno	Parámetros	TiempoEjecucion(s)	UsoMemoria(MB)
Australia	Average	18.0042	1443.6046
	SD	0.9697	0.0088
	Max	21.8654	1443.6117
	Min	17.3976	1443.5615
Canadá	Average	7.9976	641.7468
	SD	0.1035	0.0089
	Max	8.2977	641.7539
	Min	7.8486	641.7037
Gran Bretaña	Average	8.4456	641.7482
	SD	0.6794	0.0089
	Max	10.4663	641.7553
	Min	7.9468	641.7051
India	Average	7.9354	641.7469
	SD	0.0658	0.0089
	Max	8.1430	641.7541
	Min	7.8528	641.7039
Irlanda	Average	21.8858	1443.6059
	SD	1.2787	0.0089
	Max	24.3620	1443.6131
	Min	18.8086	1443.5629
Jamaica	Average	7.3608	641.7489
	SD	0.4533	0.0089
	Max	8.2585	641.7560
	Min	6.8932	641.7058
Nueva Zelanda	Average	39.6337	2566.2041
	SD	3.3606	0.0089
	Max	46.6266	2566.2113
	Min	36.9776	2566.1611
Singapur	Average	1.1950	160.6302
	SD	0.0289	0.0089
	Max	1.3039	160.6373
	Min	1.1720	160.5872
Sudáfrica	Average	1.2024	160.6304
	SD	0.0174	0.0089
	Max	1.2418	160.6375
	Min	1.1757	160.5873
USA	Average	117.6108	5773.6177
	SD	2.4147	0.0089
	Max	123.7766	5773.6249
	Min	114.2720	5773.5747

Tabla 5.3.1.3. Ejecuciones por bloque (tres) Online

Himno	Parámetros	TiempoEjecucion(s)	UsoMemoria(MB)
Australia	Average	0.5408	14.7064
	SD	0.0563	0.0089
	Max	0.7940	14.7136
	Min	0.5083	14.6634
Canadá	Average	0.2088	6.6804
	SD	0.1224	0.0088
	Max	0.7763	6.6875
	Min	0.1548	6.6373
Gran Bretaña	Average	0.2311	6.6817
	SD	0.0614	0.0089
	Max	0.4012	6.6889
	Min	0.1683	6.6387
India	Average	0.4312	6.6805
	SD	0.3514	0.0089
	Max	1.4407	6.6876
	Min	0.1724	6.6375
Irlanda	Average	0.5199	14.7078
	SD	0.0562	0.0089
	Max	0.6917	14.7149
	Min	0.4415	14.6647
Jamaica	Average	0.2154	6.6825
	SD	0.0679	0.0088
	Max	0.4195	6.6896
	Min	0.1721	6.6394
Nueva Zelanda	Average	1.1003	25.9419
	SD	0.1118	0.0088
	Max	1.5752	25.9490
	Min	1.0116	25.8989
Singapur	Average	0.0969	1.8632
	SD	0.2227	0.0088
	Max	1.2734	1.8703
	Min	0.0381	1.8201
Sudáfrica	Average	0.0655	1.8633
	SD	0.0217	0.0089
	Max	0.1600	1.8705
	Min	0.0434	1.8203
USA	Average	4.0908	58.0303
	SD	1.0739	0.0089
	Max	6.5882	58.0374
	Min	3.1649	57.9872

Tabla 5.3.2.1. Ejecuciones por bloque (treinta) Mac

Himno	Parámetros	TiempoEjecucion(s)	UsoMemoria(MB)
Australia	Average	0.3499	14.7064
	SD	0.0378	0.0089
	Max	0.4705	14.7136
	Min	0.3005	14.6634
Canadá	Average	0.1268	6.6804
	SD	0.0449	0.0088
	Max	0.3506	6.6875
	Min	0.1036	6.6373
Gran Bretaña	Average	0.1163	6.6817
	SD	0.0120	0.0089
	Max	0.1542	6.6889
	Min	0.1001	6.6387
India	Average	0.1218	6.6805
	SD	0.0181	0.0089
	Max	0.1719	6.6876
	Min	0.1005	6.6375
Irlanda	Average	0.3488	14.7078
	SD	0.0339	0.0089
	Max	0.4482	14.7149
	Min	0.3038	14.6647
Jamaica	Average	0.1162	6.6825
	SD	0.0127	0.0088
	Max	0.1542	6.6896
	Min	0.1022	6.6394
Nueva Zelanda	Average	0.8040	25.9419
	SD	0.0692	0.0088
	Max	1.0133	25.9490
	Min	0.6857	25.8989
Singapur	Average	0.0309	1.8632
	SD	0.0109	0.0088
	Max	0.0727	1.8703
	Min	0.0217	1.8201
Sudáfrica	Average	0.0309	1.8633
	SD	0.0092	0.0089
	Max	0.0610	1.8705
	Min	0.0200	1.8203
USA	Average	2.5386	58.0303
	SD	0.1246	0.0089
	Max	2.8250	58.0374
	Min	2.3207	57.9872

Tabla 5.3.2.2. Ejecuciones por bloque (treinta) Windows

Himno	Parámetros	TiempoEjecucion(s)	UsoMemoria(MB)
Australia	Average	0.3269	14.7064
	SD	0.0103	0.0089
	Max	0.3563	14.7136
	Min	0.3192	14.6634
Canadá	Average	0.1557	6.6804
	SD	0.0308	0.0088
	Max	0.2372	6.6875
	Min	0.1233	6.6373
Gran Bretaña	Average	0.1174	6.6817
	SD	0.0097	0.0089
	Max	0.1526	6.6889
	Min	0.1114	6.6387
India	Average	0.1174	6.6805
	SD	0.0091	0.0089
	Max	0.1531	6.6876
	Min	0.1128	6.6375
Irlanda	Average	0.3279	14.7078
	SD	0.0108	0.0089
	Max	0.3716	14.7149
	Min	0.3185	14.6647
Jamaica	Average	0.1164	6.6825
	SD	0.0081	0.0088
	Max	0.1443	6.6896
	Min	0.1112	6.6394
Nueva Zelanda	Average	0.7075	25.9419
	SD	0.0116	0.0088
	Max	0.7492	25.9490
	Min	0.6989	25.8989
Singapur	Average	0.0294	1.8632
	SD	0.0039	0.0088
	Max	0.0493	1.8703
	Min	0.0274	1.8201
Sudáfrica	Average	0.0300	1.8633
	SD	0.0044	0.0089
	Max	0.0490	1.8705
	Min	0.0275	1.8203
USA	Average	2.2517	58.0303
	SD	0.0133	0.0088
	Max	2.3010	58.0374
	Min	2.2297	57.9872

Tabla 5.3.2.3. Ejecuciones por bloque (treinta) Online

Himno	Parámetros	TiempoEjecucion(s)	UsoMemoria(MB)
Australia	Average	0.0819	0.4172
	SD	0.0291	0.0089
	Max	0.1932	0.4243
	Min	0.0573	0.3742
Canadá	Average	0.0771	0.3295
	SD	0.0241	0.0089
	Max	0.1360	0.3367
	Min	0.0533	0.2865
Gran Bretaña	Average	0.0762	0.3309
	SD	0.0347	0.0089
	Max	0.1894	0.3381
	Min	0.0441	0.2879
India	Average	0.0645	0.3297
	SD	0.0230	0.0089
	Max	0.1288	0.3368
	Min	0.0470	0.2866
Irlanda	Average	0.0780	0.4186
	SD	0.0301	0.0089
	Max	0.1845	0.4257
	Min	0.0524	0.3755
Jamaica	Average	0.0733	0.3316
	SD	0.0232	0.0089
	Max	0.1178	0.3388
	Min	0.0444	0.2886
Nueva Zelanda	Average	0.1009	0.5390
	SD	0.0396	0.0088
	Max	0.2070	0.5461
	Min	0.0766	0.4959
Singapur	Average	0.0654	0.2754
	SD	0.0210	0.0089
	Max	0.1293	0.2826
	Min	0.0424	0.2324
Sudáfrica	Average	0.0604	0.2756
	SD	0.0253	0.0089
	Max	0.1490	0.2827
	Min	0.0389	0.2325
USA	Average	0.1395	0.8740
	SD	0.0418	0.0088
	Max	0.2806	0.8811
	Min	0.1132	0.8309

Tabla 5.3.3.1. Ejecuciones por bloque (trescientos) Mac

Himno	Parámetros	TiempoEjecucion(s)	UsoMemoria(MB)
Australia	Average	0.0384	0.4172
	SD	0.0095	0.0089
	Max	0.0588	0.4243
	Min	0.0271	0.3742
Canadá	Average	0.0356	0.3295
	SD	0.0083	0.0089
	Max	0.0670	0.3367
	Min	0.0254	0.2865
Gran Bretaña	Average	0.0348	0.3309
	SD	0.0104	0.0089
	Max	0.0673	0.3381
	Min	0.0232	0.2879
India	Average	0.0338	0.3297
	SD	0.0153	0.0089
	Max	0.1033	0.3368
	Min	0.0235	0.2866
Irlanda	Average	0.0374	0.4186
	SD	0.0108	0.0089
	Max	0.0711	0.4257
	Min	0.0280	0.3755
Jamaica	Average	0.0322	0.3316
	SD	0.0061	0.0089
	Max	0.0462	0.3388
	Min	0.0238	0.2886
Nueva Zelanda	Average	0.0457	0.5390
	SD	0.0111	0.0088
	Max	0.0790	0.5461
	Min	0.0335	0.4959
Singapur	Average	0.0285	0.2754
	SD	0.0065	0.0089
	Max	0.0550	0.2826
	Min	0.0210	0.2324
Sudáfrica	Average	0.0251	0.2756
	SD	0.0059	0.0089
	Max	0.0414	0.2827
	Min	0.0170	0.2325
USA	Average	0.0785	0.8740
	SD	0.0285	0.0088
	Max	0.2104	0.8811
	Min	0.0610	0.8309

Tabla 5.3.3.2. Ejecuciones por bloque (trescientos) Windows

Himno	Parámetros	TiempoEjecucion(s)	UsoMemoria(MB)
Australia	Average	0.0355	0.4172
	SD	0.0125	0.0089
	Max	0.1008	0.4243
	Min	0.0312	0.3742
Canadá	Average	0.0308	0.3295
	SD	0.0147	0.0089
	Max	0.1064	0.3367
	Min	0.0261	0.2865
Gran Bretaña	Average	0.0297	0.3309
	SD	0.0092	0.0089
	Max	0.0661	0.3381
	Min	0.0252	0.2879
India	Average	0.0283	0.3297
	SD	0.0079	0.0089
	Max	0.0697	0.3368
	Min	0.0258	0.2866
Irlanda	Average	0.0337	0.4186
	SD	0.0058	0.0089
	Max	0.0615	0.4257
	Min	0.0303	0.3755
Jamaica	Average	0.0293	0.3316
	SD	0.0079	0.0089
	Max	0.0677	0.3388
	Min	0.0258	0.2886
Nueva Zelanda	Average	0.0426	0.5390
	SD	0.0106	0.0088
	Max	0.0782	0.5461
	Min	0.0378	0.4959
Singapur	Average	0.0243	0.2754
	SD	0.0034	0.0089
	Max	0.0416	0.2826
	Min	0.0223	0.2324
Sudáfrica	Average	0.0246	0.2756
	SD	0.0063	0.0089
	Max	0.0579	0.2827
	Min	0.0224	0.2325
USA	Average	0.0705	0.8740
	SD	0.0095	0.0088
	Max	0.1158	0.8811
	Min	0.0637	0.8309

Tabla 5.3.3.3. Ejecuciones por bloque (trescientos) Online

Capítulo 6

The HAJ

6.1. The HAJ: Pseudocode

Input : m: a prime number greater than the maximum ASCII value
Input : num_matrices: the number of matrices
Input : block_size: the block size
Output: ciphertext: the encrypted message
Output: decrypted_plaintext: the decrypted message
Step 1: Define m as a prime number greater than the maximum ASCII value to be encrypted
Step 2: Define the number of matrices and the block size
Step 3: Generate the matrices randomly, ensuring their determinant is non-zero modulo m
Step 4: Prompt the user for the plaintext
Step 5: Ensure the plaintext has a length that is a multiple of the block size by adding padding if necessary
Step 6: Initialize the ciphertext string
Step 7: For each block of size block size in the plaintext **for** $i \leftarrow 1$ **to** *length(plaintext) step block_size* **do**
    Step 8: Get the plaintext block and the corresponding key matrix
    Step 9: Perform the multiplication of the plaintext block by the key matrix modulo m
    Step 10: Add the resulting ciphertext block to the ciphertext string
end
Step 11: Display the resulting ciphertext

Step 12: Initialize the decrypted plaintext string
Step 13: For each block of size block size in the ciphertext **for** $i \leftarrow 1$ **to** *length(ciphertext) step block_ size* **do**
    Step 14: Get the ciphertext block and the corresponding key matrix
    Step 15: Calculate the modular inverse matrix using the Gauss-Jacques method $InvMod, I \leftarrow$ `gauss_jacques(`K, m`)`
    Step 16: Perform the multiplication of the ciphertext block by the inverse matrix modulo m
    Step 17: Add the resulting plaintext block to the decrypted plaintext string
end
Step 18: Display the resulting decrypted plaintext

Function `gauss_jacques(`K, m`):`
    return *InvMod, I*

Function `extended_euclidean(`a, b`):`
    return x, y, d

6.2. Resultados

Para ver a detalle los resultados obtenidos en esta sección, ver el Apéndice C (7.7.3).

Capítulo 7

Conclusiones y trabajos futuros

7.1. Conclusiones y Trabajo Futuro

En esta investigación se desarrolló el algoritmo criptográfico simétrico HAJ (Hill-Alarcón-Jacques), una propuesta que integra el cifrado de Hill con el método Gauss-Jacques para el cálculo eficiente de matrices inversas modulares. La combinación de ambas técnicas permitió construir un esquema de cifrado basado en operaciones matriciales, procesamiento por bloques y múltiples rondas de transformación.

Durante el desarrollo del trabajo se analizaron los fundamentos matemáticos del cifrado de Hill y del método Gauss-Jacques, así como su integración dentro de una misma arquitectura criptográfica. Asimismo, se realizaron pruebas experimentales que permitieron evaluar el comportamiento del algoritmo bajo diferentes configuraciones, observándose un uso estable de memoria y tiempos de ejecución competitivos, especialmente al emplear estrategias de segmentación por bloques.

Los resultados obtenidos muestran que la combinación del método de Gauss-Jacques y el algoritmo extendido de Euclides constituye una alternativa viable para el cálculo de inversos modulares en sistemas criptográficos basados en matrices. Esto permitió implementar el algoritmo HAJ de manera funcional y analizar su comportamiento computacional en distintos escenarios de prueba.

Desde el punto de vista criptográfico, el algoritmo incorpora múltiples rondas de cifrado, generación dinámica de subclaves y procesamiento independiente por bloques. Estas características incrementan la complejidad

estructural del sistema y amplían el espacio efectivo de claves utilizado durante el proceso de cifrado. Aunque en esta investigación no se realizó una evaluación formal basada en teoría de la información, la arquitectura propuesta presenta elementos que sugieren una posible aproximación a niveles superiores de dispersión y aleatoriedad de la información respecto a esquemas clásicos basados únicamente en el cifrado de Hill.

Como trabajo futuro, se propone complementar esta investigación mediante una evaluación criptográfica formal basada en métricas de entropía de Shannon, información mutua entre texto plano y texto cifrado, efecto avalancha y pruebas estadísticas de aleatoriedad. Asimismo, resulta de interés analizar la resistencia del algoritmo frente a diferentes técnicas de criptoanálisis y comparar su desempeño con otros algoritmos criptográficos simétricos ampliamente utilizados.

Finalmente, este trabajo aporta una nueva propuesta dentro del área de la criptografía basada en matrices y establece una base para futuras investigaciones relacionadas con mecanismos de cifrado, autenticación e intercambio seguro de información utilizando estructuras algebraicas.

Referencias

Acharya, B., Panigrahy, S. K., Patra, S. K., y Panda, G. (2009). Image encryption using advanced hill cipher algorithm. *International Journal of Recent Trends in Engineering*, *1*(1), 663–667.

Acharya, B., Rath, G. S., Patra, S. K., y Panigrahy, S. K. (2007). Novel methods of generating self-invertible matrix for hill cipher algorithm. *International Journal of Security*, *1*, 14–21.

Agrawal, M., y Gupta, D. (2013). Cryptanalysis of hill cipher with known ciphertext attack. *International Journal of Computer Applications*, *72*(6), 39–44.

Alarcón-Narváez, D., y Jacques García, F. A. (2021). Towards a symmetric crypto algorithm: The haj. En *Itng 2021 18th international conference on information technology-new generations* (pp. 121–126).

Alarcón-Narváez, D., Lizama-Pérez, L. A., y Jacques-García, F. A. (2026). Autopotency and conjugacy of non-diagonalizable matrices for challenge–response authentication. *Cryptography*, *10*(1), 7.

Asad, M. M., Marouf, I., Al-Haija, Q. A., y AlShuaibi, A. (2019). Performance analysis of 128-bit modular inverse based extended euclidean using altera fpga kit. *Procedia Computer Science*, *160*, 543–548.

Bajard, J. C., Meloni, N., y Plantard, T. (2005). Study of modular inversion in rns. En *Advanced signal processing algorithms, architectures, and implementations xv* (Vol. 5910, p. 59100T).

Birje, M. N., Kumbi, A. A., y Sutagundar, A. V. (2017). Internet of things: a survey of architecture, requirements and applications. *International Journal of Hyperconnectivity and the Internet of Things (IJHIoT)*, *1*(2), 45–71.

Bos, J. W. (2014). Constant time modular inversion. *Journal of Cryptographic Engineering*, *4*(4), 275–281.

Cantón, D. (2023). *Gauss-jacques method.* `https://github.com/dCantonE/`

`gauss-jacques`. (GitHub)

Daemen, J., y Rijmen, V. (2020). *The design of rijndael: The advanced encryption standard (aes)*. Springer Nature.

de Benito Crosetti, B., y Ibáñez, J. M. S. (2016). La investigación basada en diseño en tecnología educativa. *Revista Interuniversitaria de Investigación en Tecnología Educativa*.

DOMO. (2020, January). *Data never sleeps 8.0*. Descargado Jan 28, 2020, de `https://www.domo.com/learn/data-never-sleeps-8` (Last checked on Nov 15, 2020)

El Fishawy, N. F., y Zaid, O. M. A. (2007). Quality of encryption measurement of bitmap images with rc6, mrc6, and rijndael block cipher algorithms. *IJ Network Security*, *5*(3), 241–251.

Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., y Whiting, D. (2000). Improved cryptanalysis of rijndael. En *International workshop on fast software encryption* (pp. 213–230).

García, F. A. J. (2018). El método gauss-jacques propuesto para la obtención de matrices inversas modulares de tamaño variable sin límite teórico. *DIGITAL CIENCIA UAQRO*, *11*(1).

GROBET, L. A. B., y cols. (2004). Descartes y bacon: Algunos aspectos metodológicos. *UNAM*.

Hill, L. S. (1929). Cryptography in an algebraic alphabet. *The American Mathematical Monthly*, *36*(6), 306–312.

Jacques-García, F. A., Uribe-Mejía, D., Macías-Bobadilla, G., y Chaparro-Sánchez, R. (2022). On modular inverse matrices a computation approach. *South Florida Journal of Development*, *3*(3), 3100–3111.

Jangid, R. K., Mohmmad, N., Didel, A., y Taterh, S. (2014). Hybrid approach of image encryption using dna cryptography and tf hill cipher algorithm. En *2014 international conference on communication and signal processing* (pp. 934–938).

Johnsonbaugh, R. (2005). *Matemáticas discretas*. Pearson Educación.

Khalaf, A. A., Abd El-karim, M. S., y Hamed, H. F. (2015). Proposed triple hill cipher algorithm for increasing the security level of encrypted binary data and its implementation using fpga. En *2015 17th international conference on advanced communication technology (icact)* (pp. 454–459).

Lee, Y. K., Park, Y. S., Kim, Y. S., Lee, S. W., y Jun, S. I. (2010, marzo 30). *Rijndael block cipher apparatus and encryption/decryption method thereof*. Google Patents. (US Patent 7,688,974)

Mahendran, R., y Mani, K. (2017). Generation of key matrix for hill cipher encryption using classical cipher. En *2017 world congress on computing and communication technologies (wccct)* (pp. 51–54).

McDermott, R. (2002). Experimental methodology in political science. *Political Analysis*, 325–342.

Paar, C., y Pelzl, J. (2010). *Understanding cryptography: A textbook for students and practitioners*. Springer Science & Business Media.

Paragas, J. R., Sison, A. M., y Medina, R. (2019b). An improved hill cipher algorithm using cbc and hexadecimal s-box. En *2019 ieee eurasia conference on iot, communication and engineering (ecice)* (pp. 77–81).

Paragas, J. R., Sison, A. M., y Medina, R. P. (2019a). Hill cipher modification: A simplified approach. En *2019 ieee 11th international conference on communication software and networks (iccsn)* (pp. 821–825).

Phiamphu, D., y Saha, P. (2018). Redesigned the architecture of extended-euclidean algorithm for modular multiplicative inverse and jacobi symbol. En *2018 2nd international conference on trends in electronics and informatics (icoei)* (pp. 1345–1349).

Putrie, V. M., Sari, C. A., Rachmawanto, E. H., y cols. (2018). Super encryption using transposition-hill cipher for digital color image. En *2018 international seminar on research of information technology and intelligent systems (isriti)* (pp. 152–157).

Saeednia, S. (2000). How to make the hill cipher secure. *Cryptologia*, *24*(4), 353–360.

Schneier, B. (1996). *Applied cryptography: Protocols, algorithms, and source code in c* (2.ª ed.). John Wiley & Sons.

Seysen, M. (2005). Using an rsa accelerator for modular inversion. En *International workshop on cryptographic hardware and embedded systems* (pp. 226–236).

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, *27*(3), 379–423.

Shannon, C. E. (1949). Communication theory of secrecy systems. *The Bell system technical journal*, *28*(4), 656–715.

Singh, S., y Srivastava, S. (2014a). Cryptanalysis of hill cipher using frequency attack. *International Journal of Computer Applications*, *86*(2), 1–5.

Singh, S., y Srivastava, S. (2014b). Cryptanalysis of hill cipher using known plaintext attack. *International Journal of Scientific and Research Publications*, *4*(4), 1–4.

Srinivas, N. S., y Akramuddin, M. (2016). Fpga based hardware implementation of aes rijndael algorithm for encryption and decryption. En *2016 international conference on electrical, electronics, and optimization techniques (iceeot)* (pp. 1769–1776).

Stallings, W. (2017). *Cryptography and network security: Principles and practice*. Pearson Education.

Walpole, R. E., Myers, R. H., y Myers, S. L. (1999). *Probabilidad y estadística para ingenieros*. Pearson educación.

William, S. (2006). *Cryptography and network security*. Pearson Education India.

Apéndice A

7.2. Ejecuciones MacBook

En esta sección se colocan todas las tablas de las treinta ejecuciones realizadas en la computadora MacBook con los diez diferentes himnos nacionales.

7.3. Ejecuciones Windows

En esta sección se colocan todas las tablas de las treinta ejecuciones realizadas en la computadora Windows con los diez diferentes himnos nacionales.

7.4. Ejecuciones Online

En esta sección se colocan todas las tablas de las treinta ejecuciones realizadas en la plataforma online con los diez diferentes himnos nacionales.

