# UNIVERSIDAD NACIONAL “SIGO XX”

## DIRECCIÓN DE POSTGRADO

## DIPLOMADO EN REDES Y SEGURIDAD INFORMÁTICA


# ESTÁNDAR DE ENCRIPTACIÓN AVANZADA AES RIJNDAEL ACELERADO MEDIANTE UNIDAD DE PROCESAMIENTO GRÁFICO GPU

TRABAJO DE GRADO

**AUTOR: DANIEL JIMENEZ JEREZ**

**La Paz - Bolivia**
**2018**

# Índice general

# Índice de figuras

# Índice de cuadros

# Índice de Anexos

# Capítulo 1
# Introducción

En este capítulo se muestran: el panorama general del problema que se desea solucionar, la importancia teórica y práctica del desarrollo de la solución, el método empleado en el trabajo y el alcance que tiene la solución desarrollada.

## 1.1. Problemática

El paradigma computacional ha cambiado bastante desde la comercialización de la computadora personal (PC). De acuerdo a la ley de Moore: “el número de componentes de un circuito integrado seguirá doblándose cada año, y en 1975 serán mil veces más complejos que en 1965” [Moore, 1965, p. 2].

Sin embargo actualmente, con la popularidad que alcanzaron los conceptos de Dispositivos Inteligentes (Smart Devices), Dispositivos Portátiles (Wearables) y el Internet de las Cosas (IOT), se observa claramente que los circuitos integrados han llegado a un límite de tamaño tan pequeño que es muy difícil de reducir desde hace algunos años.

Para solventar tal limitación es que los fabricantes de microprocesadores cambiaron el paradigma de desarrollo de hardware de la arquitectura mono-núcleo a la arquitectura multi-núcleo. Posterior a este cambio se pudo observar que el límite del número de procesadores de uso general ha llegado también a un límite difícil de superar debido al calentamiento al que son sometidos los circuitos integrados dentro de dichos procesadores ya que la temperatura es inversamente proporcional al tamaño de los dispositivos y al trabajo de cómputo que se designa a cada circuito; por ello muchas aplicaciones actuales hacen uso de recursos definidos para tareas específicas, por ejemplo

Cuadro 1: Aplicaciones del algoritmo AES

| USO | APLICACIONES |
|---|---|
| Compresiòn de datos | 7z, Amanda Backup, PeaZip, PKZIP, RAR, WinZip, UltraISO |
| Encriptación de archivos | Gpg4win, Ncrypt |
| Encriptación de particiones de disco duro | NTFS, BTRFS |
| Encriptación de discos duros | BitLocker, CipherShed, DiskCryptor, FileVault, GBDE, Geli, LibreCrypt, LUKS, Private Disk, VeraCrypt |
| Seguridad en comunicaciones LAN | CCMP, ITU-T G.hn, IPsec |
| Seguridad en comunicaciones en Internet | GPG, TLS, SSL |
| Otras aplicaciones | KeePass Password Safe, Pidgin, Google Allo, Facebook Messenger, WhatsApp |

**Fuente:** Implementaciones AES, Aplicaciones, Wikipedia

para el cómputo de bloques muy grandes de imágenes o videos, se utilizan tarjetas gráficas que procesan volúmenes gigantescos de datos para entregar el resultado nuevamente al procesador central o bien para mostrar el resultado por pantalla u otro dispositivo de salida.

Debido a la carga de tareas que se asigna a la CPU[1], ésta puede llegar a formar colas insostenibles de procesos, por lo tanto, se intentará demostrar la factibilidad de la distribución de tareas a los procesadores de la Unidad de Procesamiento Gráfico (GPU[2]) para incrementar la velocidad de ejecución del algoritmo Estándar de Encriptación Avanzada (AES Rijndael).

## 1.2. Importancia teórica y práctica

El Estándar de Encriptación Avanzada (AES Rijndael) es utilizado en muchas aplicaciones actuales debido a su característica de patrón abierto para uso público y privado en aplicaciones personales y empresariales.

Cualquier incremento en la velocidad de ejecución de este algoritmo es de gran importancia práctica, ya que, como se muestra en el cuadro 1, los protocolos SSL y TLS trabajan con encriptación AES Rijndael, y es sabido que una gran parte de los servicios brindados en VPN y HTTPS para red local y/o internet transmiten grandes bloques de información enctriptada, por lo cual la ejecución de este proceso debe ser lo más rápida posible a fin de evitar latencia en las comunicaciones.

Por otra parte, en lo que respecta a la teoría de hilos y paralelismo de ejecución de procesos, la investigación del uso de tarjetas gráficas como unidades de procesamiento de datos es todavía un área joven sobre la cual se está comenzando a investigar, con el desarrollo de la tecnología TITAN de NVidia [3], misma que pone a disposición mallas de miles de procesadores Tensor y CUDA; dichos procesadores son núcleos de uso específico y no cuentan con las capacidades de los procesadores de uso general.

Entre los aspectos que caracterizan estas tecnologías de procesamiento están principalmente las operaciones matriciales, las operaciones de bucle independiente y las operaciones de transformación de datos.

## 1.3. Método de investigación

El método que de enfoque para el desarrollo del proyecto es el método

[1] Unidad Central de Procesamiento (Central Processing Unit)
[2] Graphics Processing Unit
[3] Tecnología NVidia Titan RTX

cuantitativo, ya que los objetivos se demuestran con tablas comparativas de resultados; empírico, ya que se ejecutaron programas de cómputo para llegar a los resultados; racionalista, ya que los resultados se toman en cuenta sin ninguna interpretación previa y positivista, ya que se desea demostrar el la aserción del incremento de velocidad de ejecución del algoritmo AES Rijndael en GPU con respecto a la velocidad de ejecución en CPU.

# Capítulo 2
# Justificación del problema

En este capítulo se muestran la justificación del problema con una visión desde diferentes ámbitos o enfoques de acuerdo a los paradigmas tecnológico y científico.

## 2.1. Justificación tecnológica

Como se mencionó anteriormente, de acuerdo al paradigma actual del desarrollo de hardware de microprocesadores, se llegó a un límite difícil de superar con los materiales de fabricación actuales.

Intel propuso el modelo de desarrollo de hardware “Tick-Tock” que consiste en un lanzamiento cada 18 meses (1 año y medio), ambas palabras hacen referencia a:

**Tick:** Mejoría de una arquitectura anterior
**Tock:** Lanzamiento de una nueva arquitectura

Pero si se realiza una comparación del último “Tock” que realizó Intel con respecto al microprocesador de la gama i7, se obtiene un resultado claro, y es que en 6 años se redujo el tamaño de transistor de 22nm a 14nm, duplicando los procesadores para llegar a un total de 8 procesadores físicos del CPU Intel i7-3770 del año 2012 al procesador Intel i9-9900 del año 2018. Pero la frecuencia se incrementó tan solo en 2MHz, por lo tanto se puede concluir que el paradigma tiene como objetivo llegar a multiplicar el número de procesadores pero no así la frecuencia de trabajo de los mismos.