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	14.3504	5.9011
2	33	14.2019	5.9248
3	88	14.3617	5.9291
4	64	21.8156	5.9295
5	55	15.8586	5.9299
6	51	16.8142	5.9303
7	74	15.2743	5.9307
8	56	13.9836	5.9311
9	25	14.1974	5.9315
10	57	14.5327	5.9319
11	21	13.3327	5.9323
12	16	13.7034	5.9326
13	7	15.9926	5.9330
14	83	12.6617	5.9334
15	52	11.8213	5.9338
16	18	11.7951	5.9342
17	39	11.7608	5.9346
18	72	11.7796	5.9350
19	95	11.7669	5.9354
20	93	11.7723	5.9358
21	42	11.7540	5.9362
22	78	11.8916	5.9366
23	40	11.7728	5.9370
24	90	11.8036	5.9374
25	89	11.7625	5.9378
26	2	11.7553	5.9382
27	10	11.8032	5.9386
28	20	11.7897	5.9390
29	91	12.2509	5.9394
30	1	11.7482	5.9398

Tabla 7.2.0.1. Ejecuciones Australia

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	2.3817	1.5815
2	33	2.4151	1.6053
3	88	2.4914	1.6095
4	64	2.3323	1.6099
5	55	2.2553	1.6103
6	51	2.2873	1.6107
7	74	2.3417	1.6111
8	56	2.6014	1.6115
9	25	2.7470	1.6119
10	57	2.1229	1.6123
11	21	2.1499	1.6127
12	16	2.1704	1.6131
13	7	2.2346	1.6135
14	83	2.4932	1.6139
15	52	2.1174	1.6143
16	18	2.1295	1.6147
17	39	2.0808	1.6150
18	72	2.1397	1.6154
19	95	2.1256	1.6158
20	93	2.1468	1.6162
21	42	2.1427	1.6166
22	78	2.5015	1.6170
23	40	2.3932	1.6174
24	90	2.5185	1.6178
25	89	2.4215	1.6182
26	2	2.0443	1.6186
27	10	2.0686	1.6190
28	20	2.1036	1.6194
29	91	2.0245	1.6198
30	1	2.0341	1.6202

Tabla 7.2.0.2. Ejecuciones Canadá

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	15.6925	3.7979
2	33	7.7717	3.8216
3	88	7.2507	3.8258
4	64	7.2996	3.8262
5	55	7.2922	3.8266
6	51	7.2084	3.8270
7	74	7.1861	3.8274
8	56	7.2461	3.8278
9	25	7.3317	3.8282
10	57	8.8844	3.8286
11	21	8.2661	3.8290
12	16	8.1684	3.8294
13	7	8.1973	3.8298
14	83	8.0942	3.8302
15	52	9.5346	3.8306
16	18	8.6023	3.8310
17	39	7.3658	3.8314
18	72	7.2470	3.8318
19	95	7.1887	3.8322
20	93	7.2102	3.8326
21	42	7.1823	3.8330
22	78	7.3176	3.8334
23	40	7.2564	3.8338
24	90	7.1960	3.8342
25	89	7.4100	3.8346
26	2	9.9644	3.8350
27	10	8.0172	3.8354
28	20	6.6204	3.8358
29	91	6.6855	3.8362
30	1	6.6307	3.8366

Tabla 7.2.0.3. Ejecuciones Gran Bretaña

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	2.5406	1.7323
2	33	2.4026	1.7547
3	88	2.3730	1.7590
4	64	2.4330	1.7594
5	55	2.5847	1.7598
6	51	2.5719	1.7602
7	74	2.6977	1.7606
8	56	2.4750	1.7610
9	25	2.3464	1.7614
10	57	2.3300	1.7618
11	21	2.3115	1.7622
12	16	2.5788	1.7626
13	7	2.5914	1.7630
14	83	2.5804	1.7633
15	52	2.6250	1.7637
16	18	2.6291	1.7641
17	39	2.6935	1.7645
18	72	2.6876	1.7649
19	95	2.9756	1.7653
20	93	2.6872	1.7657
21	42	2.7657	1.7661
22	78	2.5909	1.7665
23	40	2.5194	1.7669
24	90	2.4193	1.7673
25	89	2.4106	1.7677
26	2	2.4339	1.7681
27	10	2.4093	1.7685
28	20	2.6932	1.7689
29	91	2.9071	1.7693
30	1	2.6146	1.7697

Tabla 7.2.0.4. Ejecuciones India

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	39.7620	9.8112
2	33	32.7001	9.8349
3	88	30.0376	9.8392
4	64	27.8112	9.8396
5	55	25.1709	9.8400
6	51	24.6611	9.8404
7	74	24.7321	9.8408
8	56	24.6942	9.8412
9	25	24.7442	9.8416
10	57	24.6746	9.8420
11	21	24.8116	9.8424
12	16	25.0090	9.8428
13	7	27.3592	9.8431
14	83	29.9095	9.8435
15	52	34.8187	9.8439
16	18	34.3114	9.8443
17	39	30.2694	9.8447
18	72	26.6265	9.8451
19	95	26.4575	9.8455
20	93	26.5297	9.8459
21	42	25.7142	9.8463
22	78	29.0122	9.8467
23	40	24.8461	9.8471
24	90	24.6202	9.8475
25	89	24.6128	9.8479
26	2	24.6616	9.8483
27	10	24.7085	9.8487
28	20	24.6641	9.8491
29	91	32.6549	9.8495
30	1	31.2563	9.8499

Tabla 7.2.0.5. Ejecuciones Irlanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	14.4339	5.3455
2	33	14.3996	5.3680
3	88	12.7167	5.3723
4	64	13.3491	5.3727
5	55	12.0234	5.3731
6	51	12.1258	5.3734
7	74	12.0848	5.3738
8	56	12.5756	5.3742
9	25	11.0547	5.3746
10	57	10.1736	5.3750
11	21	10.4694	5.3754
12	16	10.6030	5.3758
13	7	10.5941	5.3762
14	83	10.7006	5.3766
15	52	10.9196	5.3770
16	18	10.9812	5.3774
17	39	10.9582	5.3778
18	72	10.7570	5.3782
19	95	10.7885	5.3786
20	93	15.1518	5.3790
21	42	11.4417	5.3794
22	78	13.6779	5.3798
23	40	12.6391	5.3802
24	90	10.5019	5.3806
25	89	10.5865	5.3810
26	2	10.4347	5.3814
27	10	12.0455	5.3818
28	20	11.0829	5.3822
29	91	10.9564	5.3826
30	1	10.4282	5.3830

Tabla 7.2.0.6. Ejecuciones Jamaica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	149.5322	21.6751
2	33	140.0672	21.6989
3	88	127.1416	21.7031
4	64	125.4030	21.7035
5	55	89.1015	21.7039
6	51	83.2787	21.7043
7	74	82.7655	21.7047
8	56	91.0856	21.7051
9	25	89.6444	21.7055
10	57	83.0942	21.7059
11	21	82.7334	21.7063
12	16	82.8837	21.7067
13	7	96.9464	21.7071
14	83	82.7683	21.7075
15	52	82.9839	21.7079
16	18	83.0696	21.7083
17	39	90.5928	21.7087
18	72	89.8964	21.7090
19	95	82.8060	21.7094
20	93	82.6864	21.7098
21	42	83.0216	21.7102
22	78	96.5093	21.7106
23	40	82.8504	21.7110
24	90	82.7112	21.7114
25	89	82.6670	21.7118
26	2	96.4049	21.7122
27	10	84.0469	21.7126
28	20	82.6896	21.7130
29	91	82.6005	21.7134
30	1	89.5861	21.7138

Tabla 7.2.0.7. Ejecuciones Nueva Zelanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	1.3809	0.6602
2	33	0.6462	0.6840
3	88	0.8663	0.6882
4	64	0.9309	0.6886
5	55	0.9673	0.6890
6	51	0.8766	0.6894
7	74	0.6196	0.6898
8	56	0.6588	0.6902
9	25	0.8789	0.6906
10	57	0.7119	0.6910
11	21	0.6496	0.6914
12	16	0.6552	0.6918
13	7	0.6406	0.6922
14	83	0.5914	0.6926
15	52	0.6806	0.6930
16	18	0.7035	0.6934
17	39	0.5999	0.6938
18	72	0.5969	0.6942
19	95	0.5803	0.6946
20	93	0.5757	0.6950
21	42	0.6274	0.6954
22	78	0.5926	0.6958
23	40	0.6493	0.6962
24	90	0.6106	0.6966
25	89	0.6205	0.6969
26	2	0.6423	0.6973
27	10	0.6933	0.6977
28	20	0.7119	0.6981
29	91	0.6989	0.6985
30	1	0.5553	0.6989

Tabla 7.2.0.8. Ejecuciones Singapur

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	0.8155	0.8001
2	33	0.8763	0.8239
3	88	0.7600	0.8281
4	64	0.7736	0.8285
5	55	0.8247	0.8289
6	51	0.7419	0.8293
7	74	0.7541	0.8297
8	56	0.7343	0.8301
9	25	0.7357	0.8305
10	57	0.9386	0.8309
11	21	0.7987	0.8313
12	16	0.7296	0.8317
13	7	0.7283	0.8321
14	83	0.7435	0.8325
15	52	0.7560	0.8329
16	18	0.7555	0.8333
17	39	0.8011	0.8337
18	72	0.7449	0.8341
19	95	0.7982	0.8345
20	93	0.8321	0.8349
21	42	0.7745	0.8353
22	78	0.7842	0.8357
23	40	0.7239	0.8361
24	90	0.7224	0.8365
25	89	0.7223	0.8369
26	2	0.7127	0.8372
27	10	0.7257	0.8376
28	20	0.7149	0.8380
29	91	0.7274	0.8384
30	1	0.7231	0.8388

Tabla 7.2.0.9. Ejecuciones Sudáfrica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	576.2381	44.5054
2	33	371.7113	44.5291
3	88	263.5326	44.5334
4	64	269.5301	44.5338
5	55	269.7129	44.5342
6	51	270.4762	44.5346
7	74	255.3731	44.5350
8	56	269.1320	44.5354
9	25	269.6243	44.5358
10	57	269.1759	44.5362
11	21	256.7433	44.5366
12	16	272.2554	44.5370
13	7	275.1785	44.5373
14	83	274.6065	44.5377
15	52	255.7148	44.5381
16	18	270.3130	44.5385
17	39	269.4095	44.5389
18	72	262.8320	44.5393
19	95	263.7705	44.5397
20	93	271.5450	44.5401
21	42	270.2623	44.5405
22	78	255.1275	44.5409
23	40	269.8919	44.5413
24	90	270.3898	44.5417
25	89	268.9571	44.5421
26	2	256.5565	44.5425
27	10	269.9001	44.5429
28	20	269.6620	44.5433
29	91	261.2466	44.5437
30	1	263.4715	44.5441

Tabla 7.2.0.10. Ejecuciones USA

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	8.0297	5.9011
2	33	7.9957	5.9248
3	88	7.9580	5.9291
4	64	8.0145	5.9295
5	55	7.9899	5.9299
6	51	7.9621	5.9303
7	74	8.0340	5.9307
8	56	7.9658	5.9311
9	25	8.0906	5.9315
10	57	8.3862	5.9319
11	21	8.3806	5.9323
12	16	8.3426	5.9326
13	7	8.3229	5.9330
14	83	8.2908	5.9334
15	52	8.3126	5.9338
16	18	8.2932	5.9342
17	39	8.3665	5.9346
18	72	8.3510	5.9350
19	95	8.3658	5.9354
20	93	8.3723	5.9358
21	42	8.3972	5.9362
22	78	8.4089	5.9366
23	40	8.3550	5.9370
24	90	8.3711	5.9374
25	89	8.4111	5.9378
26	2	8.4265	5.9382
27	10	8.4198	5.9386
28	20	8.4288	5.9390
29	91	8.4273	5.9394
30	1	8.5023	5.9398

Tabla 7.3.0.1. Ejecuciones Australia

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	1.1491	1.5522
2	33	1.1005	1.5759
3	88	1.1057	1.5802
4	64	1.1141	1.5806
5	55	1.1215	1.5810
6	51	1.1008	1.5814
7	74	1.1580	1.5818
8	56	1.1596	1.5822
9	25	1.1153	1.5825
10	57	1.0879	1.5829
11	21	1.0978	1.5833
12	16	1.0678	1.5837
13	7	1.0744	1.5841
14	83	1.0596	1.5845
15	52	1.0643	1.5849
16	18	1.0709	1.5853
17	39	1.0734	1.5857
18	72	1.0672	1.5861
19	95	1.0748	1.5865
20	93	1.0746	1.5869
21	42	1.0775	1.5873
22	78	1.0772	1.5877
23	40	1.0627	1.5881
24	90	1.0766	1.5885
25	89	1.0625	1.5889
26	2	1.0603	1.5893
27	10	1.0661	1.5897
28	20	1.0641	1.5901
29	91	1.0673	1.5905
30	1	1.0835	1.5909

Tabla 7.3.0.2. Ejecuciones Canadá

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	4.0311	3.7979
2	33	3.9638	3.8216
3	88	3.9822	3.8258
4	64	3.9351	3.8262
5	55	3.9478	3.8266
6	51	3.9137	3.8270
7	74	3.9787	3.8274
8	56	3.9357	3.8278
9	25	3.9769	3.8282
10	57	3.9860	3.8286
11	21	3.9517	3.8290
12	16	3.9520	3.8294
13	7	3.9531	3.8298
14	83	3.9227	3.8302
15	52	3.9307	3.8306
16	18	4.0548	3.8310
17	39	3.9412	3.8314
18	72	3.9703	3.8318
19	95	3.9997	3.8322
20	93	3.9909	3.8326
21	42	3.9656	3.8330
22	78	3.9473	3.8334
23	40	3.9575	3.8338
24	90	3.9821	3.8342
25	89	4.1067	3.8346
26	2	3.9457	3.8350
27	10	3.9677	3.8354
28	20	4.0318	3.8358
29	91	4.0683	3.8362
30	1	4.0651	3.8366

Tabla 7.3.0.3. Ejecuciones Gran Bretaña

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	1.2925	1.7323
2	33	1.2731	1.7547
3	88	1.2774	1.7590
4	64	1.2738	1.7594
5	55	1.2714	1.7598
6	51	1.3227	1.7602
7	74	1.2959	1.7606
8	56	1.2879	1.7610
9	25	1.2993	1.7614
10	57	1.2742	1.7618
11	21	1.2611	1.7622
12	16	1.2699	1.7626
13	7	1.2592	1.7630
14	83	1.2600	1.7633
15	52	1.2853	1.7637
16	18	1.2655	1.7641
17	39	1.2615	1.7645
18	72	1.2588	1.7649
19	95	1.2698	1.7653
20	93	1.2499	1.7657
21	42	1.2516	1.7661
22	78	1.2595	1.7665
23	40	1.2789	1.7669
24	90	1.2637	1.7673
25	89	1.2719	1.7677
26	2	1.2483	1.7681
27	10	1.2641	1.7685
28	20	1.2546	1.7689
29	91	1.2669	1.7693
30	1	1.2551	1.7697

Tabla 7.3.0.4. Ejecuciones India

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	17.1499	9.8112
2	33	18.2555	9.8349
3	88	17.4808	9.8392
4	64	17.1464	9.8396
5	55	17.0091	9.8400
6	51	17.2028	9.8404
7	74	17.2346	9.8408
8	56	17.3645	9.8412
9	25	17.3564	9.8416
10	57	17.2426	9.8420
11	21	17.4085	9.8424
12	16	17.4947	9.8428
13	7	17.9208	9.8431
14	83	17.5496	9.8435
15	52	17.4567	9.8439
16	18	17.5803	9.8443
17	39	17.6673	9.8447
18	72	17.6046	9.8451
19	95	17.6194	9.8455
20	93	17.6885	9.8459
21	42	18.0908	9.8463
22	78	17.8246	9.8467
23	40	18.0977	9.8471
24	90	17.9078	9.8475
25	89	18.3467	9.8479
26	2	18.0092	9.8483
27	10	17.9466	9.8487
28	20	18.0771	9.8491
29	91	18.1221	9.8495
30	1	18.0739	9.8499

Tabla 7.3.0.5. Ejecuciones Irlanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	6.9464	5.3455
2	33	6.7943	5.3680
3	88	6.8213	5.3723
4	64	6.8719	5.3727
5	55	6.9158	5.3731
6	51	6.9003	5.3734
7	74	6.8617	5.3738
8	56	6.8784	5.3742
9	25	6.8249	5.3746
10	57	6.9602	5.3750
11	21	6.8429	5.3754
12	16	6.8344	5.3758
13	7	7.0659	5.3762
14	83	7.1098	5.3766
15	52	7.1312	5.3770
16	18	7.0646	5.3774
17	39	7.0875	5.3778
18	72	7.0635	5.3782
19	95	7.2556	5.3786
20	93	7.4415	5.3790
21	42	7.2147	5.3794
22	78	7.1916	5.3798
23	40	7.2149	5.3802
24	90	7.1473	5.3806
25	89	7.2858	5.3810
26	2	7.1809	5.3814
27	10	7.1408	5.3818
28	20	7.1446	5.3822
29	91	7.5738	5.3826
30	1	7.2624	5.3830

Tabla 7.3.0.6. Ejecuciones Jamaica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	58.3150	21.6751
2	33	59.1799	21.6989
3	88	58.1338	21.7031
4	64	57.9498	21.7035
5	55	58.3765	21.7039
6	51	58.4438	21.7043
7	74	58.6567	21.7047
8	56	59.0951	21.7051
9	25	59.7743	21.7055
10	57	60.2913	21.7059
11	21	60.0846	21.7063
12	16	60.3207	21.7067
13	7	60.8430	21.7071
14	83	61.3216	21.7075
15	52	62.2072	21.7079
16	18	61.8969	21.7083
17	39	62.2787	21.7087
18	72	62.5365	21.7090
19	95	62.9838	21.7094
20	93	63.4478	21.7098
21	42	63.7324	21.7102
22	78	64.1252	21.7106
23	40	63.3025	21.7110
24	90	62.2127	21.7114
25	89	62.5508	21.7118
26	2	62.9208	21.7122
27	10	63.4386	21.7126
28	20	63.4995	21.7130
29	91	62.6152	21.7134
30	1	63.7128	21.7138

Tabla 7.3.0.7. Ejecuciones Nueva Zelanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	0.6527	0.6602
2	33	0.3281	0.6840
3	88	0.3224	0.6882
4	64	0.3249	0.6886
5	55	0.3388	0.6890
6	51	0.3241	0.6894
7	74	0.3169	0.6898
8	56	0.3172	0.6902
9	25	0.3175	0.6906
10	57	0.3123	0.6910
11	21	0.3168	0.6914
12	16	0.3133	0.6918
13	7	0.3123	0.6922
14	83	0.3120	0.6926
15	52	0.3159	0.6930
16	18	0.3255	0.6934
17	39	0.3146	0.6938
18	72	0.3143	0.6942
19	95	0.3315	0.6946
20	93	0.3389	0.6950
21	42	0.3174	0.6954
22	78	0.3075	0.6958
23	40	0.3119	0.6962
24	90	0.3153	0.6966
25	89	0.3122	0.6969
26	2	0.3145	0.6973
27	10	0.3123	0.6977
28	20	0.3209	0.6981
29	91	0.3663	0.6985
30	1	0.3257	0.6989

Tabla 7.3.0.8. Ejecuciones Singapur

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	0.4581	0.8001
2	33	0.4135	0.8239
3	88	0.4152	0.8281
4	64	0.4152	0.8285
5	55	0.4372	0.8289
6	51	0.4185	0.8293
7	74	0.4185	0.8297
8	56	0.4176	0.8301
9	25	0.4171	0.8305
10	57	0.4096	0.8309
11	21	0.4037	0.8313
12	16	0.4075	0.8317
13	7	0.4082	0.8321
14	83	0.4070	0.8325
15	52	0.4082	0.8329
16	18	0.4114	0.8333
17	39	0.4117	0.8337
18	72	0.4107	0.8341
19	95	0.4086	0.8345
20	93	0.4036	0.8349
21	42	0.4102	0.8353
22	78	0.4191	0.8357
23	40	0.4119	0.8361
24	90	0.4010	0.8365
25	89	0.4066	0.8369
26	2	0.4078	0.8372
27	10	0.4086	0.8376
28	20	0.4091	0.8380
29	91	0.4123	0.8384
30	1	0.4132	0.8388

Tabla 7.3.0.9. Ejecuciones Sudáfrica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	176.1961	44.5054
2	33	176.3226	44.5291
3	88	179.7449	44.5334
4	64	185.9810	44.5338
5	55	185.6480	44.5342
6	51	213.1193	44.5346
7	74	188.0051	44.5350
8	56	184.8002	44.5354
9	25	180.8368	44.5358
10	57	192.8784	44.5362
11	21	200.9485	44.5366
12	16	203.2780	44.5370
13	7	201.3078	44.5373
14	83	207.4352	44.5377
15	52	211.0270	44.5381
16	18	203.0654	44.5385
17	39	202.7078	44.5389
18	72	207.6439	44.5393
19	95	209.4856	44.5397
20	93	199.9644	44.5401
21	42	208.4658	44.5405
22	78	209.5396	44.5409
23	40	190.1480	44.5413
24	90	211.7355	44.5417
25	89	204.8193	44.5421
26	2	202.3045	44.5425
27	10	202.9196	44.5429
28	20	201.9916	44.5433
29	91	198.5669	44.5437
30	1	204.5149	44.5441

Tabla 7.3.0.10. Ejecuciones USA

No.	semilla	TiempoEjecucion	UsoMemoria
1	36	5.4156	5.9011
2	33	5.4597	5.9248
3	88	5.4159	5.9291
4	64	5.4074	5.9295
5	55	5.4663	5.9299
6	51	5.4706	5.9303
7	74	5.4262	5.9307
8	56	5.4376	5.9311
9	25	5.4248	5.9315
10	57	5.4245	5.9319
11	21	5.4369	5.9323
12	16	5.4067	5.9326
13	7	5.4226	5.9330
14	83	5.4543	5.9334
15	52	5.4737	5.9338
16	18	5.5236	5.9342
17	39	5.4271	5.9346
18	72	5.4548	5.9350
19	95	5.4670	5.9354
20	93	5.4975	5.9358
21	42	5.5784	5.9362
22	78	5.7460	5.9366
23	40	5.4811	5.9370
24	90	5.5383	5.9374
25	89	5.6600	5.9378
26	2	5.5732	5.9382
27	10	5.6613	5.9386
28	20	5.6324	5.9390
29	91	5.5137	5.9394
30	1	5.4861	5.9398

Tabla 7.4.0.1. Ejecuciones Australia

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	0.9610	1.5815
2	33	0.9312	1.6053
3	88	0.9047	1.6095
4	64	0.9343	1.6099
5	55	1.0079	1.6103
6	51	0.9889	1.6107
7	74	0.9923	1.6111
8	56	0.9439	1.6115
9	25	0.9214	1.6119
10	57	0.9110	1.6123
11	21	0.9360	1.6127
12	16	0.9256	1.6131
13	7	0.9205	1.6135
14	83	0.9295	1.6139
15	52	0.9119	1.6143
16	18	0.9054	1.6147
17	39	0.9339	1.6150
18	72	0.8894	1.6154
19	95	0.8801	1.6158
20	93	0.9037	1.6162
21	42	0.8974	1.6166
22	78	0.9365	1.6170
23	40	0.9216	1.6174
24	90	0.8720	1.6178
25	89	0.9146	1.6182
26	2	0.8981	1.6186
27	10	0.9344	1.6190
28	20	0.9662	1.6194
29	91	0.9043	1.6198
30	1	0.9237	1.6202

Tabla 7.4.0.2. Ejecuciones Canadá

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	3.0172	3.7979
2	33	3.1060	3.8216
3	88	3.0395	3.8258
4	64	3.0266	3.8262
5	55	3.1065	3.8266
6	51	3.0383	3.8270
7	74	3.1122	3.8274
8	56	3.0643	3.8278
9	25	3.0476	3.8282
10	57	3.0544	3.8286
11	21	3.0031	3.8290
12	16	3.0480	3.8294
13	7	3.0571	3.8298
14	83	2.9755	3.8302
15	52	3.0397	3.8306
16	18	3.0080	3.8310
17	39	3.0450	3.8314
18	72	3.0755	3.8318
19	95	3.0138	3.8322
20	93	3.0684	3.8326
21	42	3.0341	3.8330
22	78	2.9959	3.8334
23	40	3.0175	3.8338
24	90	3.0233	3.8342
25	89	3.0145	3.8346
26	2	3.0353	3.8350
27	10	2.9992	3.8354
28	20	3.0269	3.8358
29	91	3.0482	3.8362
30	1	3.0879	3.8366

Tabla 7.4.0.3. Ejecuciones Gran Bretaña

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	1.0423	1.7323
2	33	0.9749	1.7547
3	88	0.9959	1.7590
4	64	1.0108	1.7594
5	55	1.0207	1.7598
6	51	1.0027	1.7602
7	74	0.9820	1.7606
8	56	0.9860	1.7610
9	25	1.0152	1.7614
10	57	1.0771	1.7618
11	21	1.0331	1.7622
12	16	1.0553	1.7626
13	7	1.0368	1.7630
14	83	1.0410	1.7633
15	52	1.0599	1.7637
16	18	1.0300	1.7641
17	39	1.0255	1.7645
18	72	1.0224	1.7649
19	95	1.0134	1.7653
20	93	1.0553	1.7657
21	42	1.0305	1.7661
22	78	1.0311	1.7665
23	40	1.0436	1.7669
24	90	1.0495	1.7673
25	89	1.0739	1.7677
26	2	1.0430	1.7681
27	10	1.0342	1.7685
28	20	1.0405	1.7689
29	91	1.0330	1.7693
30	1	1.0508	1.7697

Tabla 7.4.0.4. Ejecuciones India

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	11.4185	9.8112
2	33	11.4363	9.8349
3	88	11.6517	9.8392
4	64	11.9211	9.8396
5	55	11.7286	9.8400
6	51	11.4697	9.8404
7	74	11.5654	9.8408
8	56	11.3809	9.8412
9	25	11.4486	9.8416
10	57	11.4646	9.8420
11	21	11.3622	9.8424
12	16	11.3758	9.8428
13	7	11.3951	9.8431
14	83	11.6450	9.8435
15	52	11.6245	9.8439
16	18	11.4643	9.8443
17	39	11.4494	9.8447
18	72	11.4258	9.8451
19	95	11.4128	9.8455
20	93	11.7393	9.8459
21	42	12.6112	9.8463
22	78	12.3009	9.8467
23	40	11.9741	9.8471
24	90	11.4897	9.8475
25	89	11.9518	9.8479
26	2	12.0305	9.8483
27	10	11.7315	9.8487
28	20	11.7190	9.8491
29	91	11.8303	9.8495
30	1	11.4560	9.8499

Tabla 7.4.0.5. Ejecuciones Irlanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	5.0070	5.3455
2	33	4.9370	5.3680
3	88	4.9976	5.3723
4	64	4.9999	5.3727
5	55	4.9783	5.3731
6	51	4.9764	5.3734
7	74	5.0489	5.3738
8	56	5.0183	5.3742
9	25	4.9936	5.3746
10	57	4.9453	5.3750
11	21	4.9937	5.3754
12	16	5.0015	5.3758
13	7	5.0577	5.3762
14	83	5.1037	5.3766
15	52	5.2021	5.3770
16	18	5.1018	5.3774
17	39	5.0306	5.3778
18	72	5.0600	5.3782
19	95	5.1197	5.3786
20	93	5.0440	5.3790
21	42	4.9764	5.3794
22	78	5.0195	5.3798
23	40	4.9988	5.3802
24	90	4.8887	5.3806
25	89	4.9045	5.3810
26	2	4.9500	5.3814
27	10	4.9161	5.3818
28	20	4.9749	5.3822
29	91	4.9755	5.3826
30	1	4.9714	5.3830

Tabla 7.4.0.6. Ejecuciones Jamaica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	56.5359	29.3477
2	33	57.0518	29.3715
3	88	56.3435	29.3757
4	64	56.1977	29.3761
5	55	57.7483	29.3765
6	51	59.4291	29.3769
7	74	57.8269	29.3773
8	56	57.4591	29.3777
9	25	56.9440	29.3781
10	57	57.1308	29.3785
11	21	56.9113	29.3789
12	16	55.8240	29.3793
13	7	57.1367	29.3797
14	83	55.1673	29.3801
15	52	55.2158	29.3805
16	18	57.6819	29.3809
17	39	55.9168	29.3813
18	72	57.1656	29.3816
19	95	56.7037	29.3820
20	93	54.9234	29.3824
21	42	55.1713	29.3828
22	78	55.1111	29.3832
23	40	56.0563	29.3836
24	90	55.5951	29.3840
25	89	54.8559	29.3844
26	2	55.1864	29.3848
27	10	55.6512	29.3852
28	20	56.7584	29.3856
29	91	58.2788	29.3860
30	1	57.9043	29.3864

Tabla 7.4.0.7. Ejecuciones Nueva Zelanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	0.3312	0.6602
2	33	0.2965	0.6840
3	88	0.2868	0.6882
4	64	0.2926	0.6886
5	55	0.2985	0.6890
6	51	0.2887	0.6894
7	74	0.2815	0.6898
8	56	0.2809	0.6902
9	25	0.2821	0.6906
10	57	0.2848	0.6910
11	21	0.2846	0.6914
12	16	0.2837	0.6918
13	7	0.2846	0.6922
14	83	0.2845	0.6926
15	52	0.2938	0.6930
16	18	0.2865	0.6934
17	39	0.2813	0.6938
18	72	0.2920	0.6942
19	95	0.3007	0.6946
20	93	0.2807	0.6950
21	42	0.2863	0.6954
22	78	0.2807	0.6958
23	40	0.2842	0.6962
24	90	0.2824	0.6966
25	89	0.2858	0.6969
26	2	0.2912	0.6973
27	10	0.2905	0.6977
28	20	0.2858	0.6981
29	91	0.2803	0.6985
30	1	0.2810	0.6989

Tabla 7.4.0.8. Ejecuciones Singapur

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	0.4188	0.8001
2	33	0.3600	0.8239
3	88	0.3661	0.8281
4	64	0.3608	0.8285
5	55	0.3853	0.8289
6	51	0.3780	0.8293
7	74	0.3683	0.8297
8	56	0.3694	0.8301
9	25	0.3677	0.8305
10	57	0.3776	0.8309
11	21	0.3664	0.8313
12	16	0.3614	0.8317
13	7	0.3699	0.8321
14	83	0.3678	0.8325
15	52	0.3653	0.8329
16	18	0.3718	0.8333
17	39	0.3691	0.8337
18	72	0.3645	0.8341
19	95	0.3808	0.8345
20	93	0.3660	0.8349
21	42	0.3708	0.8353
22	78	0.3690	0.8357
23	40	0.3600	0.8361
24	90	0.3592	0.8365
25	89	0.3679	0.8369
26	2	0.3588	0.8372
27	10	0.3691	0.8376
28	20	0.3603	0.8380
29	91	0.3727	0.8384
30	1	0.3670	0.8388

Tabla 7.4.0.9. Ejecuciones Sudáfrica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	36	100.1093	44.5054
2	33	99.8156	44.5291
3	88	98.4328	44.5334
4	64	98.1536	44.5338
5	55	99.1016	44.5342
6	51	99.0964	44.5346
7	74	99.8390	44.5350
8	56	98.4348	44.5354
9	25	101.3207	44.5358
10	57	99.0333	44.5362
11	21	98.8905	44.5366
12	16	102.3217	44.5370
13	7	108.1331	44.5373
14	83	108.8219	44.5377
15	52	106.8869	44.5381
16	18	106.4466	44.5385
17	39	110.1514	44.5389
18	72	107.1344	44.5393
19	95	105.6100	44.5397
20	93	105.5375	44.5401
21	42	104.9162	44.5405
22	78	107.4110	44.5409
23	40	106.0681	44.5413
24	90	105.8779	44.5417
25	89	107.3028	44.5421
26	2	107.9918	44.5425
27	10	105.2622	44.5429
28	20	103.5466	44.5433
29	91	103.6095	44.5437
30	1	105.5746	44.5441

Tabla 7.4.0.10. Ejecuciones USA

Apéndice B

7.5. Tres Bloques

7.5.1. Ejecuciones Macbook

7.5.2. Ejecuciones Windows

7.5.3. Ejecuciones Online

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	39.8834	1443.5615
2	494	42.1422	1443.5980
3	50	51.9753	1443.6010
4	494	62.2508	1443.6014
5	220	84.7213	1443.6018
6	264	90.8278	1443.6022
7	380	90.0060	1443.6026
8	225	86.6590	1443.6030
9	395	88.3257	1443.6034
10	492	88.3927	1443.6038
11	226	80.5521	1443.6042
12	166	42.1246	1443.6046
13	246	37.9679	1443.6050
14	124	36.9479	1443.6054
15	94	42.2059	1443.6058
16	214	39.1747	1443.6061
17	123	36.8300	1443.6065
18	227	41.0890	1443.6069
19	252	39.8744	1443.6073
20	442	38.0970	1443.6077
21	53	41.1137	1443.6081
22	442	38.9724	1443.6085
23	49	36.8167	1443.6089
24	381	39.8273	1443.6093
25	74	40.1570	1443.6097
26	476	37.8393	1443.6101
27	56	40.0121	1443.6105
28	323	39.5115	1443.6109
29	404	37.9418	1443.6113
30	181	39.5405	1443.6117

Tabla 7.5.1.1. Ejecuciones Australia

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	14.6893	641.7037
2	494	12.0618	641.7402
3	50	11.9351	641.7432
4	494	12.4194	641.7436
5	220	11.7029	641.7440
6	264	11.6521	641.7444
7	380	12.4621	641.7448
8	225	12.5932	641.7452
9	395	11.9697	641.7456
10	492	11.7034	641.7460
11	226	11.9157	641.7464
12	166	11.6824	641.7468
13	246	11.4990	641.7472
14	124	12.3103	641.7476
15	94	11.7316	641.7480
16	214	11.6145	641.7484
17	123	12.2096	641.7488
18	227	12.7844	641.7491
19	252	11.7844	641.7495
20	442	11.5690	641.7499
21	53	11.8702	641.7503
22	442	11.3357	641.7507
23	49	11.4970	641.7511
24	381	11.7660	641.7515
25	74	11.6420	641.7519
26	476	11.7411	641.7523
27	56	12.1573	641.7527
28	323	12.6209	641.7531
29	404	11.7017	641.7535
30	181	12.7418	641.7539

Tabla 7.5.1.2. Ejecuciones Canadá

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	11.8031	641.7051
2	494	11.7287	641.7416
3	50	11.6182	641.7446
4	494	12.6235	641.7450
5	220	11.5244	641.7454
6	264	12.3563	641.7458
7	380	13.0857	641.7462
8	225	12.4015	641.7466
9	395	11.9179	641.7470
10	492	13.2173	641.7474
11	226	11.4282	641.7477
12	166	11.9375	641.7481
13	246	11.4869	641.7485
14	124	11.7671	641.7489
15	94	11.6651	641.7493
16	214	12.4245	641.7497
17	123	12.9458	641.7501
18	227	12.2036	641.7505
19	252	11.6760	641.7509
20	442	12.1305	641.7513
21	53	11.4135	641.7517
22	442	11.4497	641.7521
23	49	11.6799	641.7525
24	381	11.7975	641.7529
25	74	11.5857	641.7533
26	476	12.9513	641.7537
27	56	12.9305	641.7541
28	323	12.7073	641.7545
29	404	11.6403	641.7549
30	181	12.0592	641.7553

Tabla 7.5.1.3. Ejecuciones Gran Bretaña

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	503.1189	5773.5747
2	494	422.7724	5773.6112
3	50	389.7225	5773.6141
4	494	390.2231	5773.6145
5	220	413.9742	5773.6149
6	264	392.4382	5773.6153
7	380	414.7932	5773.6157
8	225	390.7701	5773.6161
9	395	399.7029	5773.6165
10	492	413.6865	5773.6169
11	226	398.5466	5773.6173
12	166	398.6644	5773.6177
13	246	392.8839	5773.6181
14	124	388.3481	5773.6185
15	94	390.7062	5773.6189
16	214	394.7313	5773.6193
17	123	386.7843	5773.6197
18	227	409.6642	5773.6201
19	252	385.1677	5773.6205
20	442	387.4466	5773.6209
21	53	386.7338	5773.6213
22	442	391.0803	5773.6217
23	49	388.4775	5773.6221
24	381	396.9918	5773.6225
25	74	391.4213	5773.6229
26	476	393.0378	5773.6233
27	56	406.8595	5773.6237
28	323	401.3270	5773.6241
29	404	381.8468	5773.6245
30	181	382.1991	5773.6249

Tabla 7.5.1.4. Ejecuciones USA

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	12.9854	641.7039
2	494	12.7790	641.7404
3	50	11.8951	641.7433
4	494	12.8587	641.7437
5	220	11.7352	641.7441
6	264	11.5923	641.7445
7	380	12.3675	641.7449
8	225	11.8295	641.7453
9	395	11.5985	641.7457
10	492	12.1571	641.7461
11	226	12.5442	641.7465
12	166	12.2721	641.7469
13	246	11.5010	641.7473
14	124	12.0724	641.7477
15	94	11.4414	641.7481
16	214	11.4279	641.7485
17	123	11.7469	641.7489
18	227	11.5900	641.7493
19	252	11.4902	641.7497
20	442	12.1604	641.7501
21	53	12.8578	641.7505
22	442	12.4613	641.7509
23	49	11.4431	641.7513
24	381	12.0434	641.7517
25	74	12.6297	641.7521
26	476	11.7549	641.7525
27	56	11.6345	641.7529
28	323	12.0219	641.7533
29	404	11.6444	641.7537
30	181	12.2625	641.7541

Tabla 7.5.1.5. Ejecuciones India

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	40.1789	1443.5629
2	494	37.4842	1443.5994
3	50	36.0606	1443.6024
4	494	38.4813	1443.6028
5	220	37.1574	1443.6032
6	264	36.5239	1443.6036
7	380	38.3344	1443.6039
8	225	37.7614	1443.6043
9	395	36.1188	1443.6047
10	492	38.3484	1443.6051
11	226	38.0896	1443.6055
12	166	37.2221	1443.6059
13	246	38.4829	1443.6063
14	124	39.5404	1443.6067
15	94	35.7892	1443.6071
16	214	38.7384	1443.6075
17	123	40.4618	1443.6079
18	227	37.3255	1443.6083
19	252	37.2805	1443.6087
20	442	38.3081	1443.6091
21	53	36.6165	1443.6095
22	442	37.3698	1443.6099
23	49	37.3192	1443.6103
24	381	37.1902	1443.6107
25	74	35.9651	1443.6111
26	476	38.9325	1443.6115
27	56	36.6240	1443.6119
28	323	36.1518	1443.6123
29	404	39.6416	1443.6127
30	181	36.3675	1443.6131

Tabla 7.5.1.6. Ejecuciones Irlanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	12.0745	641.7058
2	494	11.8155	641.7423
3	50	12.7458	641.7453
4	494	12.0405	641.7457
5	220	11.6141	641.7461
6	264	11.7470	641.7465
7	380	11.6513	641.7469
8	225	11.6890	641.7473
9	395	11.7744	641.7477
10	492	12.6525	641.7481
11	226	12.8649	641.7485
12	166	11.8508	641.7489
13	246	12.4820	641.7493
14	124	11.4587	641.7497
15	94	12.1055	641.7501
16	214	11.5222	641.7505
17	123	11.5204	641.7509
18	227	11.6748	641.7512
19	252	11.8742	641.7516
20	442	12.7389	641.7520
21	53	12.8834	641.7524
22	442	12.1042	641.7528
23	49	12.9713	641.7532
24	381	11.5866	641.7536
25	74	11.6168	641.7540
26	476	11.5096	641.7544
27	56	11.5725	641.7548
28	323	11.6520	641.7552
29	404	11.9387	641.7556
30	181	12.6339	641.7560

Tabla 7.5.1.7. Ejecuciones Jamaica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	119.8615	2566.1611
2	494	118.2083	2566.1976
3	50	120.2364	2566.2006
4	494	123.3332	2566.2009
5	220	119.7762	2566.2013
6	264	114.5242	2566.2017
7	380	113.7427	2566.2021
8	225	122.2423	2566.2025
9	395	116.3444	2566.2029
10	492	115.1376	2566.2033
11	226	118.3893	2566.2037
12	166	119.7089	2566.2041
13	246	113.9133	2566.2045
14	124	114.5596	2566.2049
15	94	114.6434	2566.2053
16	214	119.4676	2566.2057
17	123	114.6392	2566.2061
18	227	117.2440	2566.2065
19	252	114.3595	2566.2069
20	442	116.0893	2566.2073
21	53	116.7791	2566.2077
22	442	115.5206	2566.2081
23	49	114.4229	2566.2085
24	381	116.6249	2566.2089
25	74	114.8377	2566.2093
26	476	120.5374	2566.2097
27	56	120.5909	2566.2101
28	323	122.1658	2566.2105
29	404	116.4388	2566.2109
30	181	116.1058	2566.2113

Tabla 7.5.1.8. Ejecuciones Nueva Zelanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	2.5828	160.5872
2	494	2.1746	160.6237
3	50	2.0318	160.6266
4	494	2.0289	160.6270
5	220	2.0495	160.6274
6	264	2.0132	160.6278
7	380	2.0305	160.6282
8	225	2.0421	160.6286
9	395	2.0416	160.6290
10	492	2.0285	160.6294
11	226	2.0217	160.6298
12	166	2.0074	160.6302
13	246	2.0374	160.6306
14	124	2.0220	160.6310
15	94	1.7836	160.6314
16	214	1.7888	160.6318
17	123	1.8405	160.6322
18	227	1.8896	160.6326
19	252	1.9519	160.6330
20	442	1.9157	160.6334
21	53	1.9868	160.6338
22	442	2.0518	160.6342
23	49	1.9492	160.6346
24	381	1.7764	160.6350
25	74	1.7717	160.6354
26	476	1.8485	160.6357
27	56	1.8281	160.6361
28	323	1.8394	160.6365
29	404	1.8266	160.6369
30	181	1.7881	160.6373

Tabla 7.5.1.9. Ejecuciones Singapur

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	2.1487	160.5873
2	494	2.0942	160.6238
3	50	2.0629	160.6268
4	494	2.0326	160.6272
5	220	2.0548	160.6276
6	264	2.0569	160.6280
7	380	2.1549	160.6284
8	225	2.1198	160.6288
9	395	2.2497	160.6292
10	492	2.0207	160.6296
11	226	2.0631	160.6300
12	166	2.0688	160.6304
13	246	2.0542	160.6308
14	124	2.0286	160.6312
15	94	2.0617	160.6316
16	214	2.0450	160.6320
17	123	2.0916	160.6323
18	227	2.0640	160.6327
19	252	2.0589	160.6331
20	442	2.0431	160.6335
21	53	1.8424	160.6339
22	442	2.1057	160.6343
23	49	1.9418	160.6347
24	381	1.8527	160.6351
25	74	1.8818	160.6355
26	476	1.9121	160.6359
27	56	1.8945	160.6363
28	323	1.9335	160.6367
29	404	1.8010	160.6371
30	181	1.8363	160.6375

Tabla 7.5.1.10. Ejecuciones Sudáfrica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	23.0380	1443.5615
2	494	23.0869	1443.5980
3	50	22.9859	1443.6010
4	494	23.2552	1443.6014
5	220	23.1168	1443.6018
6	264	23.0163	1443.6022
7	380	23.1781	1443.6026
8	225	22.9744	1443.6030
9	395	23.1494	1443.6034
10	492	23.3034	1443.6038
11	226	23.0387	1443.6042
12	166	23.6496	1443.6046
13	246	23.0599	1443.6050
14	124	22.9164	1443.6054
15	94	23.2532	1443.6058
16	214	22.9735	1443.6061
17	123	23.2576	1443.6065
18	227	23.0128	1443.6069
19	252	22.8555	1443.6073
20	442	23.0730	1443.6077
21	53	23.0900	1443.6081
22	442	23.5529	1443.6085
23	49	23.3784	1443.6089
24	381	22.9705	1443.6093
25	74	23.5220	1443.6097
26	476	23.0515	1443.6101
27	56	23.0607	1443.6105
28	323	23.1499	1443.6109
29	404	22.9876	1443.6113
30	181	23.3395	1443.6117

Tabla 7.5.2.1. Ejecuciones Australia

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	7.6247	641.7037
2	494	7.5237	641.7402
3	50	7.5132	641.7432
4	494	7.5400	641.7436
5	220	7.4926	641.7440
6	264	7.5256	641.7444
7	380	7.5013	641.7448
8	225	7.5375	641.7452
9	395	7.5192	641.7456
10	492	7.5732	641.7460
11	226	7.4925	641.7464
12	166	7.5108	641.7468
13	246	7.5038	641.7472
14	124	7.5269	641.7476
15	94	7.4880	641.7480
16	214	7.6959	641.7484
17	123	7.6875	641.7488
18	227	7.6633	641.7491
19	252	7.6348	641.7495
20	442	7.4749	641.7499
21	53	7.5978	641.7503
22	442	7.4698	641.7507
23	49	7.5188	641.7511
24	381	7.4929	641.7515
25	74	7.5346	641.7519
26	476	7.5061	641.7523
27	56	7.4622	641.7527
28	323	7.5191	641.7531
29	404	7.4753	641.7535
30	181	7.4812	641.7539

Tabla 7.5.2.2. Ejecuciones Canadá

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	7.4475	641.7051
2	494	7.4157	641.7416
3	50	7.3680	641.7446
4	494	7.4015	641.7450
5	220	7.3970	641.7454
6	264	7.4434	641.7458
7	380	7.3794	641.7462
8	225	7.3580	641.7466
9	395	7.6571	641.7470
10	492	7.4334	641.7474
11	226	7.4208	641.7477
12	166	7.5076	641.7481
13	246	7.4332	641.7485
14	124	7.5341	641.7489
15	94	7.6396	641.7493
16	214	7.4529	641.7497
17	123	7.4395	641.7501
18	227	7.4103	641.7505
19	252	7.4103	641.7509
20	442	7.4034	641.7513
21	53	7.4113	641.7517
22	442	7.4244	641.7521
23	49	7.4473	641.7525
24	381	7.6601	641.7529
25	74	7.3968	641.7533
26	476	7.4053	641.7537
27	56	7.4188	641.7541
28	323	7.3991	641.7545
29	404	7.4102	641.7549
30	181	7.4298	641.7553

Tabla 7.5.2.3. Ejecuciones Gran Bretaña

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	7.5232	641.7039
2	494	7.4652	641.7404
3	50	7.4596	641.7433
4	494	7.4389	641.7437
5	220	7.4174	641.7441
6	264	7.4326	641.7445
7	380	7.4222	641.7449
8	225	7.6475	641.7453
9	395	7.5135	641.7457
10	492	7.4679	641.7461
11	226	7.4533	641.7465
12	166	7.4214	641.7469
13	246	7.4531	641.7473
14	124	7.4242	641.7477
15	94	7.4591	641.7481
16	214	7.4255	641.7485
17	123	7.4233	641.7489
18	227	7.4401	641.7493
19	252	7.4540	641.7497
20	442	7.4476	641.7501
21	53	7.4174	641.7505
22	442	7.4417	641.7509
23	49	7.6496	641.7513
24	381	7.3832	641.7517
25	74	7.3474	641.7521
26	476	7.3691	641.7525
27	56	7.4256	641.7529
28	323	7.3758	641.7533
29	404	7.3733	641.7537
30	181	7.3483	641.7541

Tabla 7.5.2.4. Ejecuciones India

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	23.2184	1443.5629
2	494	22.8688	1443.5994
3	50	23.0243	1443.6024
4	494	22.8853	1443.6028
5	220	23.0673	1443.6032
6	264	22.9473	1443.6036
7	380	22.8294	1443.6039
8	225	22.8344	1443.6043
9	395	23.1084	1443.6047
10	492	23.1190	1443.6051
11	226	22.9109	1443.6055
12	166	23.0024	1443.6059
13	246	23.3048	1443.6063
14	124	22.9316	1443.6067
15	94	22.9325	1443.6071
16	214	22.9733	1443.6075
17	123	22.9163	1443.6079
18	227	22.8956	1443.6083
19	252	22.9617	1443.6087
20	442	22.8788	1443.6091
21	53	23.0034	1443.6095
22	442	23.0274	1443.6099
23	49	22.8842	1443.6103
24	381	23.0432	1443.6107
25	74	23.1248	1443.6111
26	476	23.0937	1443.6115
27	56	23.3602	1443.6119
28	323	23.1548	1443.6123
29	404	23.2014	1443.6127
30	181	23.5341	1443.6131

Tabla 7.5.2.5. Ejecuciones Irlanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	7.4282	641.7058
2	494	7.3469	641.7423
3	50	7.3927	641.7453
4	494	7.3522	641.7457
5	220	7.3407	641.7461
6	264	7.3379	641.7465
7	380	7.6096	641.7469
8	225	7.4238	641.7473
9	395	7.4563	641.7477
10	492	7.3954	641.7481
11	226	7.3740	641.7485
12	166	7.4079	641.7489
13	246	7.4143	641.7493
14	124	7.3658	641.7497
15	94	7.3758	641.7501
16	214	7.3663	641.7505
17	123	7.3939	641.7509
18	227	7.3891	641.7512
19	252	7.6091	641.7516
20	442	7.6125	641.7520
21	53	7.5347	641.7524
22	442	7.6798	641.7528
23	49	7.4160	641.7532
24	381	7.4002	641.7536
25	74	7.4564	641.7540
26	476	7.3978	641.7544
27	56	7.4586	641.7548
28	323	7.5242	641.7552
29	404	7.4150	641.7556
30	181	7.4351	641.7560

Tabla 7.5.2.6. Ejecuciones Jamaica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	54.3414	2566.1611
2	494	52.2945	2566.1976
3	50	51.8744	2566.2006
4	494	51.6832	2566.2009
5	220	53.3833	2566.2013
6	264	51.8849	2566.2017
7	380	56.3990	2566.2021
8	225	52.5048	2566.2025
9	395	51.3456	2566.2029
10	492	51.0499	2566.2033
11	226	52.4719	2566.2037
12	166	51.6003	2566.2041
13	246	51.6575	2566.2045
14	124	51.7888	2566.2049
15	94	52.3099	2566.2053
16	214	52.3889	2566.2057
17	123	51.5992	2566.2061
18	227	51.7372	2566.2065
19	252	52.3096	2566.2069
20	442	52.9065	2566.2073
21	53	52.5949	2566.2077
22	442	52.4240	2566.2081
23	49	51.9163	2566.2085
24	381	51.9813	2566.2089
25	74	52.4891	2566.2093
26	476	51.8339	2566.2097
27	56	51.5638	2566.2101
28	323	51.5937	2566.2105
29	404	51.8690	2566.2109
30	181	52.8373	2566.2113

Tabla 7.5.2.7. Ejecuciones Nueva Zelanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	1.1308	160.5872
2	494	1.0835	160.6237
3	50	1.0966	160.6266
4	494	1.1047	160.6270
5	220	1.1929	160.6274
6	264	1.0850	160.6278
7	380	1.0893	160.6282
8	225	1.1033	160.6286
9	395	1.0815	160.6290
10	492	1.0865	160.6294
11	226	1.0908	160.6298
12	166	1.0876	160.6302
13	246	1.1042	160.6306
14	124	1.0818	160.6310
15	94	1.0971	160.6314
16	214	1.0797	160.6318
17	123	1.0904	160.6322
18	227	1.0806	160.6326
19	252	1.0788	160.6330
20	442	1.1260	160.6334
21	53	1.0780	160.6338
22	442	1.0872	160.6342
23	49	1.0766	160.6346
24	381	1.0819	160.6350
25	74	1.0838	160.6354
26	476	1.0857	160.6357
27	56	1.0887	160.6361
28	323	1.0844	160.6365
29	404	1.0962	160.6369
30	181	1.0854	160.6373

Tabla 7.5.2.8. Ejecuciones Singapur

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	1.1438	160.5873
2	494	1.0982	160.6238
3	50	1.1134	160.6268
4	494	1.1788	160.6272
5	220	1.0993	160.6276
6	264	1.0924	160.6280
7	380	1.0957	160.6284
8	225	1.0864	160.6288
9	395	1.0937	160.6292
10	492	1.0807	160.6296
11	226	1.0967	160.6300
12	166	1.0899	160.6304
13	246	1.0918	160.6308
14	124	1.0994	160.6312
15	94	1.0849	160.6316
16	214	1.0882	160.6320
17	123	1.0810	160.6323
18	227	1.0960	160.6327
19	252	1.1506	160.6331
20	442	1.0839	160.6335
21	53	1.0836	160.6339
22	442	1.0919	160.6343
23	49	1.0763	160.6347
24	381	1.0799	160.6351
25	74	1.0792	160.6355
26	476	1.0822	160.6359
27	56	1.0776	160.6363
28	323	1.0935	160.6367
29	404	1.0851	160.6371
30	181	1.0910	160.6375

Tabla 7.5.2.9. Ejecuciones Sudáfrica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	193.5934	5773.5747
2	494	217.9005	5773.6112
3	50	210.4369	5773.6141
4	494	207.6210	5773.6145
5	220	208.0157	5773.6149
6	264	210.3865	5773.6153
7	380	208.2264	5773.6157
8	225	206.2674	5773.6161
9	395	205.5877	5773.6165
10	492	205.3390	5773.6169
11	226	208.0705	5773.6173
12	166	205.2700	5773.6177
13	246	201.6678	5773.6181
14	124	209.0466	5773.6185
15	94	204.3677	5773.6189
16	214	204.9835	5773.6193
17	123	204.4909	5773.6197
18	227	206.4308	5773.6201
19	252	206.0497	5773.6205
20	442	206.6907	5773.6209
21	53	204.3839	5773.6213
22	442	205.6935	5773.6217
23	49	211.6385	5773.6221
24	381	207.4570	5773.6225
25	74	210.0140	5773.6229
26	476	210.0777	5773.6233
27	56	210.3805	5773.6237
28	323	205.3188	5773.6241
29	404	210.7834	5773.6245
30	181	206.7182	5773.6249

Tabla 7.5.2.10. Ejecuciones USA

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	17.5506	1443.5615
2	494	17.4711	1443.5980
3	50	17.4829	1443.6010
4	494	17.9603	1443.6014
5	220	18.0916	1443.6018
6	264	18.4175	1443.6022
7	380	18.2678	1443.6026
8	225	18.5367	1443.6030
9	395	18.3625	1443.6034
10	492	18.1493	1443.6038
11	226	17.8719	1443.6042
12	166	21.8654	1443.6046
13	246	17.8448	1443.6050
14	124	17.4176	1443.6054
15	94	17.6936	1443.6058
16	214	17.5890	1443.6061
17	123	17.5161	1443.6065
18	227	17.4889	1443.6069
19	252	17.7667	1443.6073
20	442	17.5582	1443.6077
21	53	17.7995	1443.6081
22	442	17.3976	1443.6085
23	49	17.4761	1443.6089
24	381	17.6691	1443.6093
25	74	17.5738	1443.6097
26	476	17.4884	1443.6101
27	56	17.5057	1443.6105
28	323	17.5674	1443.6109
29	404	17.9488	1443.6113
30	181	20.7977	1443.6117