Cuadro 2: Comparación procesadores Intel i7

| **Característica** | **I9-9900K** | **I7-3770** |
|---|---|---|
| Núcleos | 8 | 4 |
| Hilos | 16 | 8 |
| Serie | Coffee Lake | Ivy Bridge |
| Socket | FCLGA1151 | FCLGA1155 |
| Fecha de lanzamiento | 4º trimestre de 2018 | 2º trimestre de 2012 |
| Cache | 16 MB | 8 MB |
| Set de instrucciones | SSE4.1, SSE4.2, AVX2 | SSE4.1/4.2, AVX |
| Litografía | 14 nm | 22 nm |
| Velocidad de bus | 8 GT/s DMI3 | 5 GT/s DMI |
| Solución térmica | PCG 2015D (130W) | 2011D |
| Máximo tamaño de memoria | 64 GB | 32 GB |
| Tipo de memoria | DDR4-2666 | DDR3 1333/1600 |
| Ancho de banda de memoria | 41.6 GB/s | 25.6 GB/s |
| Frecuencia base para gráficos | 350 MHz | 650 MHz |
| Frecuencia máxima para gráficos | 1.20 GHz | 1.15 GHz |
| Configuración de PCI Express | 1x16, 2x8, 1x8+2x4 | 1x16, 2x8, 1x8 & 2x4 |

**Fuente:** UserBenchmark, 2018

Por lo tanto se justifica el uso de la Unidad de Procesamiento Gráfico para la distribución de tareas repetitivas aptas para un enfoque de ejecución en paralelo, ya que de acuerdo a la tabla comparativa anterior se observa que el paradigma de desarrollo de hardware por parte de los fabricantes es hacia un ecosistema de procesadores en paralelo en lugar de un bajo número de procesadores trabajando a frecuencias altas.

## 2.2. Justificación científica

En lo referente al ámbito científico, la utilidad de esta investigación radica en la profundización del estudio acerca de los métodos de ejecución de aplicaciones en múltiples hilos utilizando la Unidad de Procesamiento Gráfico (GPU), con la finalidad de llegar a utilizar las mallas de procesadores disponibles en las tarjetas gráficas, este estudio a la fecha de realización de la presente investigación, aún no se ha profundizado, ni es de aplicación general.

Por tanto, al concluir esta investigación, se habrán implantado los conocimientos necesarios para lograr aplicar métodos de ejecución de tareas en paralelo en la GPU utilizados en otros métodos de encriptación y otros procesos no necesariamente relacionados con criptografía.

# Capítulo 3
# Problema de investigación

En este capítulo se muestra en detalle la problemática de la investigación.

## 3.1. Determinación del problema

En el contexto de la computación, los procesos a ejecutarse ingresan a una cola de espera que dependiendo del trabajo del procesador, estos procesos pueden tardar un tiempo considerable en ser atendidos, e inclusive desecharse por la excesiva carga de trabajo de la CPU. En las comunicaciones encriptadas, dada la lógica inherente de los procesos de encriptación, los tiempos de ejecución pueden llegar a incrementarse hasta un punto insostenible para un solo procesador. Por lo cual los fabricantes vieron necesario el incremento del número de procesadores a fin de atender los procesos en la cola y evitar desechar tareas por los tiempos de ejecución.

Desde otro punto de vista, se cuenta con otra alternativa de procesamiento descentralizado mediante la delegación de trabajos a la GPU. Por tanto se verificará la reducción del tiempo de ejecución del algoritmo de encriptación AES Rijndael mediante la delegación de procesos a la GPU haciendo uso del entorno de programación Python, que cuenta con compatibilidad nativa para la ejecución de tareas en matrices de procesadores CUDA mediante la librería Numba[1].

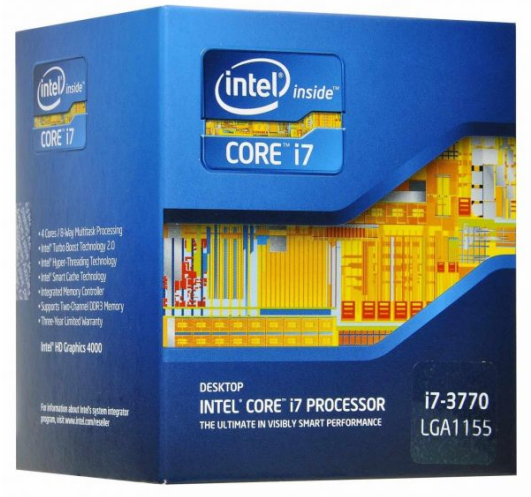
[1] NumPy+Mamba=Numba: Array-oriented Python Compiler for NumPy

## 3.2. Límites y alcances

El algoritmo implementado y modificado para la obtención de resultados en CPU y GPU, cumple con la definición de la Publicación de Estándares de Procesamiento de Información Federal 197[2]. [Information, 2001]. Dicha publicación fué aprobada por el Instituto Nacional de Estándares y Tecnología[3] después de la verificación en la Reforma de Administración de Tecnologías de la Información[4] de 1996.

El relleno de datos de 128 bits cumple con el Estándar de Criptografía de Llave Pública PKCS #7 plasmado en el reporte RFC 2315 [Kaliski, 1998].

Figura 1: Microprocesador Intel i7-3770




**Fuente:** Intel® Core™ i7-3770 Processor, ark.intel.com

En cuanto al hardware, se realiza la comparación de tiempo de ejecución en el microprocesador Intel i7-3770[5] con respecto a la tarjeta gráfica NVidia GTX 650 Ti[6]. Lo que representa una comparativa de trabajo simultáneo de 8 núcleos trabajando a 3.4GHz versus 768 núcleos trabajando a 928MHz para las operaciones pasibles a paralelismo.

[2] Federal Information Processing Standards Publications - FIPS 197
[3] National Institute of Standards and Technology - NIST
[4] Information Technology Management Reform
[5] Intel® Core™ i7-3770 Processor, ark.intel.com
[6] NVidia GeForce GTX-650Ti, www.geforce.com

El algoritmo utilizado modificado para esta investigación fué escrito por Pablo Caro y es de código abierto, se puede encontrar en la plataforma Github[7].

Figura 2: Tárjeta Gráfica NVidia GeForce GTX-650Ti


**Fuente:** NVidia GeForce GTX-650Ti, www.geforce.com

Se realizaron modificaciones a este algoritmo mediante la librería Numba que cuenta con decoradores predefinidos que compilan el código a ser paralelizado previa ejecución del programa. Esta librería genera kernels compatibles con plataformas CUDA de distintas arquitecturas.