Tabla 7.5.3.1. Ejecuciones Australia

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	7.9378	641.7037
2	494	7.9112	641.7402
3	50	7.9169	641.7432
4	494	8.0394	641.7436
5	220	7.9100	641.7440
6	264	7.9855	641.7444
7	380	7.8732	641.7448
8	225	7.9028	641.7452
9	395	7.8486	641.7456
10	492	7.9042	641.7460
11	226	8.2977	641.7464
12	166	8.0978	641.7468
13	246	8.0009	641.7472
14	124	8.0223	641.7476
15	94	7.9816	641.7480
16	214	8.0093	641.7484
17	123	7.9954	641.7488
18	227	8.0283	641.7491
19	252	7.9260	641.7495
20	442	7.8942	641.7499
21	53	8.0342	641.7503
22	442	8.0706	641.7507
23	49	8.1062	641.7511
24	381	8.2144	641.7515
25	74	8.0612	641.7519
26	476	8.0435	641.7523
27	56	8.0619	641.7527
28	323	8.0851	641.7531
29	404	7.8660	641.7535
30	181	7.9009	641.7539

Tabla 7.5.3.2. Ejecuciones Canadá

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	8.2493	641.7051
2	494	8.1189	641.7416
3	50	8.1716	641.7446
4	494	8.0510	641.7450
5	220	8.0809	641.7454
6	264	7.9855	641.7458
7	380	7.9716	641.7462
8	225	7.9468	641.7466
9	395	7.9994	641.7470
10	492	8.5558	641.7474
11	226	8.1423	641.7477
12	166	8.0682	641.7481
13	246	8.0342	641.7485
14	124	8.1506	641.7489
15	94	8.6957	641.7493
16	214	8.3296	641.7497
17	123	8.3313	641.7501
18	227	10.4663	641.7505
19	252	9.8958	641.7509
20	442	8.1792	641.7513
21	53	8.4610	641.7517
22	442	8.0620	641.7521
23	49	8.2625	641.7525
24	381	9.3801	641.7529
25	74	10.2786	641.7533
26	476	8.9998	641.7537
27	56	8.0788	641.7541
28	323	8.0964	641.7545
29	404	8.1072	641.7549
30	181	8.2185	641.7553

Tabla 7.5.3.3. Ejecuciones Gran Bretaña

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	115.2660	5773.5747
2	494	115.7653	5773.6112
3	50	120.1941	5773.6141
4	494	116.9952	5773.6145
5	220	115.5716	5773.6149
6	264	118.4727	5773.6153
7	380	117.7409	5773.6157
8	225	115.9715	5773.6161
9	395	114.6044	5773.6165
10	492	114.2720	5773.6169
11	226	114.5963	5773.6173
12	166	118.7947	5773.6177
13	246	117.4609	5773.6181
14	124	116.0320	5773.6185
15	94	116.1758	5773.6189
16	214	114.3143	5773.6193
17	123	117.7591	5773.6197
18	227	116.2560	5773.6201
19	252	115.0675	5773.6205
20	442	117.5192	5773.6209
21	53	118.4528	5773.6213
22	442	118.2361	5773.6217
23	49	118.4972	5773.6221
24	381	123.7766	5773.6225
25	74	122.0181	5773.6229
26	476	122.0099	5773.6233
27	56	120.5917	5773.6237
28	323	118.6733	5773.6241
29	404	118.2043	5773.6245
30	181	119.0359	5773.6249

Tabla 7.5.3.4. Ejecuciones USA

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	8.0834	641.7039
2	494	7.8896	641.7404
3	50	7.9298	641.7433
4	494	7.8716	641.7437
5	220	7.9031	641.7441
6	264	7.9721	641.7445
7	380	7.8954	641.7449
8	225	7.8997	641.7453
9	395	7.9423	641.7457
10	492	7.9240	641.7461
11	226	7.9522	641.7465
12	166	7.9177	641.7469
13	246	7.9816	641.7473
14	124	7.9248	641.7477
15	94	7.9382	641.7481
16	214	7.9804	641.7485
17	123	7.9178	641.7489
18	227	8.0444	641.7493
19	252	7.9253	641.7497
20	442	7.9186	641.7501
21	53	8.0086	641.7505
22	442	7.9523	641.7509
23	49	7.8848	641.7513
24	381	8.1430	641.7517
25	74	7.9019	641.7521
26	476	7.8682	641.7525
27	56	7.9069	641.7529
28	323	7.8528	641.7533
29	404	7.8553	641.7537
30	181	7.8750	641.7541

Tabla 7.5.3.5. Ejecuciones India

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	19.6757	1443.5629
2	494	18.8086	1443.5994
3	50	19.1252	1443.6024
4	494	24.3620	1443.6028
5	220	24.2944	1443.6032
6	264	24.0308	1443.6036
7	380	21.8177	1443.6039
8	225	22.1115	1443.6043
9	395	20.7522	1443.6047
10	492	21.3597	1443.6051
11	226	22.9551	1443.6055
12	166	22.4592	1443.6059
13	246	22.0762	1443.6063
14	124	22.0815	1443.6067
15	94	22.9095	1443.6071
16	214	21.3703	1443.6075
17	123	22.1096	1443.6079
18	227	22.0060	1443.6083
19	252	22.5108	1443.6087
20	442	23.3610	1443.6091
21	53	21.4608	1443.6095
22	442	21.4370	1443.6099
23	49	22.1533	1443.6103
24	381	21.8136	1443.6107
25	74	21.1490	1443.6111
26	476	21.4441	1443.6115
27	56	22.3418	1443.6119
28	323	21.4469	1443.6123
29	404	21.5115	1443.6127
30	181	21.6388	1443.6131

Tabla 7.5.3.6. Ejecuciones Irlanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	8.2585	641.7058
2	494	6.9006	641.7423
3	50	6.9905	641.7453
4	494	7.4405	641.7457
5	220	7.1268	641.7461
6	264	7.1230	641.7465
7	380	7.1092	641.7469
8	225	7.0340	641.7473
9	395	7.0850	641.7477
10	492	7.2837	641.7481
11	226	7.1494	641.7485
12	166	7.2678	641.7489
13	246	7.1146	641.7493
14	124	6.9250	641.7497
15	94	7.0085	641.7501
16	214	6.9580	641.7505
17	123	6.8932	641.7509
18	227	7.1389	641.7512
19	252	6.9521	641.7516
20	442	6.9247	641.7520
21	53	6.9412	641.7524
22	442	7.3863	641.7528
23	49	7.9326	641.7532
24	381	7.9460	641.7536
25	74	8.0399	641.7540
26	476	8.0164	641.7544
27	56	7.9438	641.7548
28	323	8.0007	641.7552
29	404	7.9733	641.7556
30	181	7.9585	641.7560

Tabla 7.5.3.7. Ejecuciones Jamaica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	37.8860	2566.1611
2	494	37.0387	2566.1976
3	50	37.8301	2566.2006
4	494	37.9174	2566.2009
5	220	38.0171	2566.2013
6	264	38.1264	2566.2017
7	380	37.5737	2566.2021
8	225	38.3070	2566.2025
9	395	38.1308	2566.2029
10	492	37.9013	2566.2033
11	226	37.8465	2566.2037
12	166	37.8745	2566.2041
13	246	37.3296	2566.2045
14	124	37.1493	2566.2049
15	94	37.5066	2566.2053
16	214	37.2276	2566.2057
17	123	37.2656	2566.2061
18	227	37.5917	2566.2065
19	252	42.9201	2566.2069
20	442	36.9776	2566.2073
21	53	37.0376	2566.2077
22	442	42.7423	2566.2081
23	49	37.0428	2566.2085
24	381	40.6504	2566.2089
25	74	43.7420	2566.2093
26	476	46.4783	2566.2097
27	56	46.0227	2566.2101
28	323	46.6266	2566.2105
29	404	46.0656	2566.2109
30	181	44.1840	2566.2113

Tabla 7.5.3.8. Ejecuciones Nueva Zelanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	1.3039	160.5872
2	494	1.1720	160.6237
3	50	1.1757	160.6266
4	494	1.1721	160.6270
5	220	1.1865	160.6274
6	264	1.1832	160.6278
7	380	1.1903	160.6282
8	225	1.1906	160.6286
9	395	1.2286	160.6290
10	492	1.1745	160.6294
11	226	1.1837	160.6298
12	166	1.1836	160.6302
13	246	1.1832	160.6306
14	124	1.1722	160.6310
15	94	1.1803	160.6314
16	214	1.1772	160.6318
17	123	1.1917	160.6322
18	227	1.2453	160.6326
19	252	1.1751	160.6330
20	442	1.1762	160.6334
21	53	1.2056	160.6338
22	442	1.2460	160.6342
23	49	1.1810	160.6346
24	381	1.2165	160.6350
25	74	1.1906	160.6354
26	476	1.1858	160.6357
27	56	1.2130	160.6361
28	323	1.2037	160.6365
29	404	1.1729	160.6369
30	181	1.1895	160.6373

Tabla 7.5.3.9. Ejecuciones Singapur

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	1.2288	160.5873
2	494	1.2418	160.6238
3	50	1.2042	160.6268
4	494	1.2214	160.6272
5	220	1.2149	160.6276
6	264	1.1972	160.6280
7	380	1.2140	160.6284
8	225	1.2028	160.6288
9	395	1.1903	160.6292
10	492	1.2068	160.6296
11	226	1.1996	160.6300
12	166	1.2316	160.6304
13	246	1.1846	160.6308
14	124	1.1985	160.6312
15	94	1.1952	160.6316
16	214	1.2400	160.6320
17	123	1.1786	160.6323
18	227	1.1908	160.6327
19	252	1.1999	160.6331
20	442	1.1757	160.6335
21	53	1.1845	160.6339
22	442	1.2005	160.6343
23	49	1.1901	160.6347
24	381	1.2037	160.6351
25	74	1.1906	160.6355
26	476	1.1817	160.6359
27	56	1.2074	160.6363
28	323	1.2099	160.6367
29	404	1.1806	160.6371
30	181	1.2059	160.6375

Tabla 7.5.3.10. Ejecuciones Sudáfrica

7.6. Treinta Bloques

7.6.1. Ejecuciones Macbook

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.7940	14.6634
2	494	0.5490	14.6999
3	50	0.5211	14.7028
4	494	0.5267	14.7032
5	220	0.5605	14.7036
6	264	0.5576	14.7040
7	380	0.5168	14.7044
8	225	0.6728	14.7048
9	395	0.5227	14.7052
10	492	0.5156	14.7056
11	226	0.5146	14.7060
12	166	0.5311	14.7064
13	246	0.5433	14.7068
14	124	0.5261	14.7072
15	94	0.5148	14.7076
16	214	0.5209	14.7080
17	123	0.5292	14.7084
18	227	0.5279	14.7088
19	252	0.5416	14.7092
20	442	0.5437	14.7096
21	53	0.5178	14.7100
22	442	0.5253	14.7104
23	49	0.5127	14.7108
24	381	0.5295	14.7112
25	74	0.5117	14.7116
26	476	0.5321	14.7120
27	56	0.5212	14.7124
28	323	0.5166	14.7128
29	404	0.5083	14.7132
30	181	0.5195	14.7136

Tabla 7.6.1.1. Ejecuciones Australia

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.7763	6.6373
2	494	0.4597	6.6738
3	50	0.1666	6.6768
4	494	0.1622	6.6772
5	220	0.2464	6.6776
6	264	0.2905	6.6780
7	380	0.2041	6.6784
8	225	0.1914	6.6788
9	395	0.1548	6.6792
10	492	0.1613	6.6796
11	226	0.2213	6.6800
12	166	0.1791	6.6804
13	246	0.1625	6.6807
14	124	0.1646	6.6811
15	94	0.1604	6.6815
16	214	0.1622	6.6819
17	123	0.2086	6.6823
18	227	0.1807	6.6827
19	252	0.1590	6.6831
20	442	0.1581	6.6835
21	53	0.1610	6.6839
22	442	0.1631	6.6843
23	49	0.1833	6.6847
24	381	0.1656	6.6851
25	74	0.1935	6.6855
26	476	0.1662	6.6859
27	56	0.1718	6.6863
28	323	0.1730	6.6867
29	404	0.1571	6.6871
30	181	0.1597	6.6875

Tabla 7.6.1.2. Ejecuciones Canadá

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.3280	6.6387
2	494	0.2447	6.6752
3	50	0.2549	6.6782
4	494	0.1693	6.6786
5	220	0.2100	6.6790
6	264	0.3106	6.6794
7	380	0.2840	6.6797
8	225	0.1881	6.6801
9	395	0.2113	6.6805
10	492	0.1683	6.6809
11	226	0.3637	6.6813
12	166	0.2905	6.6817
13	246	0.2232	6.6821
14	124	0.1907	6.6825
15	94	0.2273	6.6829
16	214	0.1799	6.6833
17	123	0.2205	6.6837
18	227	0.1920	6.6841
19	252	0.1786	6.6845
20	442	0.1828	6.6849
21	53	0.1719	6.6853
22	442	0.1724	6.6857
23	49	0.2107	6.6861
24	381	0.1928	6.6865
25	74	0.1797	6.6869
26	476	0.2265	6.6873
27	56	0.2907	6.6877
28	323	0.4012	6.6881
29	404	0.1916	6.6885
30	181	0.2760	6.6889

Tabla 7.6.1.3. Ejecuciones Gran Bretaña

7.6.2. Ejecuciones Windows

7.6.3. Ejecuciones Online

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.2773	6.6375
2	494	0.2044	6.6740
3	50	0.1784	6.6769
4	494	0.1986	6.6773
5	220	0.1995	6.6777
6	264	0.2468	6.6781
7	380	0.2446	6.6785
8	225	0.2696	6.6789
9	395	0.1724	6.6793
10	492	0.2272	6.6797
11	226	0.2217	6.6801
12	166	0.2220	6.6805
13	246	0.2123	6.6809
14	124	0.2363	6.6813
15	94	0.3320	6.6817
16	214	0.2537	6.6821
17	123	0.2626	6.6825
18	227	0.2491	6.6829
19	252	0.4881	6.6833
20	442	0.2705	6.6837
21	53	0.2760	6.6841
22	442	0.6920	6.6845
23	49	0.8929	6.6849
24	381	1.0680	6.6853
25	74	0.4026	6.6857
26	476	0.4053	6.6861
27	56	0.4566	6.6865
28	323	1.4407	6.6869
29	404	1.0932	6.6872
30	181	1.2401	6.6876

Tabla 7.6.1.4. Ejecuciones India

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.6419	14.6647
2	494	0.5554	14.7013
3	50	0.4986	14.7042
4	494	0.5383	14.7046
5	220	0.5227	14.7050
6	264	0.5348	14.7054
7	380	0.5103	14.7058
8	225	0.5079	14.7062
9	395	0.5481	14.7066
10	492	0.5500	14.7070
11	226	0.6917	14.7074
12	166	0.5532	14.7078
13	246	0.5511	14.7082
14	124	0.5316	14.7086
15	94	0.5459	14.7090
16	214	0.5302	14.7094
17	123	0.5125	14.7098
18	227	0.5729	14.7102
19	252	0.5683	14.7106
20	442	0.4452	14.7110
21	53	0.4778	14.7114
22	442	0.4415	14.7118
23	49	0.4494	14.7122
24	381	0.4448	14.7125
25	74	0.4748	14.7129
26	476	0.4622	14.7133
27	56	0.4800	14.7137
28	323	0.4760	14.7141
29	404	0.4888	14.7145
30	181	0.4923	14.7149

Tabla 7.6.1.5. Ejecuciones Irlanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.3213	6.6394
2	494	0.2681	6.6759
3	50	0.2933	6.6789
4	494	0.3000	6.6793
5	220	0.4195	6.6797
6	264	0.3972	6.6801
7	380	0.2827	6.6805
8	225	0.2028	6.6809
9	395	0.1741	6.6813
10	492	0.1770	6.6817
11	226	0.1754	6.6821
12	166	0.1919	6.6825
13	246	0.1935	6.6828
14	124	0.1778	6.6832
15	94	0.1804	6.6836
16	214	0.1751	6.6840
17	123	0.1805	6.6844
18	227	0.1798	6.6848
19	252	0.1808	6.6852
20	442	0.1904	6.6856
21	53	0.1773	6.6860
22	442	0.1794	6.6864
23	49	0.1801	6.6868
24	381	0.1721	6.6872
25	74	0.1829	6.6876
26	476	0.1819	6.6880
27	56	0.1923	6.6884
28	323	0.1762	6.6888
29	404	0.1829	6.6892
30	181	0.1747	6.6896

Tabla 7.6.1.6. Ejecuciones Jamaica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	1.3604	25.8989
2	494	1.1504	25.9354
3	50	1.0614	25.9383
4	494	1.0376	25.9387
5	220	1.1729	25.9391
6	264	1.1308	25.9395
7	380	1.0815	25.9399
8	225	1.0154	25.9403
9	395	1.0667	25.9407
10	492	1.0713	25.9411
11	226	1.0640	25.9415
12	166	1.0643	25.9419
13	246	1.0715	25.9423
14	124	1.1083	25.9427
15	94	1.0832	25.9431
16	214	1.0191	25.9435
17	123	1.0407	25.9439
18	227	1.0299	25.9443
19	252	1.0810	25.9447
20	442	1.0497	25.9451
21	53	1.0116	25.9455
22	442	1.1115	25.9459
23	49	1.0633	25.9463
24	381	1.1037	25.9467
25	74	1.1170	25.9471
26	476	1.5752	25.9475
27	56	1.0875	25.9479
28	323	1.0136	25.9482
29	404	1.0284	25.9486
30	181	1.1367	25.9490

Tabla 7.6.1.7. Ejecuciones Nueva Zelanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1179	1.8201
2	494	0.0827	1.8566
3	50	0.0586	1.8596
4	494	0.0727	1.8600
5	220	0.0634	1.8604
6	264	0.0637	1.8608
7	380	0.0621	1.8612
8	225	0.0627	1.8616
9	395	0.0628	1.8620
10	492	0.0610	1.8624
11	226	0.0597	1.8628
12	166	0.0575	1.8632
13	246	0.0542	1.8636
14	124	0.0453	1.8640
15	94	0.0418	1.8644
16	214	1.2734	1.8647
17	123	0.0532	1.8651
18	227	0.0494	1.8655
19	252	0.0481	1.8659
20	442	0.0563	1.8663
21	53	0.0562	1.8667
22	442	0.0424	1.8671
23	49	0.0381	1.8675
24	381	0.0533	1.8679
25	74	0.0465	1.8683
26	476	0.0384	1.8687
27	56	0.0403	1.8691
28	323	0.0445	1.8695
29	404	0.0446	1.8699
30	181	0.0572	1.8703

Tabla 7.6.1.8. Ejecuciones Singapur

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1600	1.8203
2	494	0.0667	1.8568
3	50	0.0681	1.8598
4	494	0.0848	1.8602
5	220	0.0742	1.8606
6	264	0.0822	1.8610
7	380	0.0760	1.8613
8	225	0.0780	1.8617
9	395	0.0708	1.8621
10	492	0.0670	1.8625
11	226	0.0633	1.8629
12	166	0.0545	1.8633
13	246	0.0533	1.8637
14	124	0.0837	1.8641
15	94	0.0738	1.8645
16	214	0.0749	1.8649
17	123	0.0519	1.8653
18	227	0.0572	1.8657
19	252	0.0496	1.8661
20	442	0.0496	1.8665
21	53	0.0538	1.8669
22	442	0.0477	1.8673
23	49	0.0502	1.8677
24	381	0.0606	1.8681
25	74	0.0572	1.8685
26	476	0.0660	1.8689
27	56	0.0503	1.8693
28	323	0.0436	1.8697
29	404	0.0536	1.8701
30	181	0.0434	1.8705

Tabla 7.6.1.9. Ejecuciones Sudáfrica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	3.6086	57.9872
2	494	3.1649	58.0238
3	50	3.2711	58.0267
4	494	3.2000	58.0271
5	220	3.6782	58.0275
6	264	3.3553	58.0279
7	380	3.5140	58.0283
8	225	3.2809	58.0287
9	395	3.5310	58.0291
10	492	3.7984	58.0295
11	226	3.5662	58.0299
12	166	3.2771	58.0303
13	246	3.4526	58.0307
14	124	3.2400	58.0311
15	94	3.6582	58.0315
16	214	3.4656	58.0319
17	123	3.4630	58.0323
18	227	3.7783	58.0327
19	252	3.3635	58.0331
20	442	3.4115	58.0335
21	53	3.9556	58.0339
22	442	4.0594	58.0343
23	49	4.2580	58.0346
24	381	5.1931	58.0350
25	74	5.6690	58.0354
26	476	5.9158	58.0358
27	56	5.7867	58.0362
28	323	6.5882	58.0366
29	404	5.7115	58.0370
30	181	6.5069	58.0374

Tabla 7.6.1.10. Ejecuciones USA

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.3651	14.6634
2	494	0.3314	14.6999
3	50	0.3005	14.7028
4	494	0.3414	14.7032
5	220	0.3426	14.7036
6	264	0.3591	14.7040
7	380	0.4158	14.7044
8	225	0.3218	14.7048
9	395	0.3207	14.7052
10	492	0.3296	14.7056
11	226	0.3257	14.7060
12	166	0.4377	14.7064
13	246	0.3147	14.7068
14	124	0.3189	14.7072
15	94	0.3319	14.7076
16	214	0.3474	14.7080
17	123	0.3781	14.7084
18	227	0.3704	14.7088
19	252	0.3290	14.7092
20	442	0.3209	14.7096
21	53	0.3492	14.7100
22	442	0.3316	14.7104
23	49	0.3864	14.7108
24	381	0.3165	14.7112
25	74	0.4705	14.7116
26	476	0.3614	14.7120
27	56	0.3252	14.7124
28	323	0.3463	14.7128
29	404	0.3678	14.7132
30	181	0.3396	14.7136

Tabla 7.6.2.1. Ejecuciones Australia

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1463	6.6373
2	494	0.1121	6.6738
3	50	0.1125	6.6768
4	494	0.1097	6.6772
5	220	0.1306	6.6776
6	264	0.1438	6.6780
7	380	0.1257	6.6784
8	225	0.1130	6.6788
9	395	0.1077	6.6792
10	492	0.1312	6.6796
11	226	0.1045	6.6800
12	166	0.1075	6.6804
13	246	0.1106	6.6807
14	124	0.1057	6.6811
15	94	0.1046	6.6815
16	214	0.1189	6.6819
17	123	0.1640	6.6823
18	227	0.1095	6.6827
19	252	0.1418	6.6831
20	442	0.1073	6.6835
21	53	0.1225	6.6839
22	442	0.1164	6.6843
23	49	0.1051	6.6847
24	381	0.1110	6.6851
25	74	0.1333	6.6855
26	476	0.1135	6.6859
27	56	0.3506	6.6863
28	323	0.1328	6.6867
29	404	0.1081	6.6871
30	181	0.1036	6.6875

Tabla 7.6.2.2. Ejecuciones Canadá

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1542	6.6387
2	494	0.1102	6.6752
3	50	0.1088	6.6782
4	494	0.1092	6.6786
5	220	0.1354	6.6790
6	264	0.1367	6.6794
7	380	0.1283	6.6797
8	225	0.1165	6.6801
9	395	0.1130	6.6805
10	492	0.1202	6.6809
11	226	0.1102	6.6813
12	166	0.1251	6.6817
13	246	0.1082	6.6821
14	124	0.1173	6.6825
15	94	0.1190	6.6829
16	214	0.1128	6.6833
17	123	0.1203	6.6837
18	227	0.1057	6.6841
19	252	0.1100	6.6845
20	442	0.1176	6.6849
21	53	0.1089	6.6853
22	442	0.1015	6.6857
23	49	0.1063	6.6861
24	381	0.1046	6.6865
25	74	0.1311	6.6869
26	476	0.1001	6.6873
27	56	0.1062	6.6877
28	323	0.1137	6.6881
29	404	0.1112	6.6885
30	181	0.1280	6.6889

Tabla 7.6.2.3. Ejecuciones Gran Bretaña

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1608	6.6375
2	494	0.1105	6.6740
3	50	0.1247	6.6769
4	494	0.1140	6.6773
5	220	0.1408	6.6777
6	264	0.1419	6.6781
7	380	0.1099	6.6785
8	225	0.1046	6.6789
9	395	0.1060	6.6793
10	492	0.1410	6.6797
11	226	0.1378	6.6801
12	166	0.1129	6.6805
13	246	0.1005	6.6809
14	124	0.1149	6.6813
15	94	0.1023	6.6817
16	214	0.1316	6.6821
17	123	0.1178	6.6825
18	227	0.1521	6.6829
19	252	0.1719	6.6833
20	442	0.1250	6.6837
21	53	0.1277	6.6841
22	442	0.1077	6.6845
23	49	0.1081	6.6849
24	381	0.1103	6.6853
25	74	0.1157	6.6857
26	476	0.1134	6.6861
27	56	0.1168	6.6865
28	323	0.1047	6.6869
29	404	0.1067	6.6872
30	181	0.1217	6.6876

Tabla 7.6.2.4. Ejecuciones India

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.3673	14.6647
2	494	0.3701	14.7013
3	50	0.3809	14.7042
4	494	0.3152	14.7046
5	220	0.3475	14.7050
6	264	0.3490	14.7054
7	380	0.3051	14.7058
8	225	0.3372	14.7062
9	395	0.4482	14.7066
10	492	0.3497	14.7070
11	226	0.3479	14.7074
12	166	0.3432	14.7078
13	246	0.3368	14.7082
14	124	0.3997	14.7086
15	94	0.3038	14.7090
16	214	0.3109	14.7094
17	123	0.3164	14.7098
18	227	0.3533	14.7102
19	252	0.3150	14.7106
20	442	0.3102	14.7110
21	53	0.3478	14.7114
22	442	0.3368	14.7118
23	49	0.3814	14.7122
24	381	0.3320	14.7125
25	74	0.3199	14.7129
26	476	0.3555	14.7133
27	56	0.3234	14.7137
28	323	0.4026	14.7141
29	404	0.3980	14.7145
30	181	0.3597	14.7149

Tabla 7.6.2.5. Ejecuciones Irlanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1542	6.6394
2	494	0.1050	6.6759
3	50	0.1230	6.6789
4	494	0.1271	6.6793
5	220	0.1229	6.6797
6	264	0.1483	6.6801
7	380	0.1237	6.6805
8	225	0.1153	6.6809
9	395	0.1041	6.6813
10	492	0.1341	6.6817
11	226	0.1022	6.6821
12	166	0.1080	6.6825
13	246	0.1050	6.6828
14	124	0.1090	6.6832
15	94	0.1027	6.6836
16	214	0.1241	6.6840
17	123	0.1090	6.6844
18	227	0.1159	6.6848
19	252	0.1188	6.6852
20	442	0.1264	6.6856
21	53	0.1187	6.6860
22	442	0.1087	6.6864
23	49	0.1119	6.6868
24	381	0.1052	6.6872
25	74	0.1096	6.6876
26	476	0.1127	6.6880
27	56	0.1038	6.6884
28	323	0.1160	6.6888
29	404	0.1065	6.6892
30	181	0.1134	6.6896