El sistema operativo utilizado para la investigación es Arch Linux con el Kernel versión 4.18.10.

El driver de NVidia que se utilizó es de la versión 410.57-2 con el manejador de procesadores CUDA versión 10.0.130-2.

La versión de Python utilizada fué 3.6.7, con las librerías externas Numba v0.41 y Numpy v1.15.1.

[7] Python-AES, Pablo Caro

# Capítulo 4
# Objetivos

En este capítulo se detalla el objetivo principal y los objetivos específicos.

## 4.1. Objetivo general

Demostrar el incremento de velocidad para la ejecución del algoritmo AES Rijndael en la Unidad de Procesamiento Gráfico (GPU) con respecto a la ejecución del algoritmo en la Unidad Central de Procesamiento (CPU).

## 4.2. Objetivos específicos

- Liberar a la CPU de trabajo computacional para de forma, ejecutar otros procesos en la cola designando tareas pasibles a paralelización y cálculos matriciales a la GPU
- Mediante la implementación de un programa destinado a la ejecución en la GPU, obtener la diferencia de tiempo de ejecución del algoritmo AES utilizando solo la CPU con respecto a la ejecución del algoritmo con la asistencia de la GPU
- Paralelizar los procesos del algoritmo AES Rijndael en el modo ECB haciendo uso del entorno Python enfocado hacia la tecnología de malla de procesadores CUDA

# Capítulo 5
# Marco teórico

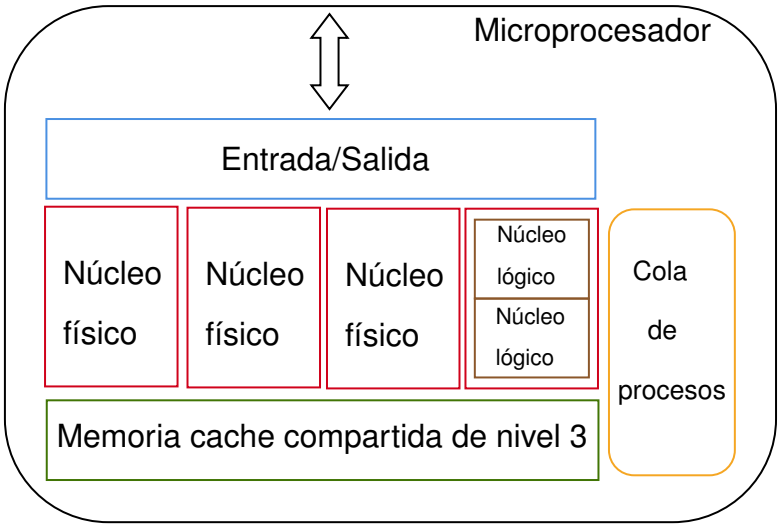
En este capítulo se muestra el marco teórico TODO.

## 5.1. Antecedentes

### 5.1.1. Computación paralela

La computación paralela es una rama de la informática que se encarga del estudio de la ejecución de una tarea dividida en sub-procesos o varias tareas independientes de forma simultánea en forma de hilos de ejecución en un grupo de procesadores llamados también procesadores multinúcleo, que luego de realizar dichas tareas sincronizan sus resultados a fin de mantener la integridad de los datos.

Figura 3: Disposición de un microprocesadores multinúcleo




**Fuente:** Elaboración propia

Los microprocesadores actuales contienen comúnmente dos tipos de núcleos,

los núcleos físicos y los núcleos lógicos. Cada zócalo de una tarjeta madre contiene un microprocesador, este contiene uno o más núcleos físicos; un núcleo físico es aquel que se encuentra físicamente dentro del circuito integrado del microprocesador, mientras que un núcleo lógico es una división virtual en 2 o más partes de un núcleo físico. Las tareas son asignadas a los núcleos físicos; estas tareas pueden dividirse en tareas más pequeñas a fin de resolver un gran problema en partes pequeñas que al final serán unidas para generar la solución, estas partes pequeñas son llamadas “hilos” y son las que se ejecutan en los núcleo lógicos.

*Las técnicas principales para lograr estas mejoras de rendimiento (mayor frecuencia de reloj y arquitecturas cada vez más inteligentes y complejas) están golpeando la llamada “Power Wall”. La industria informática ha aceptado que los futuros aumentos en rendimiento deben provenir en gran parte del incremento del número de procesadores (o núcleos) en una matriz, en vez de hacer más rápido un solo núcleo.* [Adve y col., 2008, p. 6]

El incremento de la frecuencia en los microprocesador acarrea consigo el consumo de energía y la disminución del espacio entre los transistores dentro de cada núcleo, lo que provoca un incremento considerable de la temperatura dentro del microprocesador; por tanto, para mantener el microprocesador en funcionamiento evitando su deterioro por las temperaturas elevadas es necesario buscar fuentes más óptimas de enfriado como los tubos de conducción de gas o líquido, que incrementan aún más el consumo de energía y que son costosos para una PC de escritorio.

$$T_m = T_a \cdot [(1 - F_m) + \frac{F_m}{A_m}] \tag{5.1}$$

Donde:

$F_m =$ Fracción de tiempo que el sistema utiliza el subsistema mejorado

$A_m =$ Factor de mejora que se ha introducido en el subsistema mejorado

$T_a =$ Tiempo de ejecución antiguo

$T_m =$ Tiempo de ejecución mejorado

Por tales motivos Gene Amdahl formuló la ecuación 5.1 que establece que:

*La mejora obtenida en el rendimiento de un sistema debido a la alteración de uno de sus componentes está limitada por la fracción de tiempo que se utiliza dicho componente*

Despejando la ecuación 5.1 se obtiene la aceleración del programa completo una vez que se haya paralelizado uno o más algoritmos del programa.

$$A = \frac{1}{(1 - F_m) + \frac{F_m}{A_m}} \tag{5.2}$$

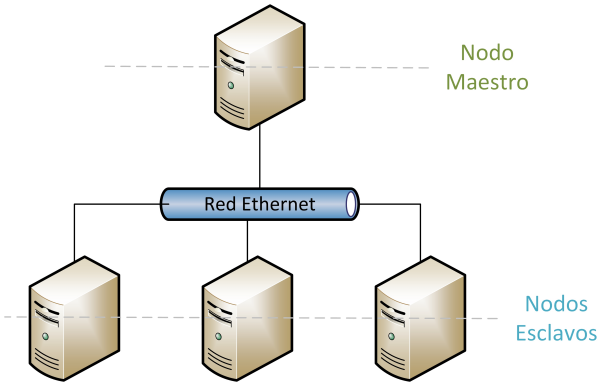
Donde:

$A =$ Aceleración o ganancia en velocidad conseguida en el sistema completo debido a la mejora de uno de sus subsistemas

$A_m =$ Factor de mejora que se ha introducido en el subsistema mejorado

$F_m =$ Fracción de tiempo que el sistema utiliza el subsistema mejorado

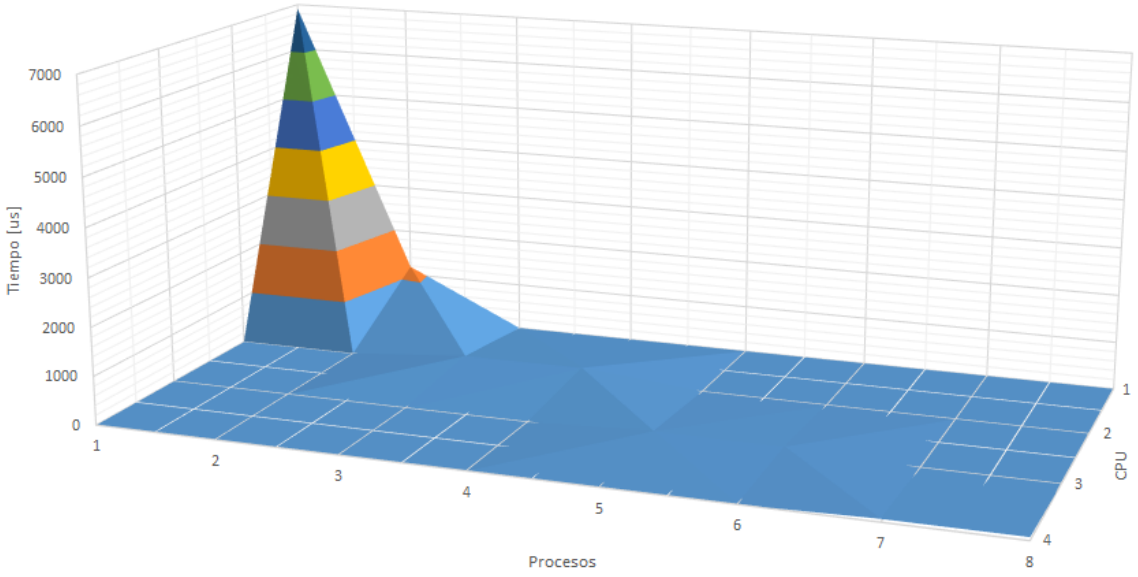
Figura 4: Clúster de alto rendimiento




**Fuente:** Jimenez y Medina, 2014, p. 2

En base a este principio se desarrollaron tecnologías de matrices de núcleos de cómputo tomando como elementos principales a los procesadores existentes y acomodándolos de tal forma que se pueda administrar la ejecución de tareas en cada procesador de manera individual y la sincronización de resultados al final del proceso. Estos arreglos reciben el nombre de clústers. Los clústers de alto rendimiento son un tipo de clústers utilizados con el propósito de ejecutar tareas exhaustivas divididas en tareas pequeñas ejecutadas en cada computador de acuerdo a la gestión realizada por el llamado nodo maestro. [Jimenez y Medina, 2014]

Figura 5: Comparación de tiempos de proceso en múltiples CPUs




**Fuente:** Jimenez y Medina, 2014, p. 7

### 5.1.2. Unidad de procesamiento gráfico (GPU)

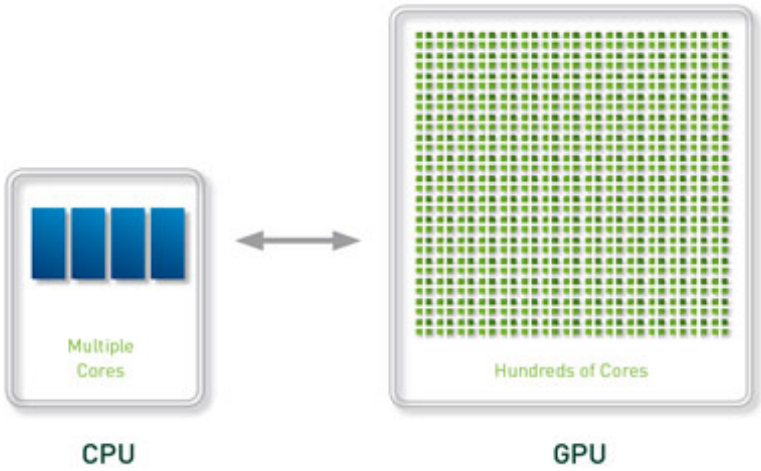
Esta unidad actúa como un co-procesador que se encarga de las operaciones matriciales o de coma flotante, por lo general los procesos gráficos de transformación o renderización son distribuidos a la o las GPUs desde el procesador central o CPU.

Dado el estudio generado sobre las plataformas GPU, los fabricantes pusieron a

disposición de los usuarios herramientas de desarrollo para utilizar las GPU como ayuda en cálculos de álgebra dispersa, tensores en dinámica de fluidos, minería de datos, inteligencia artificial, deep learning, etc, con lo cual la denominación de las GPU abiertas a otro tipo de uso más que el simple uso gráfico cambió a GPGPU[1].

Estas tarjetas están desarrolladas en base al paralelismo de núcleos de frecuencia baja con un esquema de operaciones limitado.

Figura 6: Cantidad de núcleos en CPU vs GPU



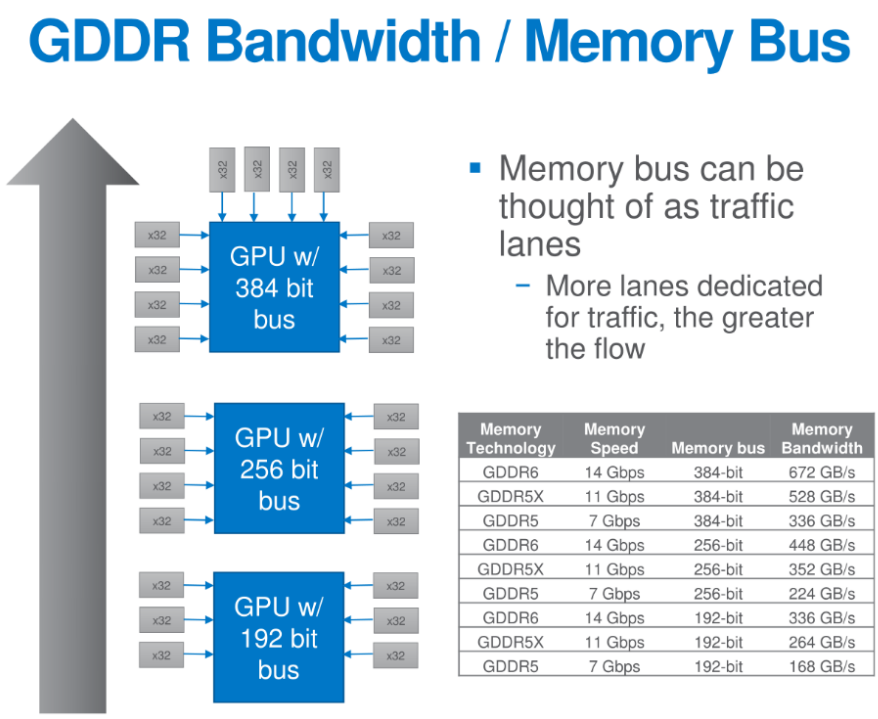

**Fuente:** SuperComputing Applications and Innovation, 2012

El obstáculo principal para el desarrollo de aplicaciones orientadas hacia la GPU es que las arquitecturas de las tarjetas gráficas son demasiado variables, a pesar de la existencia de librerías o APIs genéricas como OpenGL, muchas funcionalidades dentro de los métodos o clases son variables entre fabricantes e incluso entre modelos de dispositivos de un mismo fabricante. Las librerías genéricas utilizan un núcleo basado en el esquema de Conductos de Renderización[2], con los que se pueden tratar vectores, mapas de bits y elementos definidos pixel-pixel.