Tabla 7.6.2.6. Ejecuciones Jamaica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.8521	25.8989
2	494	0.8508	25.9354
3	50	0.7344	25.9383
4	494	0.8066	25.9387
5	220	0.8086	25.9391
6	264	0.8312	25.9395
7	380	0.7685	25.9399
8	225	0.7982	25.9403
9	395	0.9152	25.9407
10	492	0.7165	25.9411
11	226	0.8609	25.9415
12	166	0.7218	25.9419
13	246	0.7265	25.9423
14	124	1.0133	25.9427
15	94	0.8234	25.9431
16	214	0.7697	25.9435
17	123	0.8622	25.9439
18	227	0.8907	25.9443
19	252	0.8225	25.9447
20	442	0.8078	25.9451
21	53	0.7686	25.9455
22	442	0.8707	25.9459
23	49	0.6857	25.9463
24	381	0.7121	25.9467
25	74	0.7764	25.9471
26	476	0.7971	25.9475
27	56	0.7461	25.9479
28	323	0.8106	25.9482
29	404	0.8149	25.9486
30	181	0.7566	25.9490

Tabla 7.6.2.7. Ejecuciones Nueva Zelanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0541	1.8201
2	494	0.0727	1.8566
3	50	0.0340	1.8596
4	494	0.0241	1.8600
5	220	0.0271	1.8604
6	264	0.0411	1.8608
7	380	0.0404	1.8612
8	225	0.0304	1.8616
9	395	0.0411	1.8620
10	492	0.0333	1.8624
11	226	0.0242	1.8628
12	166	0.0251	1.8632
13	246	0.0264	1.8636
14	124	0.0261	1.8640
15	94	0.0233	1.8644
16	214	0.0258	1.8647
17	123	0.0254	1.8651
18	227	0.0279	1.8655
19	252	0.0255	1.8659
20	442	0.0252	1.8663
21	53	0.0226	1.8667
22	442	0.0278	1.8671
23	49	0.0217	1.8675
24	381	0.0313	1.8679
25	74	0.0282	1.8683
26	476	0.0233	1.8687
27	56	0.0426	1.8691
28	323	0.0289	1.8695
29	404	0.0241	1.8699
30	181	0.0236	1.8703

Tabla 7.6.2.8. Ejecuciones Singapur

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0534	1.8203
2	494	0.0400	1.8568
3	50	0.0249	1.8598
4	494	0.0341	1.8602
5	220	0.0610	1.8606
6	264	0.0378	1.8610
7	380	0.0408	1.8613
8	225	0.0389	1.8617
9	395	0.0349	1.8621
10	492	0.0243	1.8625
11	226	0.0288	1.8629
12	166	0.0270	1.8633
13	246	0.0369	1.8637
14	124	0.0256	1.8641
15	94	0.0279	1.8645
16	214	0.0325	1.8649
17	123	0.0267	1.8653
18	227	0.0221	1.8657
19	252	0.0274	1.8661
20	442	0.0229	1.8665
21	53	0.0254	1.8669
22	442	0.0225	1.8673
23	49	0.0200	1.8677
24	381	0.0274	1.8681
25	74	0.0344	1.8685
26	476	0.0251	1.8689
27	56	0.0284	1.8693
28	323	0.0235	1.8697
29	404	0.0286	1.8701
30	181	0.0237	1.8705

Tabla 7.6.2.9. Ejecuciones Sudáfrica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	2.5439	57.9872
2	494	2.3628	58.0238
3	50	2.4777	58.0267
4	494	2.6253	58.0271
5	220	2.3207	58.0275
6	264	2.5208	58.0279
7	380	2.6137	58.0283
8	225	2.5369	58.0287
9	395	2.5210	58.0291
10	492	2.4291	58.0295
11	226	2.6405	58.0299
12	166	2.4104	58.0303
13	246	2.4932	58.0307
14	124	2.5718	58.0311
15	94	2.5407	58.0315
16	214	2.4555	58.0319
17	123	2.7261	58.0323
18	227	2.7008	58.0327
19	252	2.4477	58.0331
20	442	2.4727	58.0335
21	53	2.7305	58.0339
22	442	2.6684	58.0343
23	49	2.4203	58.0346
24	381	2.3764	58.0350
25	74	2.5628	58.0354
26	476	2.8250	58.0358
27	56	2.4997	58.0362
28	323	2.4778	58.0366
29	404	2.7399	58.0370
30	181	2.4449	58.0374

Tabla 7.6.2.10. Ejecuciones USA

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.3563	14.6634
2	494	0.3248	14.6999
3	50	0.3375	14.7028
4	494	0.3501	14.7032
5	220	0.3360	14.7036
6	264	0.3220	14.7040
7	380	0.3213	14.7044
8	225	0.3244	14.7048
9	395	0.3194	14.7052
10	492	0.3192	14.7056
11	226	0.3201	14.7060
12	166	0.3194	14.7064
13	246	0.3197	14.7068
14	124	0.3296	14.7072
15	94	0.3245	14.7076
16	214	0.3247	14.7080
17	123	0.3231	14.7084
18	227	0.3231	14.7088
19	252	0.3208	14.7092
20	442	0.3212	14.7096
21	53	0.3200	14.7100
22	442	0.3219	14.7104
23	49	0.3252	14.7108
24	381	0.3223	14.7112
25	74	0.3253	14.7116
26	476	0.3416	14.7120
27	56	0.3212	14.7124
28	323	0.3205	14.7128
29	404	0.3507	14.7132
30	181	0.3200	14.7136

Tabla 7.6.3.1. Ejecuciones Australia

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.2198	6.6373
2	494	0.1233	6.6738
3	50	0.1370	6.6768
4	494	0.1962	6.6772
5	220	0.1324	6.6776
6	264	0.1728	6.6780
7	380	0.1569	6.6784
8	225	0.1635	6.6788
9	395	0.1458	6.6792
10	492	0.2287	6.6796
11	226	0.1448	6.6800
12	166	0.1467	6.6804
13	246	0.1233	6.6807
14	124	0.1387	6.6811
15	94	0.1438	6.6815
16	214	0.1360	6.6819
17	123	0.1734	6.6823
18	227	0.1302	6.6827
19	252	0.1234	6.6831
20	442	0.1359	6.6835
21	53	0.1374	6.6839
22	442	0.1769	6.6843
23	49	0.1392	6.6847
24	381	0.2372	6.6851
25	74	0.1646	6.6855
26	476	0.1325	6.6859
27	56	0.1250	6.6863
28	323	0.1646	6.6867
29	404	0.1525	6.6871
30	181	0.1688	6.6875

Tabla 7.6.3.2. Ejecuciones Canadá

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1463	6.6387
2	494	0.1149	6.6752
3	50	0.1142	6.6782
4	494	0.1139	6.6786
5	220	0.1157	6.6790
6	264	0.1526	6.6794
7	380	0.1175	6.6797
8	225	0.1154	6.6801
9	395	0.1135	6.6805
10	492	0.1133	6.6809
11	226	0.1114	6.6813
12	166	0.1127	6.6817
13	246	0.1284	6.6821
14	124	0.1125	6.6825
15	94	0.1139	6.6829
16	214	0.1127	6.6833
17	123	0.1142	6.6837
18	227	0.1310	6.6841
19	252	0.1135	6.6845
20	442	0.1139	6.6849
21	53	0.1168	6.6853
22	442	0.1118	6.6857
23	49	0.1141	6.6861
24	381	0.1121	6.6865
25	74	0.1134	6.6869
26	476	0.1143	6.6873
27	56	0.1126	6.6877
28	323	0.1185	6.6881
29	404	0.1150	6.6885
30	181	0.1129	6.6889

Tabla 7.6.3.3. Ejecuciones Gran Bretaña

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1531	6.6375
2	494	0.1143	6.6740
3	50	0.1145	6.6769
4	494	0.1165	6.6773
5	220	0.1271	6.6777
6	264	0.1261	6.6781
7	380	0.1149	6.6785
8	225	0.1131	6.6789
9	395	0.1141	6.6793
10	492	0.1147	6.6797
11	226	0.1145	6.6801
12	166	0.1135	6.6805
13	246	0.1134	6.6809
14	124	0.1162	6.6813
15	94	0.1424	6.6817
16	214	0.1139	6.6821
17	123	0.1130	6.6825
18	227	0.1236	6.6829
19	252	0.1141	6.6833
20	442	0.1128	6.6837
21	53	0.1146	6.6841
22	442	0.1134	6.6845
23	49	0.1139	6.6849
24	381	0.1130	6.6853
25	74	0.1129	6.6857
26	476	0.1129	6.6861
27	56	0.1142	6.6865
28	323	0.1128	6.6869
29	404	0.1136	6.6872
30	181	0.1162	6.6876

Tabla 7.6.3.4. Ejecuciones India

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.3716	14.6647
2	494	0.3264	14.7013
3	50	0.3387	14.7042
4	494	0.3271	14.7046
5	220	0.3336	14.7050
6	264	0.3237	14.7054
7	380	0.3249	14.7058
8	225	0.3204	14.7062
9	395	0.3186	14.7066
10	492	0.3202	14.7070
11	226	0.3198	14.7074
12	166	0.3314	14.7078
13	246	0.3221	14.7082
14	124	0.3212	14.7086
15	94	0.3205	14.7090
16	214	0.3216	14.7094
17	123	0.3327	14.7098
18	227	0.3425	14.7102
19	252	0.3198	14.7106
20	442	0.3185	14.7110
21	53	0.3334	14.7114
22	442	0.3234	14.7118
23	49	0.3213	14.7122
24	381	0.3348	14.7125
25	74	0.3225	14.7129
26	476	0.3382	14.7133
27	56	0.3248	14.7137
28	323	0.3225	14.7141
29	404	0.3228	14.7145
30	181	0.3391	14.7149

Tabla 7.6.3.5. Ejecuciones Irlanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1358	6.6394
2	494	0.1148	6.6759
3	50	0.1138	6.6789
4	494	0.1169	6.6793
5	220	0.1150	6.6797
6	264	0.1143	6.6801
7	380	0.1134	6.6805
8	225	0.1349	6.6809
9	395	0.1175	6.6813
10	492	0.1128	6.6817
11	226	0.1125	6.6821
12	166	0.1130	6.6825
13	246	0.1132	6.6828
14	124	0.1179	6.6832
15	94	0.1123	6.6836
16	214	0.1134	6.6840
17	123	0.1131	6.6844
18	227	0.1129	6.6848
19	252	0.1119	6.6852
20	442	0.1256	6.6856
21	53	0.1153	6.6860
22	442	0.1125	6.6864
23	49	0.1124	6.6868
24	381	0.1132	6.6872
25	74	0.1117	6.6876
26	476	0.1119	6.6880
27	56	0.1443	6.6884
28	323	0.1119	6.6888
29	404	0.1112	6.6892
30	181	0.1114	6.6896

Tabla 7.6.3.6. Ejecuciones Jamaica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.7492	25.8989
2	494	0.7186	25.9354
3	50	0.7058	25.9383
4	494	0.7010	25.9387
5	220	0.7052	25.9391
6	264	0.7008	25.9395
7	380	0.7009	25.9399
8	225	0.6990	25.9403
9	395	0.7054	25.9407
10	492	0.6989	25.9411
11	226	0.7024	25.9415
12	166	0.7019	25.9419
13	246	0.7040	25.9423
14	124	0.7014	25.9427
15	94	0.7040	25.9431
16	214	0.7050	25.9435
17	123	0.7044	25.9439
18	227	0.7022	25.9443
19	252	0.7434	25.9447
20	442	0.7060	25.9451
21	53	0.7040	25.9455
22	442	0.7030	25.9459
23	49	0.7114	25.9463
24	381	0.7038	25.9467
25	74	0.7039	25.9471
26	476	0.7025	25.9475
27	56	0.7063	25.9479
28	323	0.7070	25.9482
29	404	0.7017	25.9486
30	181	0.7209	25.9490

Tabla 7.6.3.7. Ejecuciones Nueva Zelanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0493	1.8201
2	494	0.0303	1.8566
3	50	0.0297	1.8596
4	494	0.0302	1.8600
5	220	0.0304	1.8604
6	264	0.0299	1.8608
7	380	0.0305	1.8612
8	225	0.0291	1.8616
9	395	0.0284	1.8620
10	492	0.0280	1.8624
11	226	0.0279	1.8628
12	166	0.0279	1.8632
13	246	0.0280	1.8636
14	124	0.0284	1.8640
15	94	0.0290	1.8644
16	214	0.0281	1.8647
17	123	0.0288	1.8651
18	227	0.0289	1.8655
19	252	0.0279	1.8659
20	442	0.0274	1.8663
21	53	0.0282	1.8667
22	442	0.0285	1.8671
23	49	0.0287	1.8675
24	381	0.0280	1.8679
25	74	0.0281	1.8683
26	476	0.0283	1.8687
27	56	0.0274	1.8691
28	323	0.0286	1.8695
29	404	0.0294	1.8699
30	181	0.0279	1.8703

Tabla 7.6.3.8. Ejecuciones Singapur

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0490	1.8203
2	494	0.0393	1.8568
3	50	0.0347	1.8598
4	494	0.0288	1.8602
5	220	0.0310	1.8606
6	264	0.0294	1.8610
7	380	0.0300	1.8613
8	225	0.0285	1.8617
9	395	0.0285	1.8621
10	492	0.0286	1.8625
11	226	0.0278	1.8629
12	166	0.0284	1.8633
13	246	0.0284	1.8637
14	124	0.0283	1.8641
15	94	0.0281	1.8645
16	214	0.0278	1.8649
17	123	0.0292	1.8653
18	227	0.0289	1.8657
19	252	0.0293	1.8661
20	442	0.0275	1.8665
21	53	0.0278	1.8669
22	442	0.0289	1.8673
23	49	0.0340	1.8677
24	381	0.0302	1.8681
25	74	0.0280	1.8685
26	476	0.0280	1.8689
27	56	0.0283	1.8693
28	323	0.0277	1.8697
29	404	0.0283	1.8701
30	181	0.0279	1.8705

Tabla 7.6.3.9. Ejecuciones Sudáfrica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	2.300987	57.98724174
2	494	2.2466	58.0238
3	50	2.2297	58.0267
4	494	2.2531	58.0271
5	220	2.2616	58.0275
6	264	2.2523	58.0279
7	380	2.2436	58.0283
8	225	2.2500	58.0287
9	395	2.2614	58.0291
10	492	2.2658	58.0295
11	226	2.2584	58.0299
12	166	2.2493	58.0303
13	246	2.2479	58.0307
14	124	2.2369	58.0311
15	94	2.2800	58.0315
16	214	2.2450	58.0319
17	123	2.2427	58.0323
18	227	2.2460	58.0327
19	252	2.2513	58.0331
20	442	2.2497	58.0335
21	53	2.2452	58.0339
22	442	2.2443	58.0343
23	49	2.2606	58.0346
24	381	2.2525	58.0350
25	74	2.2444	58.0354
26	476	2.2453	58.0358
27	56	2.2474	58.0362
28	323	2.2389	58.0366
29	404	2.2541	58.0370
30	181	2.2450	58.0374

Tabla 7.6.3.10. Ejecuciones USA

7.7. Trescientos Bloques

7.7.1. Ejecuciones Macbook

7.7.2. Ejecuciones Windows

7.7.3. Ejecuciones Online

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1274	0.3742
2	494	0.0980	0.4107
3	50	0.0932	0.4136
4	494	0.1163	0.4140
5	220	0.1046	0.4144
6	264	0.1228	0.4148
7	380	0.1932	0.4152
8	225	0.0803	0.4156
9	395	0.0649	0.4160
10	492	0.0647	0.4164
11	226	0.0616	0.4168
12	166	0.0612	0.4172
13	246	0.0686	0.4176
14	124	0.0617	0.4180
15	94	0.0655	0.4184
16	214	0.0834	0.4188
17	123	0.0621	0.4192
18	227	0.0675	0.4196
19	252	0.0622	0.4200
20	442	0.0605	0.4204
21	53	0.0749	0.4208
22	442	0.0889	0.4212
23	49	0.0759	0.4216
24	381	0.0638	0.4220
25	74	0.0612	0.4224
26	476	0.0969	0.4228
27	56	0.0573	0.4232
28	323	0.0642	0.4236
29	404	0.0944	0.4239
30	181	0.0595	0.4243

Tabla 7.7.1.1. Ejecuciones Australia

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1087	0.2865
2	494	0.1084	0.3230
3	50	0.0814	0.3260
4	494	0.1212	0.3264
5	220	0.1000	0.3268
6	264	0.1079	0.3272
7	380	0.0954	0.3276
8	225	0.1360	0.3279
9	395	0.0565	0.3283
10	492	0.0612	0.3287
11	226	0.0674	0.3291
12	166	0.0597	0.3295
13	246	0.0609	0.3299
14	124	0.0826	0.3303
15	94	0.0979	0.3307
16	214	0.0715	0.3311
17	123	0.1044	0.3315
18	227	0.0992	0.3319
19	252	0.0627	0.3323
20	442	0.0571	0.3327
21	53	0.0533	0.3331
22	442	0.0574	0.3335
23	49	0.0592	0.3339
24	381	0.0592	0.3343
25	74	0.0552	0.3347
26	476	0.0570	0.3351
27	56	0.0607	0.3355
28	323	0.0574	0.3359
29	404	0.0575	0.3363
30	181	0.0574	0.3367

Tabla 7.7.1.2. Ejecuciones Canadá

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1120	0.2879
2	494	0.0830	0.3244
3	50	0.1764	0.3273
4	494	0.1281	0.3277
5	220	0.0824	0.3281
6	264	0.0795	0.3285
7	380	0.1894	0.3289
8	225	0.0648	0.3293
9	395	0.0521	0.3297
10	492	0.0784	0.3301
11	226	0.0655	0.3305
12	166	0.0614	0.3309
13	246	0.0576	0.3313
14	124	0.0531	0.3317
15	94	0.0866	0.3321
16	214	0.0877	0.3325
17	123	0.0706	0.3329
18	227	0.0819	0.3333
19	252	0.0695	0.3337
20	442	0.0588	0.3341
21	53	0.0520	0.3345
22	442	0.0538	0.3349
23	49	0.0537	0.3353
24	381	0.0506	0.3357
25	74	0.0575	0.3361
26	476	0.0534	0.3365
27	56	0.0804	0.3369
28	323	0.0441	0.3373
29	404	0.0529	0.3377
30	181	0.0500	0.3381

Tabla 7.7.1.3. Ejecuciones Gran Bretaña

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1287	0.2866
2	494	0.1031	0.3232
3	50	0.0735	0.3261
4	494	0.0895	0.3265
5	220	0.0828	0.3269
6	264	0.0931	0.3273
7	380	0.1288	0.3277
8	225	0.0670	0.3281
9	395	0.0713	0.3285
10	492	0.0581	0.3289
11	226	0.0502	0.3293
12	166	0.0522	0.3297
13	246	0.0576	0.3301
14	124	0.0498	0.3305
15	94	0.0710	0.3309
16	214	0.0507	0.3313
17	123	0.0542	0.3317
18	227	0.0521	0.3321
19	252	0.0471	0.3325
20	442	0.0508	0.3329
21	53	0.0523	0.3333
22	442	0.0510	0.3337
23	49	0.0475	0.3340
24	381	0.0544	0.3344
25	74	0.0504	0.3348
26	476	0.0470	0.3352
27	56	0.0476	0.3356
28	323	0.0509	0.3360
29	404	0.0485	0.3364
30	181	0.0549	0.3368

Tabla 7.7.1.4. Ejecuciones India

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1202	0.3755
2	494	0.0984	0.4121
3	50	0.0998	0.4150
4	494	0.1845	0.4154
5	220	0.1035	0.4158
6	264	0.1118	0.4162
7	380	0.1389	0.4166
8	225	0.0739	0.4170
9	395	0.0673	0.4174
10	492	0.0603	0.4178
11	226	0.0680	0.4182
12	166	0.0531	0.4186
13	246	0.0524	0.4190
14	124	0.0576	0.4194
15	94	0.0594	0.4198
16	214	0.0829	0.4202
17	123	0.0587	0.4206
18	227	0.0888	0.4210
19	252	0.0570	0.4214
20	442	0.0859	0.4217
21	53	0.0663	0.4221
22	442	0.0706	0.4225
23	49	0.0618	0.4229
24	381	0.0558	0.4233
25	74	0.0560	0.4237
26	476	0.0556	0.4241
27	56	0.0596	0.4245
28	323	0.0565	0.4249
29	404	0.0742	0.4253
30	181	0.0599	0.4257

Tabla 7.7.1.5. Ejecuciones Irlanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1175	0.2886
2	494	0.0776	0.3251
3	50	0.1023	0.3281
4	494	0.1137	0.3285
5	220	0.0791	0.3289
6	264	0.0833	0.3293
7	380	0.0867	0.3297
8	225	0.0804	0.3300
9	395	0.0792	0.3304
10	492	0.0722	0.3308
11	226	0.0901	0.3312
12	166	0.1178	0.3316
13	246	0.1177	0.3320
14	124	0.0871	0.3324
15	94	0.0744	0.3328
16	214	0.0707	0.3332
17	123	0.0508	0.3336
18	227	0.0487	0.3340
19	252	0.0519	0.3344
20	442	0.0768	0.3348
21	53	0.0487	0.3352
22	442	0.0444	0.3356
23	49	0.0559	0.3360
24	381	0.0565	0.3364
25	74	0.0494	0.3368
26	476	0.0545	0.3372
27	56	0.0612	0.3376
28	323	0.0493	0.3380
29	404	0.0562	0.3384
30	181	0.0451	0.3388

Tabla 7.7.1.6. Ejecuciones Jamaica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.2038	0.4959
2	494	0.1906	0.5325
3	50	0.1887	0.5354
4	494	0.2070	0.5358
5	220	0.0971	0.5362
6	264	0.0791	0.5366
7	380	0.0793	0.5370
8	225	0.1006	0.5374
9	395	0.0822	0.5378
10	492	0.0800	0.5382
11	226	0.0784	0.5386
12	166	0.0841	0.5390
13	246	0.0841	0.5394
14	124	0.0766	0.5398
15	94	0.0819	0.5402
16	214	0.0772	0.5406
17	123	0.0768	0.5410
18	227	0.0829	0.5413
19	252	0.1061	0.5417
20	442	0.0806	0.5421
21	53	0.0947	0.5425
22	442	0.1003	0.5429
23	49	0.0995	0.5433
24	381	0.0779	0.5437
25	74	0.0986	0.5441
26	476	0.0926	0.5445
27	56	0.0772	0.5449
28	323	0.0801	0.5453
29	404	0.0776	0.5457
30	181	0.0903	0.5461

Tabla 7.7.1.7. Ejecuciones Nueva Zelanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0890	0.2324
2	494	0.1293	0.2689
3	50	0.0657	0.2718
4	494	0.1042	0.2722
5	220	0.0570	0.2726
6	264	0.0575	0.2730
7	380	0.0602	0.2734
8	225	0.0707	0.2738
9	395	0.0700	0.2742
10	492	0.0662	0.2746
11	226	0.1117	0.2750
12	166	0.0623	0.2754
13	246	0.0788	0.2758
14	124	0.0464	0.2762
15	94	0.0424	0.2766
16	214	0.0465	0.2770
17	123	0.0491	0.2774
18	227	0.0590	0.2778
19	252	0.0810	0.2782
20	442	0.0742	0.2786
21	53	0.0571	0.2790
22	442	0.0694	0.2794
23	49	0.0565	0.2798
24	381	0.0506	0.2802
25	74	0.0497	0.2806
26	476	0.0744	0.2810
27	56	0.0428	0.2814
28	323	0.0511	0.2818
29	404	0.0453	0.2822
30	181	0.0442	0.2826

Tabla 7.7.1.8. Ejecuciones Singapur

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1490	0.2325
2	494	0.1106	0.2691
3	50	0.1298	0.2720
4	494	0.0483	0.2724
5	220	0.0389	0.2728
6	264	0.0803	0.2732
7	380	0.0693	0.2736
8	225	0.0618	0.2740
9	395	0.0517	0.2744
10	492	0.0416	0.2748
11	226	0.0505	0.2752
12	166	0.0518	0.2756
13	246	0.0469	0.2760
14	124	0.0548	0.2764
15	94	0.0563	0.2768
16	214	0.0621	0.2772
17	123	0.0502	0.2776
18	227	0.0499	0.2780
19	252	0.0510	0.2784
20	442	0.0500	0.2788
21	53	0.0483	0.2792
22	442	0.0511	0.2796
23	49	0.0493	0.2799
24	381	0.0566	0.2803
25	74	0.0497	0.2807
26	476	0.0513	0.2811
27	56	0.0515	0.2815
28	323	0.0487	0.2819
29	404	0.0500	0.2823
30	181	0.0514	0.2827

Tabla 7.7.1.9. Ejecuciones Sudáfrica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.2806	0.8309
2	494	0.2546	0.8674
3	50	0.1574	0.8704
4	494	0.2191	0.8708
5	220	0.1187	0.8712
6	264	0.1214	0.8716
7	380	0.1174	0.8720
8	225	0.1199	0.8724
9	395	0.1215	0.8728
10	492	0.1440	0.8732
11	226	0.1172	0.8736
12	166	0.1221	0.8739
13	246	0.1163	0.8743
14	124	0.1177	0.8747
15	94	0.1189	0.8751
16	214	0.1132	0.8755
17	123	0.1150	0.8759
18	227	0.1202	0.8763
19	252	0.1732	0.8767
20	442	0.1736	0.8771
21	53	0.1207	0.8775
22	442	0.1216	0.8779
23	49	0.1155	0.8783
24	381	0.1221	0.8787
25	74	0.1347	0.8791
26	476	0.1214	0.8795
27	56	0.1285	0.8799
28	323	0.1275	0.8803
29	404	0.1333	0.8807
30	181	0.1170	0.8811

Tabla 7.7.1.10. Ejecuciones USA

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0532	0.3742
2	494	0.0588	0.4107
3	50	0.0435	0.4136
4	494	0.0458	0.4140
5	220	0.0523	0.4144
6	264	0.0546	0.4148
7	380	0.0364	0.4152
8	225	0.0327	0.4156
9	395	0.0386	0.4160
10	492	0.0365	0.4164
11	226	0.0322	0.4168
12	166	0.0271	0.4172
13	246	0.0293	0.4176
14	124	0.0287	0.4180
15	94	0.0348	0.4184
16	214	0.0294	0.4188
17	123	0.0286	0.4192
18	227	0.0303	0.4196
19	252	0.0289	0.4200
20	442	0.0479	0.4204
21	53	0.0489	0.4208
22	442	0.0368	0.4212
23	49	0.0359	0.4216
24	381	0.0336	0.4220
25	74	0.0319	0.4224
26	476	0.0436	0.4228
27	56	0.0315	0.4232
28	323	0.0361	0.4236
29	404	0.0283	0.4239
30	181	0.0550	0.4243