[1] Unidad de Procesamiento Gráfico de Uso General (General Purpose Graphics Processing Unit)
[2] Rendering Pipeline

Otro factor importante que impide hacer un uso adecuado de estos dispositivos es el límite físico con el que actualmente cuenta la conexión de memoria RAM de la GPU con el bus de la CPU para la transferencia de datos. Al cuarto trimestre de 2018 ya se cuenta con la tecnología GDDR6[3] que ofrece un ancho de banda de hasta 16Gbps frente a los 10Gbps de su predecesor GDDR5X, cabe mencionar que se lograron estos anchos de banda gracias al cambio de modo half-duplex o transferencia en ambos sentidos pero solo uno a la vez, por el modo full-duplex que transfiere los datos en ambos sentidos al mismo tiempo.

Figura 7: Comparación de tecnologías GDDR6 vs GDDR5




| Memory Technology | Memory Speed | Memory bus | Memory Bandwidth |
|---|---|---|---|
| GDDR6 | 14 Gbps | 384-bit | 672 GB/s |
| GDDR5X | 11 Gbps | 384-bit | 528 GB/s |
| GDDR5 | 7 Gbps | 384-bit | 336 GB/s |
| GDDR6 | 14 Gbps | 256-bit | 448 GB/s |
| GDDR5X | 11 Gbps | 256-bit | 352 GB/s |
| GDDR5 | 7 Gbps | 256-bit | 224 GB/s |
| GDDR6 | 14 Gbps | 192-bit | 336 GB/s |
| GDDR5X | 11 Gbps | 192-bit | 264 GB/s |
| GDDR5 | 7 Gbps | 192-bit | 168 GB/s |

**Fuente:** ExtremeTech, 2018

Pero AMD ya se encuentra desarrollando tarjetas madres con conectores PCI-E[4] 5.0 que incrementarán la velocidad de transferencia hasta los 32Gbps que conjuntamente con el almacenamiento SSD[5] lograrán impulsar el desarrollo de aplicaciones de uso general en las GPUs.

[3] Tasa Doble de transferencia de Datos (Double Data Rate)
[4] Componente Periférico de Interconexión Expresa (Peripheral Component Interconnect Express)
[5] Solid State Drive

Cuadro 3: Comparación de tecnologías PCI-E

| | **RAW Bitrate** | **Link BW** | **BW/Lane/Way** | **Total BW X16** |
|---|---|---|---|---|
| **PCIe 1.x** | 2.5 GT/s | 2 Gb/s | 250 MB/s | 8 GB/s |
| **PCIe 2.x** | 5.0 GT/s | 4 Gb/s | 500 MB/s | 16 GB/s |
| **PCIe 3.x** | 8.0 GT/s | 8 Gb/s | $\sim$1 GB/s | $\sim$32 GB/s |
| **PCIe 4.x** | 16 GT/s | 16 Gb/s | $\sim$2 GB/s | $\sim$64 GB/s |
| **PCIe 5.x** | 32 GT/s | 32 Gb/s | $\sim$4 GB/s | $\sim$128 GB/s |

**Fuente:** Smith, 2018

## 5.2. AES

El Estándar de Encriptación Avanzada fue de desarrollado mediante un concurso en 1997, por los criptógrafos Vincent Rijmen e Joan Daemen en el año 2001, como la sustitución al algoritmo DES[6] que había sido crackeado mediante la máquina DES Cracker construida por la ONG Electronic Frontier Foundation, con una inversión de 250 mil dólares. Este estándar fue aprobado y es utilizado por entes reguladores como la NSA[7] y se estandariza mediante la norma ISO/IEC 18033 [ISO/IEC JTC 1, Information technology, Subcommittee SC 27, Security techniques, 2015].

El algoritmo AES Rijndael es un algoritmo de llave simétrica, lo cual indica que se utiliza una misma llave para cifrar y descifrar los mensajes en el lado del emisor y del receptor. Por tal razón, toda la seguridad recae en proteger la clave secreta, por tal razón el abanico de claves posibles debe ser de una cantidad tan grande que el intruso deba realizar pruebas, por inclusive años, para poder descifrar el mensaje. Para el caso de DES, la clave es de 56 bits por lo que la cantidad de claves será igual a: $2^{56} = 7{,}2 \times 10^{16}$ posibles claves; un computador actual puede lograr descifrar un mensaje mediante el cálculo de la llave secreta en un tiempo

[6] Estándar de Encriptación de Datos(Data Encryption Standard)
[7] Agencia de Seguridad Nacional(National Security Agency)

de tan solo segundos.

Cuadro 4: Comparación de tecnologías PCI-E

| **Tamaño de Clave** | **Combinaciones Posibles** |
|---|---|
| 1 bit | $2$ |
| 2 bit | $4$ |
| 4 bit | $16$ |
| 8 bit | $256$ |
| 16 bit | $65536$ |
| 32 bit | $4{,}2 \times 10^9$ |
| 56 bit (DES) | $7{,}2 \times 10^{16}$ |
| 64 bit | $1{,}8 \times 10^{19}$ |
| 128 bit (AES) | $3{,}4 \times 10^{38}$ |
| 192 bit (AES) | $6{,}2 \times 10^{57}$ |
| 256 bit (AES) | $1{,}1 \times 10^{77}$ |

**Fuente:** DataQUBO, 2013

El algoritmo AES Rijndael trabaja con mensajes divididos en bloques de 128 bits y llaves de logitud de 128, 192 y 256 bits. Por lo tanto con una llave de 128 bits el atacante necesitaría generar: $2^{56} = 3{,}4 \times 10^{38}$ llaves, tarea que en la actualidad, aún con computadoras tan potentes, el trabajo tardaría millones de años.

Suponiendo la super-computadora Summit de IBM [IBM, 2018], designada para descifrar un mensaje, trabajando a $143{,}5PFlops$[8] o $143{,}5 \times 10^{15} Flops$ y sabiendo que la cantidad de segundos en un año es de: $365x24x60x60 = 31536000$. Se calcula la cantidad de años necesarios para crackear AES con una longitud de

[8] Operaciones de Punto Flotante por Segundo(Floating point Operations Per Second)

clave de 128 bits.

$$
\begin{aligned}
t &= \frac{3{,}4 \times 10^{38}}{143{,}5 \times 10^{15} x 31536000} \\
t &= \frac{23{,}69 \times 10^{15}}{315{,}36} \\
t &= 75{,}13 \times 10^{12} años
\end{aligned}
\tag{5.3}
$$

Es decir, con la última tecnología disponible actualmente se tomaría un tiempo de 75.13 billones de años en generar las llaves secretas necesarias para descifrar un mensaje. Suponiendo que solo fuese necesario generar la mitad de las llaves para encontrar la correcta, el proceso tardaría mas de 32 billones de años. Por lo tanto una llave secreta de 128 bits utilizada para cifrar un mensaje con el algoritmo AES Rijndael es suficiente seguridad para la actualidad y para unos años más en el futuro.

### 5.2.1. Estructura del algoritmo

Este algoritmo es conformado por rondas en las que se ejecutan 4 funciones matemáticas en un orden establecido. El resultado de cada ronda es llamado *Estado* que es una matriz de 4 filas por $N_b$ columnas, donde:

$$
N_b = \frac{Tamaño\ de\ bloque\ utilizado\ en\ bits}{32} \tag{5.4}
$$

De manera similar, la clave inicial se representa mediante una matriz de 4 filas y $N_k$ columnas, donde:

$$
N_k = \frac{Tamaño\ de\ la\ clave\ en\ bits}{32} \tag{5.5}
$$

Las matrices se acomodan de tal forma que cada palabra (4 bytes = 32 bits) es

representada en una columna de izquierda a derecha.

Por ejemplo la frase: “mensaje secreto.” se convierte de ASCII a su representación hexadecimal, cuyo resultado es:

6d 65 6e 73 61 6a 65 20 73 65 63 72 65 74 6f 2e

Que se acomoda en una matriz de estado como se muestra a continuación:

| | | | |
|---|---|---|---|
| 6d | 61 | 73 | 65 |
| 65 | 6a | 65 | 74 |
| 6e | 65 | 63 | 6f |
| 73 | 20 | 72 | 2e |

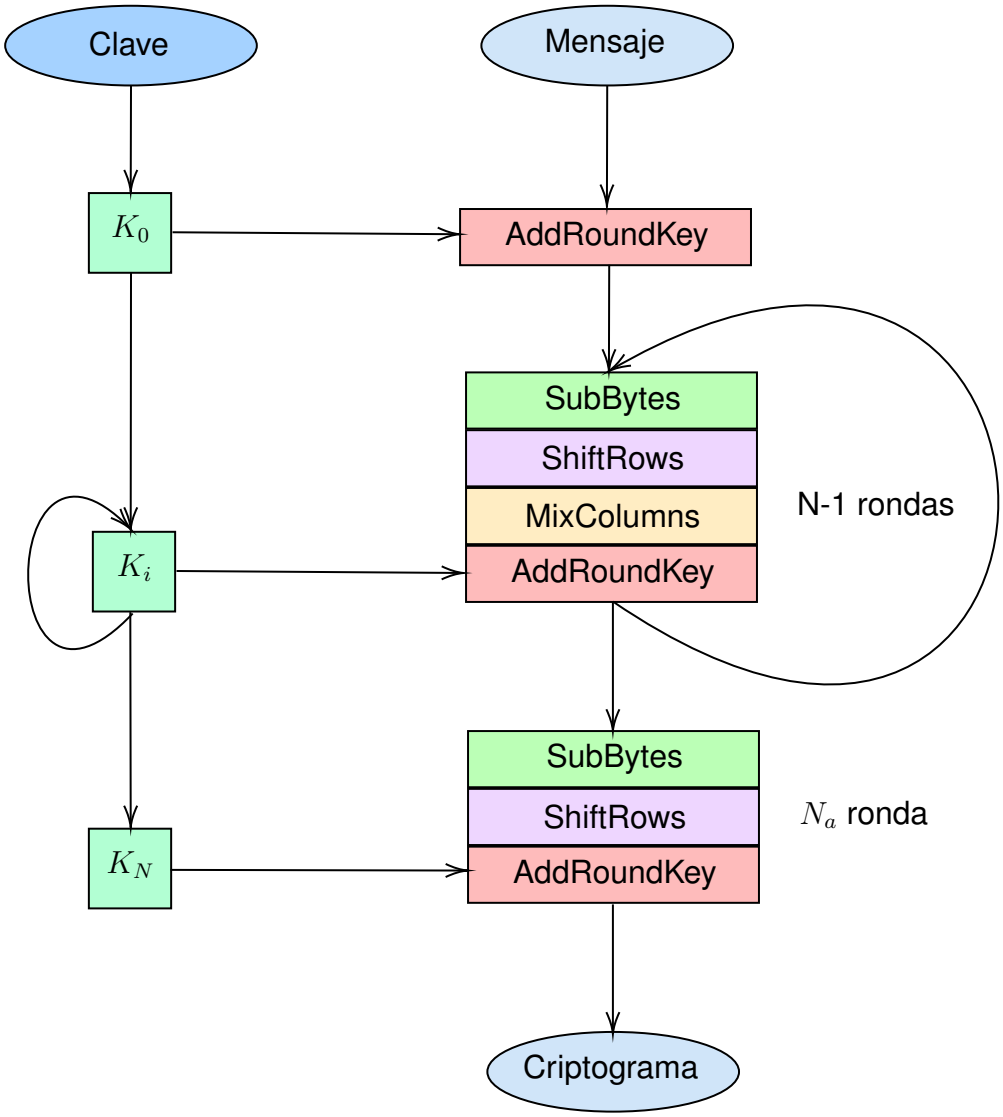
Cabe recalcar que el mensaje es de una longitud de 128 bits; en caso de no ser así, el mensaje es dividido en paquetes de 128 bits y se acomoda un relleno o padding de acuerdo a la norma PKCS#7 en el caso de AES Rijndael.