Tabla 7.7.2.1. Ejecuciones Australia

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0670	0.2865
2	494	0.0493	0.3230
3	50	0.0380	0.3260
4	494	0.0375	0.3264
5	220	0.0416	0.3268
6	264	0.0362	0.3272
7	380	0.0371	0.3276
8	225	0.0354	0.3279
9	395	0.0400	0.3283
10	492	0.0367	0.3287
11	226	0.0288	0.3291
12	166	0.0298	0.3295
13	246	0.0265	0.3299
14	124	0.0376	0.3303
15	94	0.0278	0.3307
16	214	0.0306	0.3311
17	123	0.0299	0.3315
18	227	0.0311	0.3319
19	252	0.0348	0.3323
20	442	0.0337	0.3327
21	53	0.0478	0.3331
22	442	0.0350	0.3335
23	49	0.0401	0.3339
24	381	0.0360	0.3343
25	74	0.0295	0.3347
26	476	0.0335	0.3351
27	56	0.0292	0.3355
28	323	0.0351	0.3359
29	404	0.0261	0.3363
30	181	0.0254	0.3367

Tabla 7.7.2.2. Ejecuciones Canadá

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0673	0.2879
2	494	0.0391	0.3244
3	50	0.0463	0.3273
4	494	0.0622	0.3277
5	220	0.0349	0.3281
6	264	0.0551	0.3285
7	380	0.0392	0.3289
8	225	0.0256	0.3293
9	395	0.0265	0.3297
10	492	0.0284	0.3301
11	226	0.0309	0.3305
12	166	0.0333	0.3309
13	246	0.0312	0.3313
14	124	0.0290	0.3317
15	94	0.0271	0.3321
16	214	0.0272	0.3325
17	123	0.0282	0.3329
18	227	0.0342	0.3333
19	252	0.0365	0.3337
20	442	0.0306	0.3341
21	53	0.0273	0.3345
22	442	0.0306	0.3349
23	49	0.0232	0.3353
24	381	0.0329	0.3357
25	74	0.0319	0.3361
26	476	0.0328	0.3365
27	56	0.0383	0.3369
28	323	0.0343	0.3373
29	404	0.0284	0.3377
30	181	0.0315	0.3381

Tabla 7.7.2.3. Ejecuciones Gran Bretaña

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0485	0.2866
2	494	0.0448	0.3232
3	50	0.0520	0.3261
4	494	0.0369	0.3265
5	220	0.0303	0.3269
6	264	0.0504	0.3273
7	380	0.0371	0.3277
8	225	0.0249	0.3281
9	395	0.0263	0.3285
10	492	0.0259	0.3289
11	226	0.0272	0.3293
12	166	0.0250	0.3297
13	246	0.0306	0.3301
14	124	0.0295	0.3305
15	94	0.0356	0.3309
16	214	0.0300	0.3313
17	123	0.0235	0.3317
18	227	0.0275	0.3321
19	252	0.0302	0.3325
20	442	0.0245	0.3329
21	53	0.0289	0.3333
22	442	0.0275	0.3337
23	49	0.0236	0.3340
24	381	0.1033	0.3344
25	74	0.0274	0.3348
26	476	0.0334	0.3352
27	56	0.0263	0.3356
28	323	0.0264	0.3360
29	404	0.0286	0.3364
30	181	0.0283	0.3368

Tabla 7.7.2.4. Ejecuciones India

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0711	0.3755
2	494	0.0658	0.4121
3	50	0.0474	0.4150
4	494	0.0460	0.4154
5	220	0.0529	0.4158
6	264	0.0440	0.4162
7	380	0.0280	0.4166
8	225	0.0291	0.4170
9	395	0.0367	0.4174
10	492	0.0327	0.4178
11	226	0.0328	0.4182
12	166	0.0314	0.4186
13	246	0.0294	0.4190
14	124	0.0342	0.4194
15	94	0.0301	0.4198
16	214	0.0301	0.4202
17	123	0.0309	0.4206
18	227	0.0524	0.4210
19	252	0.0380	0.4214
20	442	0.0378	0.4217
21	53	0.0349	0.4221
22	442	0.0347	0.4225
23	49	0.0313	0.4229
24	381	0.0298	0.4233
25	74	0.0292	0.4237
26	476	0.0289	0.4241
27	56	0.0332	0.4245
28	323	0.0362	0.4249
29	404	0.0326	0.4253
30	181	0.0316	0.4257

Tabla 7.7.2.5. Ejecuciones Irlanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0462	0.2886
2	494	0.0428	0.3251
3	50	0.0458	0.3281
4	494	0.0401	0.3285
5	220	0.0304	0.3289
6	264	0.0376	0.3293
7	380	0.0350	0.3297
8	225	0.0386	0.3300
9	395	0.0355	0.3304
10	492	0.0330	0.3308
11	226	0.0303	0.3312
12	166	0.0286	0.3316
13	246	0.0269	0.3320
14	124	0.0332	0.3324
15	94	0.0341	0.3328
16	214	0.0292	0.3332
17	123	0.0258	0.3336
18	227	0.0249	0.3340
19	252	0.0281	0.3344
20	442	0.0238	0.3348
21	53	0.0275	0.3352
22	442	0.0239	0.3356
23	49	0.0344	0.3360
24	381	0.0314	0.3364
25	74	0.0282	0.3368
26	476	0.0358	0.3372
27	56	0.0291	0.3376
28	323	0.0314	0.3380
29	404	0.0264	0.3384
30	181	0.0276	0.3388

Tabla 7.7.2.6. Ejecuciones Jamaica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0790	0.4959
2	494	0.0615	0.5325
3	50	0.0647	0.5354
4	494	0.0544	0.5358
5	220	0.0431	0.5362
6	264	0.0361	0.5366
7	380	0.0480	0.5370
8	225	0.0415	0.5374
9	395	0.0388	0.5378
10	492	0.0387	0.5382
11	226	0.0407	0.5386
12	166	0.0677	0.5390
13	246	0.0386	0.5394
14	124	0.0365	0.5398
15	94	0.0445	0.5402
16	214	0.0368	0.5406
17	123	0.0468	0.5410
18	227	0.0478	0.5413
19	252	0.0362	0.5417
20	442	0.0378	0.5421
21	53	0.0335	0.5425
22	442	0.0369	0.5429
23	49	0.0402	0.5433
24	381	0.0356	0.5437
25	74	0.0398	0.5441
26	476	0.0600	0.5445
27	56	0.0540	0.5449
28	323	0.0454	0.5453
29	404	0.0443	0.5457
30	181	0.0415	0.5461

Tabla 7.7.2.7. Ejecuciones Nueva Zelanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0550	0.2324
2	494	0.0419	0.2689
3	50	0.0289	0.2718
4	494	0.0301	0.2722
5	220	0.0223	0.2726
6	264	0.0250	0.2730
7	380	0.0290	0.2734
8	225	0.0244	0.2738
9	395	0.0264	0.2742
10	492	0.0210	0.2746
11	226	0.0238	0.2750
12	166	0.0322	0.2754
13	246	0.0269	0.2758
14	124	0.0328	0.2762
15	94	0.0285	0.2766
16	214	0.0308	0.2770
17	123	0.0284	0.2774
18	227	0.0284	0.2778
19	252	0.0285	0.2782
20	442	0.0310	0.2786
21	53	0.0288	0.2790
22	442	0.0215	0.2794
23	49	0.0238	0.2798
24	381	0.0233	0.2802
25	74	0.0245	0.2806
26	476	0.0267	0.2810
27	56	0.0294	0.2814
28	323	0.0288	0.2818
29	404	0.0288	0.2822
30	181	0.0227	0.2826

Tabla 7.7.2.8. Ejecuciones Singapur

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0414	0.2325
2	494	0.0355	0.2691
3	50	0.0400	0.2720
4	494	0.0231	0.2724
5	220	0.0203	0.2728
6	264	0.0245	0.2732
7	380	0.0219	0.2736
8	225	0.0240	0.2740
9	395	0.0258	0.2744
10	492	0.0241	0.2748
11	226	0.0237	0.2752
12	166	0.0231	0.2756
13	246	0.0350	0.2760
14	124	0.0202	0.2764
15	94	0.0207	0.2768
16	214	0.0201	0.2772
17	123	0.0225	0.2776
18	227	0.0236	0.2780
19	252	0.0213	0.2784
20	442	0.0246	0.2788
21	53	0.0197	0.2792
22	442	0.0261	0.2796
23	49	0.0170	0.2799
24	381	0.0210	0.2803
25	74	0.0231	0.2807
26	476	0.0274	0.2811
27	56	0.0254	0.2815
28	323	0.0230	0.2819
29	404	0.0218	0.2823
30	181	0.0317	0.2827

Tabla 7.7.2.9. Ejecuciones Sudáfrica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1122	0.8309
2	494	0.0778	0.8674
3	50	0.0867	0.8704
4	494	0.0731	0.8708
5	220	0.0679	0.8712
6	264	0.0724	0.8716
7	380	0.0640	0.8720
8	225	0.0619	0.8724
9	395	0.0619	0.8728
10	492	0.0677	0.8732
11	226	0.0703	0.8736
12	166	0.2104	0.8739
13	246	0.0767	0.8743
14	124	0.0820	0.8747
15	94	0.0641	0.8751
16	214	0.0699	0.8755
17	123	0.0693	0.8759
18	227	0.0659	0.8763
19	252	0.0633	0.8767
20	442	0.0757	0.8771
21	53	0.0657	0.8775
22	442	0.0729	0.8779
23	49	0.0686	0.8783
24	381	0.0859	0.8787
25	74	0.0673	0.8791
26	476	0.0611	0.8795
27	56	0.0610	0.8799
28	323	0.0803	0.8803
29	404	0.1230	0.8807
30	181	0.0760	0.8811

Tabla 7.7.2.10. Ejecuciones USA

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1008	0.3742
2	494	0.0343	0.4107
3	50	0.0346	0.4136
4	494	0.0338	0.4140
5	220	0.0357	0.4144
6	264	0.0340	0.4148
7	380	0.0353	0.4152
8	225	0.0357	0.4156
9	395	0.0324	0.4160
10	492	0.0321	0.4164
11	226	0.0321	0.4168
12	166	0.0324	0.4172
13	246	0.0317	0.4176
14	124	0.0323	0.4180
15	94	0.0320	0.4184
16	214	0.0314	0.4188
17	123	0.0321	0.4192
18	227	0.0318	0.4196
19	252	0.0321	0.4200
20	442	0.0321	0.4204
21	53	0.0388	0.4208
22	442	0.0394	0.4212
23	49	0.0325	0.4216
24	381	0.0317	0.4220
25	74	0.0315	0.4224
26	476	0.0327	0.4228
27	56	0.0320	0.4232
28	323	0.0317	0.4236
29	404	0.0335	0.4239
30	181	0.0312	0.4243

Tabla 7.7.3.1. Ejecuciones Australia

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1064	0.2865
2	494	0.0404	0.3230
3	50	0.0394	0.3260
4	494	0.0311	0.3264
5	220	0.0292	0.3268
6	264	0.0269	0.3272
7	380	0.0268	0.3276
8	225	0.0273	0.3279
9	395	0.0279	0.3283
10	492	0.0274	0.3287
11	226	0.0274	0.3291
12	166	0.0267	0.3295
13	246	0.0262	0.3299
14	124	0.0268	0.3303
15	94	0.0275	0.3307
16	214	0.0261	0.3311
17	123	0.0275	0.3315
18	227	0.0271	0.3319
19	252	0.0271	0.3323
20	442	0.0269	0.3327
21	53	0.0291	0.3331
22	442	0.0270	0.3335
23	49	0.0268	0.3339
24	381	0.0265	0.3343
25	74	0.0286	0.3347
26	476	0.0277	0.3351
27	56	0.0276	0.3355
28	323	0.0266	0.3359
29	404	0.0264	0.3363
30	181	0.0268	0.3367

Tabla 7.7.3.2. Ejecuciones Canadá

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0661	0.2879
2	494	0.0370	0.3244
3	50	0.0280	0.3273
4	494	0.0277	0.3277
5	220	0.0291	0.3281
6	264	0.0279	0.3285
7	380	0.0257	0.3289
8	225	0.0252	0.3293
9	395	0.0262	0.3297
10	492	0.0266	0.3301
11	226	0.0264	0.3305
12	166	0.0275	0.3309
13	246	0.0281	0.3313
14	124	0.0261	0.3317
15	94	0.0294	0.3321
16	214	0.0275	0.3325
17	123	0.0258	0.3329
18	227	0.0268	0.3333
19	252	0.0260	0.3337
20	442	0.0259	0.3341
21	53	0.0270	0.3345
22	442	0.0588	0.3349
23	49	0.0272	0.3353
24	381	0.0262	0.3357
25	74	0.0261	0.3361
26	476	0.0258	0.3365
27	56	0.0305	0.3369
28	323	0.0287	0.3373
29	404	0.0262	0.3377
30	181	0.0258	0.3381

Tabla 7.7.3.3. Ejecuciones Gran Bretaña

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0697	0.2866
2	494	0.0283	0.3232
3	50	0.0277	0.3261
4	494	0.0273	0.3265
5	220	0.0289	0.3269
6	264	0.0276	0.3273
7	380	0.0268	0.3277
8	225	0.0263	0.3281
9	395	0.0278	0.3285
10	492	0.0258	0.3289
11	226	0.0259	0.3293
12	166	0.0276	0.3297
13	246	0.0274	0.3301
14	124	0.0268	0.3305
15	94	0.0262	0.3309
16	214	0.0262	0.3313
17	123	0.0260	0.3317
18	227	0.0262	0.3321
19	252	0.0277	0.3325
20	442	0.0282	0.3329
21	53	0.0265	0.3333
22	442	0.0265	0.3337
23	49	0.0261	0.3340
24	381	0.0265	0.3344
25	74	0.0261	0.3348
26	476	0.0268	0.3352
27	56	0.0262	0.3356
28	323	0.0272	0.3360
29	404	0.0258	0.3364
30	181	0.0258	0.3368

Tabla 7.7.3.4. Ejecuciones India

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0615	0.3755
2	494	0.0328	0.4121
3	50	0.0359	0.4150
4	494	0.0323	0.4154
5	220	0.0338	0.4158
6	264	0.0363	0.4162
7	380	0.0392	0.4166
8	225	0.0354	0.4170
9	395	0.0380	0.4174
10	492	0.0321	0.4178
11	226	0.0318	0.4182
12	166	0.0339	0.4186
13	246	0.0314	0.4190
14	124	0.0313	0.4194
15	94	0.0373	0.4198
16	214	0.0341	0.4202
17	123	0.0307	0.4206
18	227	0.0309	0.4210
19	252	0.0317	0.4214
20	442	0.0314	0.4217
21	53	0.0316	0.4221
22	442	0.0322	0.4225
23	49	0.0309	0.4229
24	381	0.0310	0.4233
25	74	0.0319	0.4237
26	476	0.0311	0.4241
27	56	0.0303	0.4245
28	323	0.0303	0.4249
29	404	0.0303	0.4253
30	181	0.0304	0.4257

Tabla 7.7.3.5. Ejecuciones Irlanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0677	0.2886
2	494	0.0276	0.3251
3	50	0.0288	0.3281
4	494	0.0279	0.3285
5	220	0.0387	0.3289
6	264	0.0378	0.3293
7	380	0.0307	0.3297
8	225	0.0270	0.3300
9	395	0.0266	0.3304
10	492	0.0300	0.3308
11	226	0.0279	0.3312
12	166	0.0287	0.3316
13	246	0.0272	0.3320
14	124	0.0285	0.3324
15	94	0.0267	0.3328
16	214	0.0258	0.3332
17	123	0.0269	0.3336
18	227	0.0271	0.3340
19	252	0.0258	0.3344
20	442	0.0259	0.3348
21	53	0.0262	0.3352
22	442	0.0270	0.3356
23	49	0.0265	0.3360
24	381	0.0258	0.3364
25	74	0.0259	0.3368
26	476	0.0262	0.3372
27	56	0.0274	0.3376
28	323	0.0269	0.3380
29	404	0.0266	0.3384
30	181	0.0261	0.3388

Tabla 7.7.3.6. Ejecuciones Jamaica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0779	0.4959
2	494	0.0611	0.5325
3	50	0.0474	0.5354
4	494	0.0391	0.5358
5	220	0.0400	0.5362
6	264	0.0394	0.5366
7	380	0.0387	0.5370
8	225	0.0384	0.5374
9	395	0.0415	0.5378
10	492	0.0404	0.5382
11	226	0.0384	0.5386
12	166	0.0385	0.5390
13	246	0.0385	0.5394
14	124	0.0388	0.5398
15	94	0.0379	0.5402
16	214	0.0389	0.5406
17	123	0.0383	0.5410
18	227	0.0385	0.5413
19	252	0.0393	0.5417
20	442	0.0782	0.5421
21	53	0.0388	0.5425
22	442	0.0397	0.5429
23	49	0.0395	0.5433
24	381	0.0382	0.5437
25	74	0.0384	0.5441
26	476	0.0390	0.5445
27	56	0.0388	0.5449
28	323	0.0395	0.5453
29	404	0.0378	0.5457
30	181	0.0389	0.5461

Tabla 7.7.3.7. Ejecuciones Nueva Zelanda

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0416	0.2324
2	494	0.0249	0.2689
3	50	0.0249	0.2718
4	494	0.0253	0.2722
5	220	0.0251	0.2726
6	264	0.0240	0.2730
7	380	0.0229	0.2734
8	225	0.0232	0.2738
9	395	0.0234	0.2742
10	492	0.0239	0.2746
11	226	0.0241	0.2750
12	166	0.0236	0.2754
13	246	0.0228	0.2758
14	124	0.0223	0.2762
15	94	0.0229	0.2766
16	214	0.0242	0.2770
17	123	0.0239	0.2774
18	227	0.0245	0.2778
19	252	0.0231	0.2782
20	442	0.0252	0.2786
21	53	0.0232	0.2790
22	442	0.0234	0.2794
23	49	0.0230	0.2798
24	381	0.0252	0.2802
25	74	0.0230	0.2806
26	476	0.0230	0.2810
27	56	0.0234	0.2814
28	323	0.0227	0.2818
29	404	0.0225	0.2822
30	181	0.0236	0.2826

Tabla 7.7.3.8. Ejecuciones Singapur

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.0579	0.2325
2	494	0.0245	0.2691
3	50	0.0242	0.2720
4	494	0.0233	0.2724
5	220	0.0248	0.2728
6	264	0.0239	0.2732
7	380	0.0230	0.2736
8	225	0.0224	0.2740
9	395	0.0236	0.2744
10	492	0.0230	0.2748
11	226	0.0233	0.2752
12	166	0.0227	0.2756
13	246	0.0228	0.2760
14	124	0.0244	0.2764
15	94	0.0238	0.2768
16	214	0.0235	0.2772
17	123	0.0225	0.2776
18	227	0.0228	0.2780
19	252	0.0227	0.2784
20	442	0.0238	0.2788
21	53	0.0239	0.2792
22	442	0.0229	0.2796
23	49	0.0231	0.2799
24	381	0.0232	0.2803
25	74	0.0233	0.2807
26	476	0.0232	0.2811
27	56	0.0246	0.2815
28	323	0.0230	0.2819
29	404	0.0240	0.2823
30	181	0.0230	0.2827

Tabla 7.7.3.9. Ejecuciones Sudáfrica

No.	semilla	TiempoEjecucion(s)	UsoMemoria(MB)
1	280	0.1158	0.8309
2	494	0.0684	0.8674
3	50	0.0778	0.8704
4	494	0.0683	0.8708
5	220	0.0694	0.8712
6	264	0.0689	0.8716
7	380	0.0727	0.8720
8	225	0.0664	0.8724
9	395	0.0674	0.8728
10	492	0.0660	0.8732
11	226	0.0650	0.8736
12	166	0.0700	0.8739
13	246	0.0769	0.8743
14	124	0.0800	0.8747
15	94	0.0660	0.8751
16	214	0.0679	0.8755
17	123	0.0660	0.8759
18	227	0.0637	0.8763
19	252	0.0656	0.8767
20	442	0.0762	0.8771
21	53	0.0686	0.8775
22	442	0.0661	0.8779
23	49	0.0646	0.8783
24	381	0.0692	0.8787
25	74	0.0685	0.8791
26	476	0.0648	0.8795
27	56	0.0684	0.8799
28	323	0.0660	0.8803
29	404	0.0666	0.8807
30	181	0.0732	0.8811

Tabla 7.7.3.10. Ejecuciones USA

Linea de investigación derivada de la tesis

7.8. Artículo Enviado

Publicación derivada de la tesis.
Este artículo aún está en proceso de revisión.

cryptography

MDPI

Article

The HAJ: Initial Steps of a Novel Cryptographic Algorithm

Alarcón-Narváez Daniel [1,*], Jacques-García FA [2,†], SL Canchola-Magdaleno [3,†], RC Chaparro [3,†], LA Lizama-Pérez [2,†] and AR Pinto [3,†]

[1] Computer Science, Autonomous University of Queretaro(UAQ), Av. de las Ciências, Querétaro, 76230, Querétaro, México;
[2] Electronic, Universidad Técnica Federico Santa María, Av. Vicuña Mackenna, , 8940897, San Joaquín, Chile; jacques@uaq.edu.mx; luis.lizamap@usm.cl
[3] Computer Science, ederal University of Santa Catarina, Florianópolis, 88040-370, Brazil; isandra.canchola@uaq.mx; rchapa@uaq.mx; a.r.pinto@ufsc.br
* Correspondence: daniel.alarcon@uaq.mx
† These authors contributed equally to this work.

Abstract: We present the Hill-Alarcón-Jacques (HAJ) algorithm, a novel symmetric cipher that integrates the classical Hill cipher with the Gauss-Jacques method for modular inverse computation. The HAJ algorithm introduces three main enhancements: (i) multi-round encryption with dynamic subkey generation through circular shifts, (ii) efficient modular inversion using the Gauss-Jacques procedure, and (iii) flexible block sizes that enable deployment in both high-performance and resource-constrained environments. We discuss the potential security properties of HAJ and outline future work for formal cryptanalysis and statistical testing. Experimental results on multiple platforms demonstrate stable memory usage and competitive execution times across a variety of plaintext sizes. These findings demonstrate that HAJ exhibits strong cryptographic properties while maintaining practical efficiency, rendering it a viable candidate for modern symmetric encryption applications.

Keywords: Gauss-Jacques method; hill cipher; modular inverse calculation; security; symmetric cryptography

Received:
Revised:
Accepted:
Published:

Citation: Lastname, F.; Lastname, F.; Lastname, F. The HAJ: Initial Steps of a Novel Cryptographic Algorithm. *Cryptography* **2025**, *1*, 0. https://doi.org/

1. Introduction

Cryptography is the study and practice of techniques for securing communication in the presence of unauthorized third parties. In modern cryptography, there are two main types of encryption: symmetric cryptography and asymmetric cryptography [1].

Symmetric cryptography, also known as secret-key cryptography, uses a shared key between the sender and receiver to encrypt and decrypt the message. The security of this method depends on the shared key being known only to the sender and receiver. Examples of symmetric cryptography algorithms are DES (Data Encryption Standard), AES (Advanced Encryption Standard), and the Hill-Cipher symmetric algorithm [2].

On the other hand, asymmetric cryptography, also known as public-key cryptography, uses a pair of keys: a public key and a private key. The public key is shared with anyone who wants to send a message to the owner of the private key. The owner uses their private key to decrypt the message. Examples of asymmetric cryptography algorithms are RSA (Rivest-Shamir-Adleman), Diffie-Hellman, and ECC (Elliptic Curve Cryptography) [2].

Both types of encryption have their advantages and disadvantages and are used in different situations. The choice of the appropriate encryption method depends on the nature of the information to be encrypted and the security needs of the system.

In recent years, the design of lightweight cryptographic algorithms has gained considerable attention, especially for Internet of Things (IoT) and resource-constrained environments. Comprehensive evaluations of symmetric ciphers for IoT show that algorithmic efficiency and robustness are equally critical, as trade-offs between security and performance determine practical deployment [3,4]. These works highlight the need for symmetric ciphers that combine mathematical rigor with computational feasibility.

One of the main challenges in Hill-type ciphers is their vulnerability to linear cryptanalysis and known-plaintext attacks. Several modern studies have focused on analyzing and breaking variants of the Hill cipher. For example, Wen et al. [5] demonstrated that Hill-based schemes combined with chaotic systems can still be attacked using statistical analysis. Similarly, Teseleanu [6] analyzed a Hill-plus-chaos variant and showed weaknesses when subjected to standard cryptanalytic techniques. These findings suggest that any novel Hill variant must be thoroughly evaluated against both classical and modern attacks.

At the same time, mathematical advances in modular inverse computation remain relevant for cryptographic constructions. Bufalo [7] provided a detailed analysis of modular inverse properties and highlighted their applications in cryptography. Recent developments also include efficient methods for inversion modulo powers of integers, such as those presented by Xu et al. [8], which illustrate the ongoing interest in optimizing matrix inversion techniques for cryptographic use cases.

Finally, new designs continue to explore robust Hill-based algorithms. Rrghout et al. [9] proposed an image encryption algorithm based on a novel Hill variant, showing the community's continuous exploration of this family of ciphers. However, the existing literature indicates that security analysis—such as resistance to differential and linear attacks, as well as statistical tests of randomness—is often missing or insufficient, which motivates the present work.

In this context, we introduce a new symmetric cryptographic algorithm called Hill-Alarcon-Jacques (HAJ), which, to the best of our knowledge, represents the first attempt to combine the Hill cipher with the Gauss-Jacques method for modular inverse calculation. This novel integration has not been explored in prior works, making HAJ a unique and previously unreported contribution to the field of symmetric cryptography.

The remainder of this paper is organized as follows. Section 2 describes the computational environment. Section 3 introduces the theoretical background, including the classical Hill cipher and the Gauss-Jacques modular inverse method. Section 4 details the methodology of the proposed Hill-Alarcón-Jacques (HAJ) algorithm. Section 5 presents the algorithm pseudocode. Section 6 reports both performance results. Section 7 concludes the paper and discusses future research directions.

2. Materials

1. Equipment used:
    - Desktop HP computer with the following specifications. Processor: 11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz 2.90 Ghz RAM: 8 GB Operating System: Windows 11 Home
2. Equipment used:
    - Desktop Macbook Air computer with the following specifications: Processor: 11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz 2.90 Ghz, RAM: 8 GB, Operating System: macOs Monterrey
3. Software used:

- MATLAB (Online Trial Version): The online version of MATLAB provided by MathWorks was utilized for performing numerical computations, implementing algorithms, and visualizing results. It was accessed through a compatible web browser, and the programming and data analysis capabilities of the MATLAB environment were leveraged.
- MATLAB version R2021a (Free Desktop Version): In addition to the online version, the free desktop version of MATLAB was also employed. Specifically, version R2021a was installed on the local computer and used for code development, debugging, conducting experiments, and performing additional analysis.

3. Theoretical Framework

Our proposed algorithm follows the general principles of the Hill cipher. It involves dividing the plaintext into blocks, each of which is represented as a matrix. These matrices are then multiplied by a secret key matrix, resulting in the ciphertext. The key innovation lies in the calculation of the matrix inverse using the Gauss-Jacques method, which offers a more efficient and modular approach compared to traditional methods. By utilizing modular inverse calculation, our algorithm ensures the encryption process remains secure even with a relatively small key size. All the theoretical information is found in the article Towards a Symmetric Crypto Algorithm: The HAJ [10]. The proposed HAJ algorithm, while focused on enhancing cryptographic robustness, also has the potential to be adapted for lightweight environments such as the Internet of Things (IoT), due to its modular structure and efficient matrix-based operations. In resource-constrained environments, encryption algorithms must ensure both computational efficiency and cryptographic strength. A recent approach proposes a lightweight symmetric cryptosystem based on AES and a chaotic S-box, designed for embedded platforms. Using techniques like the Lorenz system and Hilbert scan patterns, the system achieved validated cryptographic properties through NIST tests and demonstrated efficient performance on real-world sensors [11].

3.1. Hill Cipher

The Hill-Cipher algorithm is a linear substitution encryption algorithm that uses a matrix as a key to encrypt and decrypt the message. It was proposed by Lester S. Hill in 1929 [12], and is based on the key matrix, which is an invertible square matrix. The Hill Cipher algorithm is used in data security applications to protect information from malicious attacks. This algorithm is considered one of the first symmetric encryption algorithms and has been used in military and government applications for decades. However, its use has decreased due to the arrival of more secure and efficient algorithms [13].

In the Hill Cipher, the original message is divided into blocks of length n, where n is the size of the key matrix. Each block is multiplied by the key matrix, and the result is reduced modulo a prime number to obtain the encrypted message. The decryption process involves multiplying the encrypted message by the inverse matrix of the key and reducing modulo the prime number.

Below are some relevant experiments and references related to the Hill Cipher:

Experiment 1: Known-plaintext attack In this experiment, an attacker knows some blocks of plaintext and the corresponding blocks of ciphertext. Using this information, the attacker can calculate the key matrix by using linear algebra and decrypt the rest of the ciphertext message [15].

Experiment 2: Known-ciphertext attack In this experiment, an attacker knows some blocks of ciphertext and the corresponding blocks of plaintext. Using this information, the attacker can calculate the inverse matrix of the key by using linear algebra and decrypt the rest of the ciphertext message [17].

Experiment 3: Frequency analysis In this experiment, an attacker can use frequency analysis to decrypt the ciphertext message. The attacker can use the frequency of letters in the ciphertext message to determine the most common letters and, from there, deduce the key and decrypt the entire message [18].

Experiment 4: Side-chanel attack Side-channel leakages, such as electromagnetic emissions, can significantly reduce the security margin of symmetric encryption algorithms. Recent work proposed a key classification framework based on deep transfer learning, using a divide-and-conquer strategy over semi-byte structures to reduce key enumeration complexity. The model, composed of symmetric subnetworks and an end-to-end mapping design, achieved up to 91% attack accuracy while maintaining robustness under small-sample conditions [16].

Although the Hill Cipher is a solid encryption algorithm, it is not invulnerable to attacks. The experiments mentioned above demonstrate some of the vulnerabilities of the Hill Cipher and the importance of choosing an appropriate key matrix and ensuring that the prime number used for modulo reduction is large enough. Additionally, it is important to remember that encryption algorithms are one of the many tools used in data security, and data security depends on a combination of technical security measures and management procedures.

3.2. Gauss-Jacques Method for Modular Inverse Calculation

The Gauss-Jacques method plays a crucial role in our proposed symmetric cryptographic algorithm for modular inverse calculation. This method allows us to efficiently compute the modular inverse of variable-sized matrices, making it particularly suitable for cryptographic applications.

The Gauss-Jacques method is implemented as a separate function, taking two inputs: the matrix 'K' and the prime number 'm'. It ensures that 'K' is a square matrix, and if not, it raises an error. The function returns two outputs: the modular inverse matrix 'InvMod' and the identity matrix 'I'.

The algorithm proceeds as follows:

- The original matrix 'K' is augmented with the identity matrix to form an augmented matrix.
- For each row in 'K', a pivot element is identified. If the pivot is zero, a row interchange operation is performed to ensure a non-zero pivot.
- The extended Euclidean algorithm is used to calculate the modular inverse 'x' of the pivot element with respect to 'm'.
- The current row of 'K' is multiplied by 'x' modulo 'm' to obtain the updated row.
- For each row other than the current row, row operations are performed to eliminate the non-zero elements below and above the pivot element.
- Finally, the resulting matrix is divided into two parts: 'InvMod', which contains the modular inverse matrix, and 'I', which represents the identity matrix.

3.3. Extended Euclidean Algorithm

The extended Euclidean algorithm is a fundamental tool for finding the greatest common divisor (gcd) of two integer numbers in a Euclidean space. It is used within the Gauss-Jacques method to calculate the modular inverse of the pivot element.

The extended Euclidean algorithm function takes two positive integers, 'a' and 'b', as inputs. It returns three integers, 'x', 'y', and 'd', such that 'd' represents the gcd of 'a' and 'b', and 'ax + by = d'.

The algorithm proceeds as follows:

- If 'b' is equal to zero, the function sets 'x' to 1, 'y' to 0, and 'd' to 'a' before returning.

- Otherwise, the function recursively calls itself with 'b' and 'mod(a,b)' as the new inputs, storing the results in 'x1', 'y1', and 'd1'.
- 'x' is set to 'y1', 'y' is set to 'x1 - floor(a/b)*y1', and 'd' is set to 'd1'.

4. Methodology

The HAJ algorithm retains the Hill cipher's block-matrix multiplication as the core encryption operation. The novelty lies in the computation of the modular inverse of each key matrix: instead of the standard adjugate/determinant approach, HAJ applies the Gauss-Jacques elimination method to obtain the inverse modulo m. During each round, a fresh key matrix is generated and its inverse computed by Gauss-Jacques, enabling efficient decryption and supporting the multi-round, subkey structure.

4.1. A Round of HAJ

The previous section describes the generalization for creating and understanding how the HAJ algorithm would work, i.e., the summation of the blocks and the 7 times it will be performed. Now, in the following paragraphs, we will describe how we intend to generate each block, the bits that will be used, the size of each block, and how the primary and secondary keys will be generated.

4.1.1. Obtaining the String Size

There are different functions to know the size of a string, where any of them will be used to find that value. What is interesting in this section is that we seek for that value to be a multiple of three hundred. Therefore, we will use the modulo 300 of the string length to verify it, and if necessary, some characters will be added to the string to make it a multiple of three hundred.

Example code:

$ifs \neq 0;$

$a = 300 - s;$

$fori = 1 : a$

$p(lp + i) = 'Q';$

end

end

4.1.2. Key Generation

The primary key will be generated using the Gauss-Jacques method, where the size of the matrix will be determined by the size of the resulting block in the previous steps. However, for each block, a primary key will be generated, and for each round of the 7 rounds, secondary or sub-keys will be generated to improve security at each iteration.

For the secondary keys, we will use the left circular shift. A circular shift in this case to the left is the operation of rearranging the entries in a string, either by moving the final entry to the first position while shifting all other entries to the next position, or by performing the inverse operation. A circular shift is a special type of cyclic permutation, which in turn is a special type of permutation.

This operation is used in various fields and different algorithms. An example of this in cryptography is the Data Encryption Standard algorithm, where positions are moved from the order of the primary key with respect to a shift list determined beforehand.

4.2. Encryption and Decryption Process

To demonstrate the functionality of our proposed symmetric cryptographic algorithm, we implemented the encryption and decryption processes using the provided code. The

algorithm utilizes the Hill cipher with modular inverse calculation using the Gauss-Jacques method.

For encryption, the algorithm requires the selection of a prime number, denoted as 'm', greater than the highest ASCII value intended for encryption. Additionally, the number of matrices, 'num_matrices´, and the block size, 'block_size', need to be defined. The matrices are generated randomly, ensuring that the determinant is non-zero modulo 'm'. This approach aligns with recent advancements in symmetric key generation that integrate hybrid models based on Latin square matrices and subtractive random number generators to enhance key randomness, diffusion, and resistance to brute-force attacks [14]

The plaintext is obtained from the user and is processed in blocks. If the length of the plaintext is not a multiple of the block size, padding is applied to make it a multiple. The encryption process involves multiplying the block of plaintext with the corresponding key matrix, ensuring modulo 'm' arithmetic.

The decryption follows a similar process but uses the inverse of the key matrix calculated using the Gauss-Jacques method implemented in MATLAB [19]. The encrypted text is processed in blocks, and the decryption is performed by multiplying each encrypted text block with the inverse matrix modulo 'm'.

The resulting decrypted plaintext is displayed as the final output of the algorithm.

The algorithm 4.3 shows the pseudocode for a round of HAJ where the K is the same size as plaintext. For example, if the plaintext is about 170 characters, then the K is going to be 170x170 size and the algorithm 5 shows the pseudocode by blocks.

4.3. Pseudocode of a HAJ round

Listing 1: Pseudocode for text encryption and decryption

```
m = 257  # Modulus for encryption
seeds = [36, 33, 88, ..., 91, 1]  # List of 30 seeds
File = 'NationalAnthems/NameAnthem.txt'  # Path to the text file

# Read the text file
file = open(File, 'rb')
p = file.read()
file.close()

# Ensure the length of p is a multiple of 3
if len(p) % 3 != 0:
    # pad p with zeros

# Prepare the results array
results = [[0 for _ in range(4)] for _ in range(30)]

for i in range(len(seeds)):
    # Initialize random seed
    seed = generate_random_seed()
    initialize_random_number_generator(seed)

    # Generate a random matrix k
    k = generate_random_matrix(lonpL, lonpL)

    # Encryption
    c = (p * k) % m
```

```
    # Decryption
    InvMod = modular_inverse(k, m)
    p_decrypted = (c * InvMod) % m

    # Measure memory usage and execution time
    measure_memory_usage_and_execution_time()

    # Store the results
    store_results(results, i)

# Save results to a CSV file
save_results_to_csv(results)
```

5. The HAJ: Pseudocode

Listing 2: Pseudocode for text encryption and decryption with multiple rounds

```
m = 257  # Modulus for encryption
seeds = [280, 494, 50, ..., 404, 181]  # List of 30 seeds
File = 'NationalAnthems/NameAnthem.txt'  # Path to the text file

# Read the text file
file = open(File, 'rb')
p = file.read()
file.close()

# Ensure the length of p is a multiple of 300
if len(p) % 300 != 0:
    # pad p with zeros

# Prepare the results array
results = [[0 for _ in range(4)] for _ in range(len(seeds))]

numbloque = 3
tamanho_bloque = len(p) // numbloque

for j in range(len(seeds)):
    # Initialize random seed
    seed = generate_random_seed()
    initialize_random_number_generator(seed)

    for ronda in range(num_rondas):
        for i in range(numbloque):
            k = generate_random_matrix(tamanho_bloque, tamanho_bloque)

    encryption_file = open('encryption_results.txt', 'w')
    decryption_file = open('decryption_results.txt', 'w')

    # Encrypt block by block, 7 times
    c = []
```

```
        for ronda in range(num_rondas):
            for i in range(0, len(p), tamanho_bloque):
                block = get_current_block(p, i, tamanho_bloque)
                encrypted_block = encrypt_block(block)
                c.append(encrypted_block)
            write_encryption_results_to_file(encryption_file, c)

        # Decrypt block by block, 7 times
        p_decifrado = []
        for ronda in range(num_rondas):
            for i in range(0, len(c), tamanho_bloque):
                block = get_current_block(c, i, tamanho_bloque)
                decrypted_block = decrypt_block(block)
                p_decifrado.append(decrypted_block)
            write_decryption_results_to_file(decryption_file, p_decifrado)

        close_files(encryption_file, decryption_file)

        # Measure memory usage
        measure_memory_usage()

        # Store the results
        store_results(results, j)

# Save results to a CSV file
save_results_to_csv(results)
```

6. Experimental Results

To evaluate the performance and security of our algorithm, we conducted a series of experiments using various plaintexts and key sizes. The results demonstrate that our algorithm achieves robust encryption while maintaining computational efficiency.

6.1. Phase 1

The experimentation is carried out with ten national anthems, which are listed below:

1. Australia
2. Canada
3. Great Britain
4. India
5. Ireland
6. Jamaica
7. New Zealand
8. Singapore
9. South Africa
10. USA

These anthems were used for the first part of the experimentation. In this specific chapter, the experimentation consists of a single round of the HAJ algorithm. That is, encryption and decryption are performed only once. The particularity is that a matrix of the size of the plaintext is generated. The length of each anthem is as shown in table 1:

Table 1. Leng of each anthem

Anthem	**Length**
Australia	620
Canadá	316
Gran Bretaña	496
India	336
Irlanda	800
Jamaica	591
Nueva Zelanda	1190
Singapur	204
Sudáfrica	226
USA	1705

For example, the national anthem of the USA consists of 1705 characters with all the current verses and choruses, without being processed through the algorithm to make it a multiple. Therefore, a random matrix K of 1705x1705 elements ranging from 0 to 256 is generated, which is multiplied by the plaintext to generate the ciphertext. For this particular example described here, it would be the anthem of the USA.

The tables 2, 3, and 4 show the average, standard deviation, maximum, and minimum values of the execution time expressed in seconds, as well as the memory usage represented in Mb, from the thirty runs performed for each anthem where the key is the same size as the plaintex.

Table 2. Mac Runs Summary

Anthem	Parameters	Execution Time (s)	Memory Usage (MB)
Australia	Average	13.3370	5.9330
	SD	2.2320	0.0070
	Max	21.8156	5.9398
	Min	11.7482	5.9011
Canada	Average	2.2672	1.6134
	SD	0.1914	0.0070
	Max	2.7470	1.6202
	Min	2.0245	1.5815
Great Britain	Average	7.9106	3.8298
	SD	1.6691	0.0070
	Max	15.6925	3.8366
	Min	6.6204	3.7979
India	Average	2.5626	1.7630
	SD	0.1623	0.0068
	Max	2.9756	1.7697
	Min	2.3115	1.7323
Ireland	Average	27.7280	9.8431
	SD	3.9408	0.0070
	Max	39.7620	9.8499
	Min	24.6128	9.8112
Jamaica	Average	11.6885	5.3762
	SD	1.3711	0.0068
	Max	15.1518	5.3830
	Min	10.1736	5.3455
New Zealand	Average	92.7189	21.7070
	SD	18.0516	0.0070
	Max	149.5322	21.7138
	Min	82.6005	21.6751
Singapore	Average	0.7071	0.6921
	SD	0.1679	0.0070
	Max	1.3809	0.6989
	Min	0.5553	0.6602
South Africa	Average	0.7658	0.8320
	SD	0.0520	0.0070
	Max	0.9386	0.8388
	Min	0.7127	0.8001
USA	Average	280.4113	44.5373
	SD	59.3591	0.0070
	Max	576.2381	44.5441
	Min	255.1275	44.5054

Table 3. Windows Runs Summary

Anthem	Parameters	Execution Time (s)	Memory Usage (MB)
Australia	Average	8.2658	5.9330
	SD	0.1805	0.0070
	Max	8.5023	5.9398
	Min	7.9580	5.9011
Canada	Average	1.0878	1.5841
	SD	0.0289	0.0070
	Max	1.1596	1.5909
	Min	1.0596	1.5522
Great Britain	Average	3.9785	3.8298
	SD	0.0473	0.0070
	Max	4.1067	3.8366
	Min	3.9137	3.7979
India	Average	1.2708	1.7630
	SD	0.0165	0.0068
	Max	1.3227	1.7697
	Min	1.2483	1.7323
Ireland	Average	17.6643	9.8431
	SD	0.3715	0.0070
	Max	18.3467	9.8499
	Min	17.0091	9.8112
Jamaica	Average	7.0676	5.3762
	SD	0.1952	0.0068
	Max	7.5738	5.3830
	Min	6.7943	5.3455
New Zealand	Average	61.2749	21.7070
	SD	2.0358	0.0070
	Max	64.1252	21.7138
	Min	57.9498	21.6751
Singapore	Average	0.3318	0.6921
	SD	0.0617	0.0070
	Max	0.6527	0.6989
	Min	0.3075	0.6602
South Africa	Average	0.4134	0.8321
	SD	0.0107	0.0070
	Max	0.4581	0.8388
	Min	0.4010	0.8001
USA	Average	198.1801	44.5373
	SD	11.1756	0.0070
	Max	213.1193	44.5441
	Min	176.1961	44.5054

Table 4. Online Runs Summary

Anthem	Parameters	Execution Time (s)	Memory Usage (MB)
Australia	Average	5.4928	5.9330
	SD	0.0870	0.0070
	Max	5.7460	5.9398
	Min	5.4067	5.9011
Canada	Average	0.9267	1.6134
	SD	0.0317	0.0070
	Max	1.0079	1.6202
	Min	0.8720	1.5815
Great Britain	Average	3.0410	3.8298
	SD	0.0336	0.0070
	Max	3.1122	3.8366
	Min	2.9755	3.7979
India	Average	1.0303	1.7630
	SD	0.0250	0.0068
	Max	1.0771	1.7697
	Min	0.9749	1.7323
Ireland	Average	11.6492	9.8431
	SD	0.2980	0.0070
	Max	12.6112	9.8499
	Min	11.3622	9.8112
Jamaica	Average	5.0064	5.3762
	SD	0.0669	0.0068
	Max	5.2021	5.3830
	Min	4.8887	5.3455
New Zealand	Average	56.5294	29.3796
	SD	1.1452	0.0070
	Max	59.4291	29.3864
	Min	54.8559	29.3477
Singapore	Average	0.2881	0.6921
	SD	0.0098	0.0070
	Max	0.3312	0.6989
	Min	0.2803	0.6602
South Africa	Average	0.3697	0.8321
	SD	0.0113	0.0070
	Max	0.4188	0.8388
	Min	0.3588	0.8001
USA	Average	103.6944	44.5373
	SD	3.7643	0.0070
	Max	110.1514	44.5441
	Min	98.1536	44.5054

6.2. Phase 2 Results

The anthems used for this phase are the same that we used on subsection 6.1. The difference is that the length of each anthem changed because we made them multiple of 300 for experimental purposes. The length of each anthem are as show table 5:

Table 5. Leng of each anthem for blocks

Anthem	Length
Australia	900
Canadá	600
Gran Bretaña	600
India	600
Irlanda	900
Jamaica	600
Nueva Zelanda	1200
Singapur	300
Sudáfrica	300
USA	1800

The tables 6, 7, and 8 show the execution for each anthem dividing the plaintex into blocks of 3 and creating 21 keys the size of each block, this is because there is one key for each block and each round. The tables show as well the average, standard deviation, maximum, and minimum values of the execution time and memory usage.

The tables 9, 10, and 11 show the execution for each anthem dividing the plaintex into blocks of 30 and creating 210 keys the size of each block, this is because there is one key for each block and each round. The tables show as well the average, standard deviation, maximum, and minimum values of the execution time and memory usage.

The tables 12, 13, and 14 show the execution for each anthem dividing the plaintex into blocks of 300 and creating 2100 keys the size of each block, this is because there is one key for each block and each round. The tables show as well the average, standard deviation, maximum, and minimum values of the execution time and memory usage.

6.2.1. Three Blocks

Table 6. Runs per block (three) Mac

Anthem	Parameters	ExecutionTime(s)	MemoryUsage(MB)
Australia	Average	36.4588	15.8624
	SD	1.4329	0.0089
	Max	40.495	15.8695
	Min	34.5879	15.8193
Canada	Average	11.5354	7.0801
	SD	0.584	0.0089
	Max	13.08	7.0873
	Min	10.9522	7.0371
Great Britain	Average	11.4738	7.0815
	SD	0.4413	0.0089
	Max	12.3882	7.0886
	Min	10.9528	7.0385
India	Average	11.521	7.0803
	SD	0.5554	0.0089
	Max	13.4812	7.0874
	Min	10.9673	7.0372
Ireland	Average	36.5324	15.8637
	SD	1.2092	0.0089
	Max	39.1249	15.8709
	Min	34.52	15.8207
Jamaica	Average	11.285	7.0822
	SD	0.4914	0.0089
	Max	12.6359	7.0894
	Min	10.7407	7.0392
New Zealand	Average	83.7029	28.1561
	SD	3.1496	0.0089
	Max	93.8784	28.1633
	Min	79.3689	28.1131
Singapore	Average	1.5835	1.8089
	SD	0.1966	0.0089
	Max	2.2932	1.816
	Min	1.4239	1.7658
South Africa	Average	1.6398	1.8091
	SD	0.063	0.0089
	Max	1.8269	1.8162
	Min	1.4895	1.766
USA	Average	277.0724	63.2675
	SD	17.8858	0.0089
	Max	301.9495	63.2746
	Min	246.4505	63.2244

Table 7. Runs per block (three) Windows

Anthem	Parameters	ExecutionTime(s)	MemoryUsage(MB)
Australia	Average	22.2355	15.8624
	SD	0.0704	0.0089
	Max	22.3828	15.8695
	Min	22.0792	15.8193
Canada	Average	7.1483	7.0801
	SD	0.8943	0.0089
	Max	11.5577	7.0873
	Min	6.872	7.0371
Great Britain	Average	30.0705	7.0815
	SD	124.5569	0.0089
	Max	689.5307	7.0886
	Min	6.9026	7.0385
India	Average	6.9861	7.0803
	SD	0.1767	0.0089
	Max	7.8852	7.0874
	Min	6.9125	7.0372
Ireland	Average	22.2087	15.8637
	SD	0.0743	0.0089
	Max	22.4107	15.8709
	Min	22.0785	15.8207
Jamaica	Average	6.9500	7.0822
	SD	0.0384	0.0089
	Max	7.0793	7.0894
	Min	6.9087	7.0392
New Zealand	Average	52.8028	28.1561
	SD	3.2683	0.0089
	Max	62.2822	28.1633
	Min	50.2036	28.1131
Singapore	Average	0.9744	1.8089
	SD	0.0135	0.0089
	Max	1.0212	1.8160
	Min	0.9550	1.7658
South Africa	Average	0.9827	1.8091
	SD	0.0106	0.0089
	Max	1.016	1.8162
	Min	0.9656	1.766
USA	Average	180.5225	63.2675
	SD	51.7192	0.0089
	Max	453.0388	63.2746
	Min	161.461	63.2244

Table 8. Runs per block (three) Online

Anthem	Parameters	ExecutionTime(s)	MemoryUsage(MB)
Australia	Average	16.1643	15.8624
	SD	1.0832	0.0089
	Max	19.7402	15.8695
	Min	15.4277	15.8193
Canada	Average	6.4902	7.0801
	SD	0.9334	0.0089
	Max	9.0094	7.0873
	Min	6.0004	7.0371
Great Britain	Average	6.4617	7.0815
	SD	0.7231	0.0089
	Max	8.1688	7.0886
	Min	5.9552	7.0385
India	Average	6.0322	7.0803
	SD	0.0324	0.0089
	Max	6.1227	7.0874
	Min	5.9866	7.0372
Ireland	Average	15.6482	15.8637
	SD	0.2025	0.0089
	Max	16.2677	15.8709
	Min	15.4033	15.8207
Jamaica	Average	6.6892	7.0822
	SD	0.9726	0.0089
	Max	8.6431	7.0894
	Min	5.9876	7.0392
New Zealand	Average	34.5667	28.1574
	SD	0.4773	0.0091
	Max	36.2345	28.1646
	Min	34.0755	28.1131
Singapore	Average	0.9882	1.8089
	SD	0.0109	0.0089
	Max	1.0342	1.816
	Min	0.9801	1.7658
South Africa	Average	0.9895	1.8091
	SD	0.0083	0.0089
	Max	1.0206	1.8162
	Min	0.9815	1.766
USA	Average	113.6309	63.2687
	SD	1.6772	0.0091
	Max	116.8265	63.2759
	Min	111.2799	63.2244

6.2.2. Thirty Blocks

Table 9. Runs per block (thirty) Mac

Anthem	Parameters	ExecutionTime(s)	MemoryUsage(MB)
Australia	Average	0.7155	1.5414
	SD	0.8525	0.0089
	Max	5.1868	1.5485
	Min	0.4533	1.4983
Canada	Average	0.4028	0.7251
	SD	0.9142	0.0089
	Max	5.2114	0.7322
	Min	0.1600	0.6820
Great Britain	Average	0.3778	0.7265
	SD	1.0161	0.0089
	Max	5.7564	0.7336
	Min	0.1704	0.6834
India	Average	0.3017	0.7252
	SD	0.4158	0.0089
	Max	2.4866	0.7324
	Min	0.1741	0.6822
Ireland	Average	0.7427	1.5427
	SD	0.9062	0.0089
	Max	5.4912	1.5499
	Min	0.4681	1.4997
Jamaica	Average	0.2054	0.7272
	SD	0.0472	0.0089
	Max	0.3668	0.7343
	Min	0.1560	0.6841
New Zealand	Average	1.2550	2.6832
	SD	0.3763	0.0089
	Max	3.2087	2.6903
	Min	1.1071	2.6401
Singapore	Average	0.2285	0.2338
	SD	0.7445	0.0089
	Max	4.1370	0.2409
	Min	0.0377	0.1907
South Africa	Average	0.0624	0.2339
	SD	0.0138	0.0089
	Max	0.0919	0.2411
	Min	0.0424	0.1909
USA	Average	5.1072	5.9324
	SD	1.6908	0.0089
	Max	9.7633	5.9396
	Min	3.1777	5.8894