De forma análoga, se realiza el mismo proceso para la clave secreta inicial que puede ser de 128, 192 o 256 bits. El número de rondas para llegar a calcular el criptograma varía de acuerdo al tamaño de la clave:

- Para una clave de cifrado de 128 bits el algoritmo realiza 10 rondas
- Para una clave de cifrado de 192 bits el algoritmo realiza 12 rondas
- Para una clave de cifrado de 256 bits el algoritmo realiza 14 ronda

#### 5.2.1.1. Operaciones para el proceso de cifrado

Figura 8: Algoritmo AES Rijndael




**Fuente:** Elaboración propia

Donde:

**N:** Número de rondas

**i:** Ronda actual

$N_a$**:** Enésima ronda

1. **SubBytes**

   Es la sustitución byte a byte de la matriz de estado mediante la tabla S-Box que se construye mediante dos transformaciones consecutivas. Cada byte se considera un elemento del Campo de Galois $GF(2^8)$ de cuerpos finitos; esto quiere decir que para cada valor existe un inverso aditivo y multiplicativo que elimina los problemas de redondeo y desbordamiento. Cabe mencionar que en la matemática modular se trabaja únicamente con números primos. Por tanto cada byte genera un polinomio irreducible $m(x) = x^8+x^4+x^3+x+1$, que es sustituido por su inverso multiplicativo. Una vez realizada la primera transformación se aplica la transformación afín en GF(2):

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

   De acuerdo a esta lógica se calcula la tabla S-Box para cada valor de los 256 que pueden componer un byte (8 bits).

Cuadro 5: Tabla S-Box

| HEX | | y | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **00** | **01** | **02** | **03** | **04** | **05** | **06** | **07** | **08** | **09** | **0A** | **0B** | **0C** | **0D** | **0E** | **0F** |
| x | **00** | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 01 | 67 | 2B | FE | D7 | AB | 76 |
| | **10** | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | 72 | C0 |
| | **20** | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | 71 | D8 | 31 | 15 |
| | **30** | 04 | C7 | 23 | C3 | 18 | 96 | 05 | 9A | 07 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| | **40** | 09 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | B3 | 29 | E3 | 2F | 84 |
| | **50** | 53 | D1 | 00 | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| | **60** | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 02 | 7F | 50 | 3C | 9F | A8 |
| | **70** | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| | **80** | CD | 0C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| | **90** | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | 0B | DB |
| | **A0** | E0 | 32 | 3A | 0A | 49 | 06 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| | **B0** | E7 | C8 | 37 | 6D | 8D | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 08 |
| | **C0** | BA | 78 | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 8A |
| | **D0** | 70 | 3E | B5 | 66 | 48 | 03 | F6 | 0E | 61 | 35 | 57 | B9 | 86 | C1 | 1D | 9E |
| | **E0** | E1 | F8 | 98 | 11 | 69 | D9 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 55 | 28 | DF |
| | **F0** | 8C | A1 | 89 | 0D | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | B0 | 54 | BB | 16 |

**Fuente:** Muñoz, 2004

2. **ShiftRows**

En esta transformación se rotan hacia la izquierda las filas de la matriz de estado de manera incremental; es decir, la primera fila no rota, la segunda rota 1 vez, la tercera 2 veces y la cuarta 3 veces. El número de rotaciones se mantiene constante para la matriz de estado conformada por el mensaje pero varía para la clave ya que esta puede ser de 128, 192 o 256 bits como se mencionó anteriormente, para ello se aplica la siguiente regla, tomando $C_0$ como la primera fila de la matriz de estado, $C_1$ como la segunda, $C_2$ como la tercera u $C_3$ como la cuarta, se mantiene $C_0$ sin rotar, las demás filas rotan de acuerdo a la tabla 6.

Cuadro 6: Rotaciones de las filas de la matriz de estado

| Tamaño de bloque | C1 | C2 | C3 |
|---|---|---|---|
| **128 bits ($N_b = 4$)** | 1 | 2 | 3 |
| **192 bits ($N_b = 6$)** | 1 | 2 | 3 |
| **256 bits ($N_b = 8$)** | 1 | 3 | 4 |

**Fuente:** Muñoz, 2004

Un ejemplo gráfico de la rotación de la matriz de estado se puede observar en la table 7.

Cuadro 7: ShiftRows

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $S_{0,0}$ | $S_{0,1}$ | $S_{0,2}$ | $S_{0,3}$ | => | $S_{0,0}$ | $S_{0,1}$ | $S_{0,2}$ | $S_{0,3}$ |
| $S_{1,0}$ | $S_{1,1}$ | $S_{1,2}$ | $S_{1,3}$ | | $S_{1,1}$ | $S_{1,2}$ | $S_{1,3}$ | $S_{1,0}$ |
| $S_{2,0}$ | $S_{2,1}$ | $S_{2,2}$ | $S_{2,3}$ | | $S_{2,2}$ | $S_{2,3}$ | $S_{2,0}$ | $S_{2,1}$ |
| $S_{3,0}$ | $S_{3,1}$ | $S_{3,2}$ | $S_{3,3}$ | | $S_{3,3}$ | $S_{3,0}$ | $S_{3,1}$ | $S_{3,2}$ |

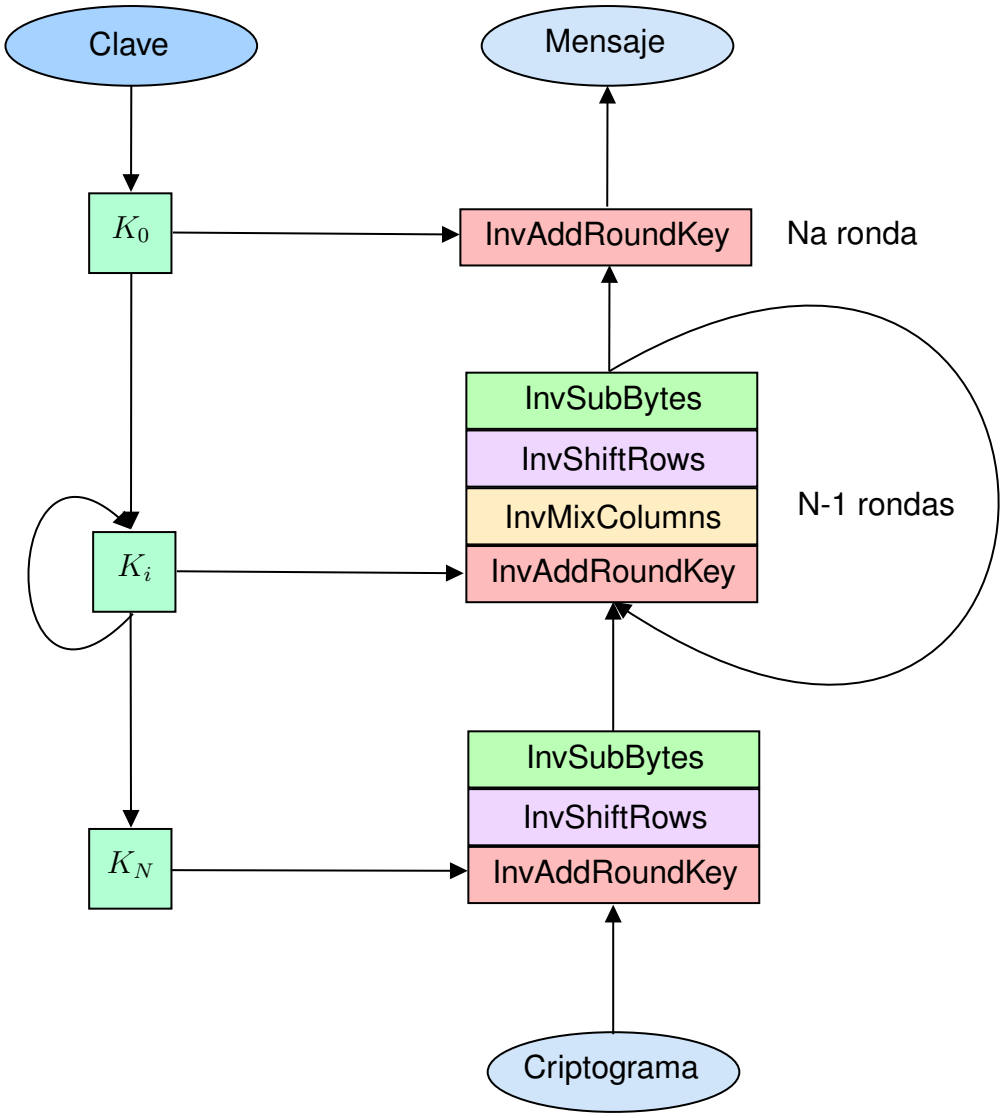
**Fuente:** Muñoz, 2004

### 3. MixColumns

El número de sub-claves a calcular es igual a N, ya que la primera operación a realizar es siempre un OR exclusivo entre la clave inicial y la primera matriz de estado, en este caso del texto en claro acomodado en la matriz de dimensión 4x4.

El proceso de descifrado es el mismo pero en orden contrario:

Figura 9: Algoritmo AES Rijndael




**Fuente:** Elaboración propia

#### 5.2.1.2. Expansión de clave

Se toma la clave inicial como la primera sub-clave $K_0$, por ejemplo:

63 6C 61 76 65 20 64 65 20 31 32 38 62 69 74 73

1. **RotWord**

Se selecciona la última columna o palabra de la sub-clave y se rota el byte superior de manera vertical.

| | | | | | |
|---|---|---|---|---|---|
| 63 | 65 | 20 | 62 | | 69 |
| 6C | 20 | 31 | 69 | | 74 |
| 61 | 64 | 32 | 74 | => | 73 |
| 76 | 65 | 38 | 73 | | 62 |

2. **SubBytes**

Se sustituyen los valores de acuerdo a la tabla de sustitución S-Box de Rijndael.

| | | |
|---|---|---|
| 69 | | F9 |
| 74 | | 92 |
| 73 | => | 8F |
| 62 | | AA |

3. **XOR [i-3]**

Se realiza la operación de OR exclusivo con la columna que se encuentra 3 posiciones atrás en la matriz de estado de la sub-clave.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 63 | 65 | 20 | 62 | | F9 | | 63 | | 9A |
| 6C | 20 | 31 | 69 | | 92 | | 6C | | FE |
| 61 | 64 | 32 | 74 | => | 8F | $\oplus$ | 61 | = | EE |
| 76 | 65 | 38 | 73 | | AA | | 76 | | DC |

# Capítulo 6
# Conclusiones

En este capítulo se muestran las conclusiones TODO.

## 6.1. Conclusiones

## Bibliografía


Adve, S. V., Adve, V. S., Agha, G., Frank, M. I., Garzarán, M. J., Hart, J. C., ... Zilles, C. (2008). Parallel Computing Research at Illinois, The UPCRC Agenda. University of Illinois at Urbana-Champaign. Recuperado desde http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.471.2755&rep=rep1&type=pdf

Daemen Joan, R. V. (2015). AES implementations. 10 de Diciembre de 2018. Recuperado desde https://en.wikipedia.org/wiki/AES_implementations

Daemen, J. & Rijmen, V. (2002). *The Design of RijndaeL: AES - The Advanced Encryption Standard (Information Security and Cryptography)*. Springer. Recuperado desde https://www.amazon.com/Design-RijndaeL-Encryption-Information - Cryptography / dp / 3540425802 ? SubscriptionId = AKIAIOBINVZYXZQZ2U3A & tag = chimbori05 - 20 & linkCode = xm2 & camp = 2025&creative=165953&creativeASIN=3540425802

DataQUBO. (2013). ¿Qué tan seguro es AES? 17 de Diciembre de 2018. Recuperado desde https://www.top500.org/system/179397

ExtremeTech. (2018). PCIe 5.0 Arriving in 2019 With 4x More Bandwidth Than PCIe 3.0. 16 de Diciembre de 2018. Recuperado desde https://www.extremetech.com/computing/250640-pci-sig-announces-plans-launch-pcie-5-0-2019-4x-bandwidth-pcie-3-0

IBM. (2018). Summit Super-Computer. 17 de Diciembre de 2018. Recuperado desde http://www.dataqubo.com/encriptacion-que-tan-seguro-es-aes/

Information, F. (2001). *Advanced Encryption Standard (AES)*.

ISO/IEC JTC 1, Information technology, Subcommittee SC 27, Security techniques. (2015). ISO/IEC 18033-1:2015. Recuperado desde https://www.iso.org/obp/ui/#iso:std:iso-iec:18033:-1:ed-2:v1:en

Jimenez, D. & Medina, A. (2014). Cluster de Alto Rendimiento. *Instituto de Electrónica Aplicada, Boletín Anual 2014, Universidad Mayor de San*

*Andrés*. https://www.scribd.com/document/376915872/Cluster-Alto-Rendimiento.

Kaliski, B. (1998). *PKCS #7: Cryptographic Message Syntax Version 1.5*. RFC. http://www.rfc-editor.org/rfc/rfc2315.txt. doi:10.17487/rfc2315

Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, *38*(8), 5.

Muñoz, A. M. (2004). *Seguridad Europea para EEUU. Algoritmo Criptográfico Rijndael*. Recuperado desde http://www.tierradelazaro.com/wp-content/uploads/2016/04/AES.pdf

Smith, R. (2018). Micron begins mass production of GDDR6. 16 de Diciembre de 2018. Recuperado desde https://www.anandtech.com/show/13012/micron-begins-mass-production-of-gddr6

SuperComputing Applications and Innovation. (2012). GPGPU (General Purpose Graphics Processing Unit). 13 de Diciembre de 2018. Recuperado desde http : / / www . hpc . cineca . it / content / gpgpu - general - purpose - graphics - processing-unit

UserBenchmark. (2018). i9-9900k vs i7-3770. 10 de Diciembre de 2018. Recuperado desde https://cpu.userbenchmark.com/Compare/Intel-Core-i9-9900K-vs-Intel-Core-i7-3770/4028vs1979