Table 10. Runs per block (thirty) Windows

Anthem	Parameters	ExecutionTime(s)	MemoryUsage(MB)
Australia	Average	0.3077	1.5414
	SD	0.0204	0.0089
	Max	0.3641	1.5485
	Min	0.2825	1.4983
Canada	Average	0.1028	0.7251
	SD	0.0098	0.0089
	Max	0.1400	0.7322
	Min	0.0934	0.682
Great Britain	Average	0.1063	0.7265
	SD	0.0165	0.0089
	Max	0.1791	0.7336
	Min	0.0938	0.6834
India	Average	0.1048	0.7252
	SD	0.0157	0.0089
	Max	0.1687	0.7324
	Min	0.0929	0.6822
Ireland	Average	0.298	1.5427
	SD	0.0135	0.0089
	Max	0.3429	1.5499
	Min	0.2849	1.4997
Jamaica	Average	0.1042	0.7272
	SD	0.0115	0.0089
	Max	0.1449	0.7343
	Min	0.0926	0.6841
New Zealand	Average	0.6614	2.6832
	SD	0.0136	0.0089
	Max	0.716	2.6903
	Min	0.6445	2.6401
Singapore	Average	0.0287	0.2338
	SD	0.008	0.0089
	Max	0.0622	0.2409
	Min	0.02	0.1907
South Africa	Average	0.0277	0.2339
	SD	0.007	0.0089
	Max	0.052	0.2411
	Min	0.0184	0.1909
USA	Average	2.1344	5.9324
	SD	0.0177	0.0089
	Max	2.1894	5.9396
	Min	2.112	5.8894

Table 11. Runs per block (thirty) Online

Anthem	Parameters	ExecutionTime(s)	MemoryUsage(MB)
Australia	Average	0.3089	1.5414
	SD	0.0091	0.0089
	Max	0.3470	1.5485
	Min	0.3034	1.4983
Canada	Average	0.1070	0.7251
	SD	0.0069	0.0089
	Max	0.1421	0.7322
	Min	0.1035	0.6820
Great Britain	Average	0.1089	0.7265
	SD	0.0093	0.0089
	Max	0.1499	0.7336
	Min	0.1043	0.6834
India	Average	0.1091	0.7252
	SD	0.0124	0.0089
	Max	0.1695	0.7324
	Min	0.1036	0.6822
Ireland	Average	0.3073	1.5427
	SD	0.0084	0.0089
	Max	0.3432	1.5499
	Min	0.3026	1.4997
Jamaica	Average	0.1083	0.7272
	SD	0.0080	0.0089
	Max	0.1460	0.7343
	Min	0.1042	0.6841
New Zealand	Average	0.6830	2.6832
	SD	0.0124	0.0089
	Max	0.7256	2.6903
	Min	0.6724	2.6401
Singapore	Average	0.0289	0.2338
	SD	0.0067	0.0089
	Max	0.0602	0.2409
	Min	0.0254	0.1907
South Africa	Average	0.0283	0.2339
	SD	0.0084	0.0089
	Max	0.0590	0.2411
	Min	0.0246	0.1909
USA	Average	2.1718	5.9324
	SD	0.0105	0.0089
	Max	2.2157	5.9396
	Min	2.1613	5.8894

6.2.3. Three Hundred Blocks

Table 12. Runs per block (three hundred) for Mac

Anthem	Parameters	ExecutionTime(s)	MemoryUsage(MB)
Australia	Average	0.0819	0.4172
	SD	0.0291	0.0089
	Max	0.1932	0.4243
	Min	0.0573	0.3742
Canada	Average	0.0771	0.3295
	SD	0.0241	0.0089
	Max	0.1360	0.3367
	Min	0.0533	0.2865
Great Britain	Average	0.0762	0.3309
	SD	0.0347	0.0089
	Max	0.1894	0.3381
	Min	0.0441	0.2879
India	Average	0.0645	0.3297
	SD	0.0230	0.0089
	Max	0.1288	0.3368
	Min	0.0470	0.2866
Ireland	Average	0.0780	0.4186
	SD	0.0301	0.0089
	Max	0.1845	0.4257
	Min	0.0524	0.3755
Jamaica	Average	0.0733	0.3316
	SD	0.0232	0.0089
	Max	0.1178	0.3388
	Min	0.0444	0.2886
New Zealand	Average	0.1009	0.5390
	SD	0.0396	0.0088
	Max	0.2070	0.5461
	Min	0.0766	0.4959
Singapore	Average	0.0654	0.2754
	SD	0.0210	0.0089
	Max	0.1293	0.2826
	Min	0.0424	0.2324
South Africa	Average	0.0604	0.2756
	SD	0.0253	0.0089
	Max	0.1490	0.2827
	Min	0.0389	0.2325
USA	Average	0.1395	0.8740
	SD	0.0418	0.0088
	Max	0.2806	0.8811
	Min	0.1132	0.8309

Table 13. Runs per block (three hundred) for Windows

Anthem	Parameters	ExecutionTime(s)	MemoryUsage(MB)
Australia	Average	0.0384	0.4172
	SD	0.0095	0.0089
	Max	0.0588	0.4243
	Min	0.0271	0.3742
Canada	Average	0.0356	0.3295
	SD	0.0083	0.0089
	Max	0.0670	0.3367
	Min	0.0254	0.2865
Gran Bretaña	Average	0.0348	0.3309
	SD	0.0104	0.0089
	Max	0.0673	0.3381
	Min	0.0232	0.2879
India	Average	0.0338	0.3297
	SD	0.0153	0.0089
	Max	0.1033	0.3368
	Min	0.0235	0.2866
Ireland	Average	0.0374	0.4186
	SD	0.0108	0.0089
	Max	0.0711	0.4257
	Min	0.0280	0.3755
Jamaica	Average	0.0322	0.3316
	SD	0.0061	0.0089
	Max	0.0462	0.3388
	Min	0.0238	0.2886
New Zealand	Average	0.0457	0.5390
	SD	0.0111	0.0088
	Max	0.0790	0.5461
	Min	0.0335	0.4959
Singapore	Average	0.0285	0.2754
	SD	0.0065	0.0089
	Max	0.0550	0.2826
	Min	0.0210	0.2324
South Africa	Average	0.0251	0.2756
	SD	0.0059	0.0089
	Max	0.0414	0.2827
	Min	0.0170	0.2325
USA	Average	0.0785	0.8740
	SD	0.0285	0.0088
	Max	0.2104	0.8811
	Min	0.0610	0.8309

Table 14. Runs per block (three hundred) Online

Anthem	Parameters	ExecutionTime(s)	MemoryUsage(MB)
Australia	Average	0.0355	0.4172
	SD	0.0125	0.0089
	Max	0.1008	0.4243
	Min	0.0312	0.3742
Canada	Average	0.0308	0.3295
	SD	0.0147	0.0089
	Max	0.1064	0.3367
	Min	0.0261	0.2865
Great Britain	Average	0.0297	0.3309
	SD	0.0092	0.0089
	Max	0.0661	0.3381
	Min	0.0252	0.2879
India	Average	0.0283	0.3297
	SD	0.0079	0.0089
	Max	0.0697	0.3368
	Min	0.0258	0.2866
Ireland	Average	0.0337	0.4186
	SD	0.0058	0.0089
	Max	0.0615	0.4257
	Min	0.0303	0.3755
Jamaica	Average	0.0293	0.3316
	SD	0.0079	0.0089
	Max	0.0677	0.3388
	Min	0.0258	0.2886
New Zealand	Average	0.0426	0.5390
	SD	0.0106	0.0088
	Max	0.0782	0.5461
	Min	0.0378	0.4959
Singapore	Average	0.0243	0.2754
	SD	0.0034	0.0089
	Max	0.0416	0.2826
	Min	0.0223	0.2324
South Africa	Average	0.0246	0.2756
	SD	0.0063	0.0089
	Max	0.0579	0.2827
	Min	0.0224	0.2325
USA	Average	0.0705	0.8740
	SD	0.0095	0.0088
	Max	0.1158	0.8811
	Min	0.0637	0.8309

6.3. Comparison Between the Performance of the Devices

In general we can see that our algorithm is very stable in memory use. In each device where the algorithm was executed had a very similar memory usage consumption.

The difference was in the execution time, with the MacBook Air being the device with the worst performance. The Windows device and the online version had very similar execution times.

The Figure 1 show the average execution time and memory usage of the runs for a single round of the HAJ. Where the key is the same size as the plaintext. In the x-axis, we can see the numbers from 1 to 10. These numbers are the anthems as listed in table 6.1.

The Figure 2 show the average execution time and memory usage of the runs by three blocks for the plaintext and running it seven times. Where the key is the size of each block. In the x-axis, we can see the numbers from 1 to 10. These numbers are the anthems as listed in table 6.1.

The Figure 3 show the average execution time and memory usage of the runs by thirty blocks for the plaintext and running it seven times. Where the key is the size of each block. In the x-axis, we can see the numbers from 1 to 10. These numbers are the anthems as listed in table 6.1.

The Figure 4 show the average execution time and memory usage of the runs by three hundred blocks for the plaintext and running it seven times. Where the key is the size of each block. In the x-axis, we can see the numbers from 1 to 10. These numbers are the anthems as listed in table 6.1.

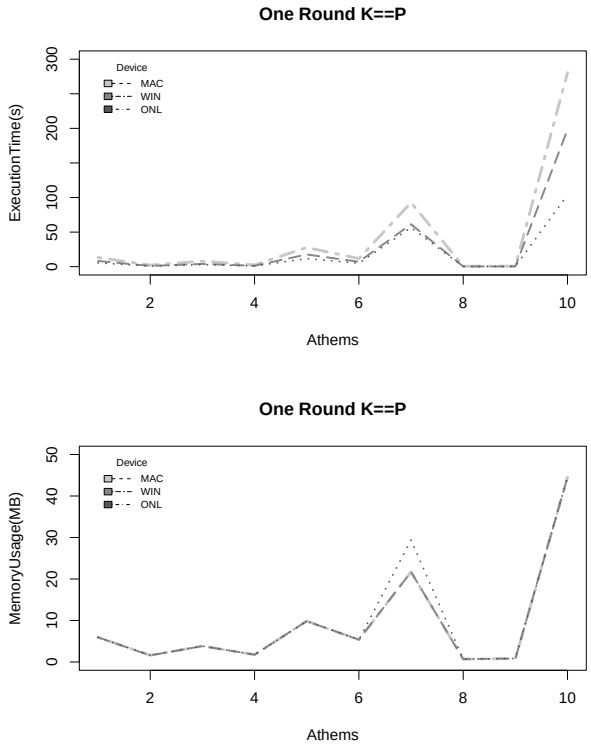

Figure 1. Average accuracy across 30 runs

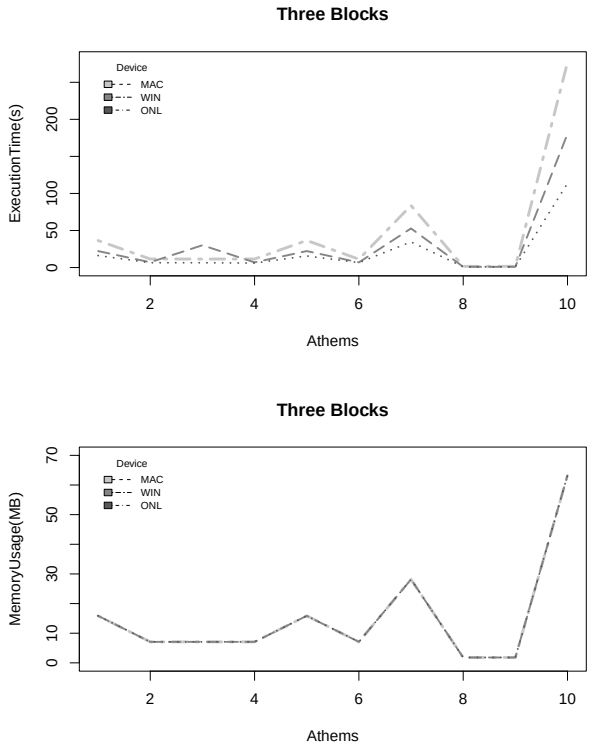

Figure 2. Average accuracy across 30 runs

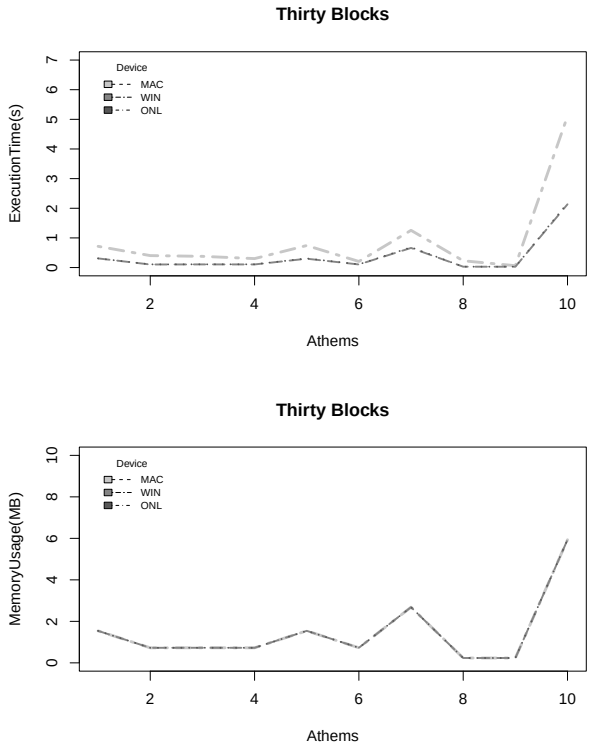

Figure 3. Average accuracy across 30 runs

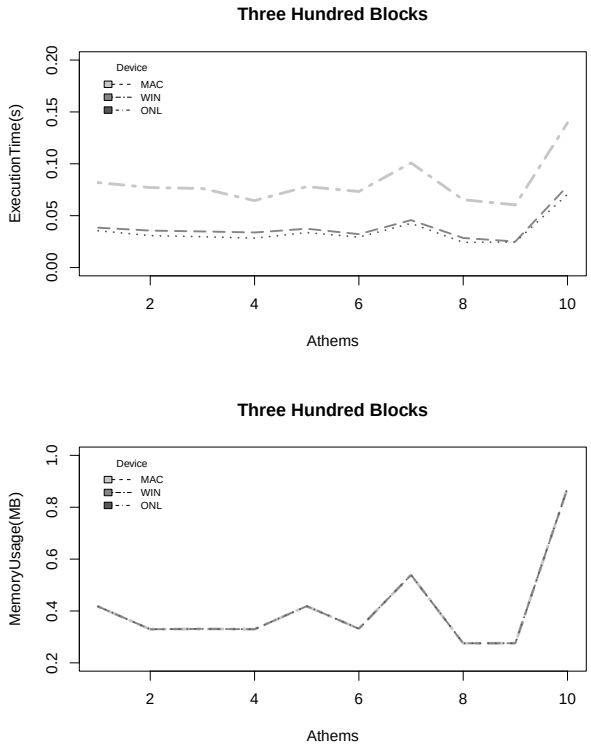

Figure 4. Average accuracy across 30 runs

7. Conclusions

In conclusion, we have proposed a symmetric cryptographic algorithm based on the Hill cipher with the Gauss-Jacques method for modular inverse calculation. The combination of these techniques provides improved security and computational efficiency compared to traditional approaches. Our algorithm presents a viable option for applications where fast and secure symmetric encryption is required. Future research could focus on further enhancing the algorithm's resistance to advanced cryptanalysis techniques and exploring its performance in resource-constrained environments.

The combination of the Gauss-Jacques method and the extended Euclidean algorithm allows for efficient and secure modular inverse calculation, which is essential for the proper functioning of our proposed symmetric cryptographic algorithm. However, we acknowledge that this work primarily focused on the algorithmic design and performance evaluation. A comprehensive security analysis, including statistical testing (e.g., NIST suite), differential and linear cryptanalysis, and evaluation against known-plaintext and chosen-plaintext attacks, remains as future work. Conducting these studies will be essential to rigorously validate the cryptographic strength of HAJ and to substantiate its potential as a secure alternative in practical applications.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest..

References

1. Stallings, W. *Cryptography and Network Security: Principles and Practice*; Pearson Education, 2017.

2. Paar, C.; Pelzl, J. *Understanding Cryptography: A Textbook for Students and Practitioners*; Springer Science & Business Media, 2010.
3. Radhakrishnan, I.; et al. Efficiency and Security Evaluation of Lightweight Cryptographic Algorithms for Resource-Constrained IoT Devices. *Sensors* **2024**, *24*(12), 4008. doi:10.3390/s24124008.
4. Sarker, K.U.; et al. A Systematic Review on Lightweight Security Algorithms for Resource-Constrained Systems. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* **2025**. doi:10.1007/s43926-025-00150-4.
5. Wen, H.; Lin, Y.; Yang, L.; Chen, R. Cryptanalysis of an Image Encryption Scheme Using Variant Hill Cipher and Chaos. *Expert Systems with Applications* **2024**, *250*, 123748. doi:10.1016/j.eswa.2024.123748.
6. Teseleanu, G. Security Analysis of an Image Encryption Scheme Based on a New Secure Variant of Hill Cipher and 1D Chaotic Maps. In *Proceedings of the 10th International Conference on Information Systems Security and Privacy (ICISSP 2024)*; SCITEPRESS: Angers, France, 2024; pp. 745–749. doi:10.5220/0012356600003648.
7. Bufalo, M. Some Properties of the Computation of the Modular Inverse with Applications in Cryptography. *Mathematics* **2023**, *11*(4), 70. doi:10.3390/math11040070.
8. Xu, G.; Tian, Y.; Yang, B. On the Inversion Modulo a Power of an Integer. *arXiv Preprint* **2025**, arXiv:2506.02491. Available online: https://arxiv.org/abs/2506.02491.
9. Rrghout, H.; Kattass, M.; Qobbi, Y.; Benazzi, N.; Jarjar, A. Robust Image Encryption Algorithm Using a New Variant of Hill Cipher. In *Proceedings of the 6th International Conference on NISS*; ACM: New York, NY, USA, 2023. doi:10.1145/3607720.3607750.
10. Alarcon, A. Towards a Symmetric Crypto Algorithm: The HAJ. ITNG 2021 18th International Conference on Information Technology-New Generations, 2021.
11. B. M. Alshammari, R. Guesmi, T. Guesmi, H. Alsaif, and A. Alzamil, "Implementing a symmetric lightweight cryptosystem in highly constrained IoT devices by using a chaotic S-box," *Symmetry*, vol. 13, no. 1, p. 129, 2021.
12. Hill, L.S. Cryptography in an Algebraic Alphabet. *The American Mathematical Monthly* **1929**, *36*, 306–312.
13. Schneier, B. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2 ed.; John Wiley & Sons, 1996.
14. N. H. M. Ali, M. M. Hoobi, and D. F. Saffo, "Development of Robust and Efficient Symmetric Random Keys Model based on the Latin Square Matrix," *Mesopotamian Journal of Cybersecurity*, vol. 4, no. 3, pp. 203–215, 2024.
15. Singh, S.; Srivastava, S. Cryptanalysis of Hill cipher using known plaintext attack. *International Journal of Scientific and Research Publications* **2014**, *4*, 1–4.
16. X. Cui, H. Zhang, X. Fang, Y. Wang, D. Wang, F. Fan, and L. Shu, "A secret key classification framework of symmetric encryption algorithm based on deep transfer learning" *Applied Sciences*, vol. 13, no. 21, p. 12025, 2023.
17. Agrawal, M.; Gupta, D. Cryptanalysis of Hill Cipher with Known Ciphertext Attack. *International Journal of Computer Applications* **2013**, *72*, 39–44.
18. Singh, S.; Srivastava, S. Cryptanalysis of Hill Cipher using Frequency Attack. *International Journal of Computer Applications* **2014**, *86*, 1–5.
19. Cantón, D. Gauss-Jacques Method. https://github.com/dCantonE/gauss-jacques, 2023. GitHub.

Publicación teórica del Modelo HAJ

7.9. Artículo Publicado

Publicación derivada de la tesis.

Este artículo fue aceptado y publicado en la revista *ITNG 2021 18th International Conference on Information Technology-New Generations. Advances in Intelligent Systems and Computing, vol 1346.* de Springer Alarcón-Narváez y Jacques García (2021) `https://doi.org/10.1007/978-3-030-70416-2_15`.

Towards a Symmetric Crypto Algorithm: The HAJ

15

Daniel Alarcón-Narváez and Fausto A. Jacques García

Abstract

In symmetric encryption, the algorithm and secret key determine the security factor. This paper presents the idea to create a multiple (7 times) and block-separated encryption algorithm. To achieve this, we will use the Hill Cipher and Gauss-Jacques methods. In addition to the above, our most significant contribution will be to obtain large secret keys, which will allow us to obtain as a possible result a meaningful approximation to the Shannon perfect secrecy, as well as the reduction of computational complexity and the verification of security through anti-bot mechanisms such as code breakers.

Keywords

AES · Block cipher · Ciphertext · Cryptography · Gauss-Jacques · Hill cipher · Modular inverse matrix · Plaintext · Secret key · Symmetric encryption

15.1 Introduction

Symmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the same key. In other words, those that require both parties to use the same secret key. Algorithms that use a shared key are known as symmetric algorithms. Figure 15.1 illustrates the basic encryption of symmetric key. It is also known as conventional encryption. The two types of attacks on an encryption algorithm are crypto-analysis, based on the encryption algorithm's properties and brute-force, which involves testing all possible keys. Any symmetric encryption scheme contains five main parts:

- Plaintext: This is the message or data to be encrypted and it is the input of the algorithm.
- Encryption algorithm: It performs substitutions or transformations on the plaintext.
- Secret key: It is also an input to the algorithm. That depending on the key that is provided, will be the output that we will obtain.
- Ciphertext: This is the message or encoded data obtained from the plaintext and the secret key as output.
- Decryption Algorithm: This is essentially the encryption algorithm that runs in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

In symmetric encryption, they use two techniques: substitution or transposition. Substitution techniques map the elements of the plaintext to elements of the ciphertext. The transposition techniques systematically transpose the positions of the elements of the plaintext. For our work, we will focus on a substitution technique called Hill Cipher. Below we listed the most common techniques in the literature [1].

- Caesar Cipher
- Monoalphabetic Ciphers
- Playfair Cipher
- Hill Cipher
- Polyalphabetic Ciphers
- One-Time Pad
- Advanced Encryption Standard (AES)

The advanced encryption standard is one of the most popular symmetric ciphers and, therefore, the most used. We mention this because our proposed algorithm pretends to be the same or more secure, with the difference also that our algorithm will require a less computational cost.

D. Alarcón-Narváez (✉) · F. A. Jacques García
Computer Science School, Autonomous University of Queretaro, Queretaro, Mexico
e-mail: dalarcon15@alumnos.uaq.mx; jacques@uaq.edu.mx

S. Latifi (ed.), *ITNG 2021 18th International Conference on Information Technology-New Generations*, Advances in Intelligent Systems and Computing 1346,
https://doi.org/10.1007/978-3-030-70416-2_15

Artículo Publicado

7.10. Linea de investigación futura (Publicado)

Publicación derivada de la tesis.

Este artículo fue aceptado y publicado en la revista *Cryptography* de MDPIAlarcón-Narváez, Lizama-Pérez, y Jacques-García (2026) `https://doi.org/10.3390/cryptography10010007`.

cryptography

MDPI

Article

Autopotency and Conjugacy of Non-Diagonalizable Matrices for Challenge–Response Authentication

Daniel Alarcón-Narváez [1,*], Luis Adrián Lizama-Pérez [2] and Fausto Abraham Jacques-García [1]

[1] Computer Science, Autonomous University of Querétaro (UAQ), Av. de las Ciencias, Querétaro 76230, Querétaro, Mexico; jacques@uaq.edu.mx
[2] Department of Electronics, Universidad Técnica Federico Santa María, Av. Vicuña Mackenna, San Joaquín 8940897, Chile; luis.lizamap@usm.cl
* Correspondence: daniel.alarcon@uaq.mx

Abstract

We present an algebraic framework for constructing challenge–response authentication protocols based on powers of non-diagonalizable matrices over finite fields. The construction relies on upper triangular Toeplitz matrices with a single Jordan block and on their structured power expansions, which induce nonlinear relations between matrix parameters and exponents through an *autopotency* phenomenon. The protocol is built from a cyclic family of matrix products derived from secret matrices $(\mathbf{A}_i)_{i=1}^n \subset GL_k(\mathbb{F}_p)$: for each index i, a product $\mathbf{P}_i = \mathbf{A}_i\mathbf{A}_{i+1}\cdots\mathbf{A}_{i+n-1}$ is formed (indices modulo n), and its power $\mathbf{P}_i^{(x)}$ is published for a secret exponent x. The resulting family of powered products is linked by conjugation via the unknown factors $\mathbf{A}_i$, enabling an interactive authentication mechanism in which the prover demonstrates the knowledge of selected factors by satisfying explicit conjugacy relations. We formalize the underlying algebraic problems in terms of factor recovery and conjugacy identification from powered products, and analyze how the enforced non-diagonalizable structure and Toeplitz constraints lead to coupled multivariate polynomial systems. These systems arise naturally from the algebraic design of the construction and do not admit immediate reductions to classical discrete logarithm settings. The framework illustrates how non-diagonalizable matrix structures and structured conjugacy relations can be used to define concrete authentication primitives in noncommutative algebraic settings, and provides a basis for further cryptanalytic and cryptographic investigation.

Keywords: autopotency; conjugacy search problem; toeplitz matrices; matrix-based cryptography; challenge–response authentication

Academic Editor: Josef Pieprzyk

Received: 22 December 2025
Revised: 9 January 2026
Accepted: 15 January 2026
Published: 18 January 2026

1. Introduction

The rapid advancement of quantum computing has precipitated an urgent need for cryptographic primitives that remain secure against quantum-capable adversaries [1,2]. Traditional public-key cryptosystems, which derive their security from the hardness of integer factorization or the discrete logarithm problem in finite fields and elliptic curves, are rendered vulnerable by Shor's algorithm [3]. This existential threat has catalyzed global standardization efforts, notably by NIST and ETSI, to identify and vet post-quantum cryptographic (PQC) schemes [4,5]. The current PQC landscape is dominated by several families [6], including lattice-based constructions (e.g., Kyber [7,8] and Dilithium [9,10]), hash-based signatures (e.g., SPHINCS+ [11]), code-based encryption (e.g., Classic McEliece [12,13]), and multivariate cryptography [14,15].