Cifrado homomórfico como servicio para tareas de minería de datos con preservación de privacidad

Shanel Reyes-Palacios
Cinvestav Tamaulipas
Ciudad Victoria, Tamps, Mexico
shanel.reyes@cinvestav.mx

Miguel Morales-Sandoval
Ciencias Computacionales, INAOE
Tonantzintla, Pue, Mexico
mmorales@inaoep.mx

Jose Juan Garcia-Hernandez
Cinvestav Tamaulipas
Ciudad Victoria, Tamps, Mexico
jjuan.garcia@cinvestav.mx

J.L. Gonzalez-Compean
Cinvestav Tamaulipas
Ciudad Victoria, Tamps, Mexico
joseluis.gonzalez@cinvestav.mx

Heidy M Marin-Castro
UPIIAP-IPN
Puebla, Pue, Mexico
heidy.marinc@gmail.com

***Abstract*—El cifrado homomórfico es una técnica que ha permitido desarrollar métodos de minería de datos (clasificación, agrupamiento) con preservación de privacidad (MDPP), al realizar operaciones directamente sobre datos cifrados, preservando la confidencialidad de la información. Sin embargo, su uso práctico sigue estando limitado por el elevado costo computacional que implica su ejecución. Este trabajo presenta el diseño y arquitectura de un servicio de cifrado homomórfico, validado con los esquemas de cifrado homomórfico mayormente usados en MDPP: Liu, CKKS y Paillier. El diseño puede soportar cualquier otro esquema de cifrado, incorpora patrones de paralelismo y esquemas de distribución de carga de trabajo. Estas características le permiten al servicio mitigar la sobrecarga asociada al costo computacional que implica la ejecución de los esquemas de cifrado homomórfico. Realizamos una evaluación con distintos tamaños de conjuntos de datos en el contexto de MDPP. Se analizó el rendimiento y la escalabilidad del servicio y se verificó la conservación de la utilidad en tareas de agrupamiento. Los resultados muestran que CKKS logró reducciones de tiempo de hasta el 88% frente a la ejecución secuencial, mientras que Liu y Paillier alcanzaron reducciones entre el 60% y el 70%, sin comprometer la calidad de los datos, lo que posiciona al servicio como una solución factible para MDPP.**

***Index Terms*—Minería de datos como servicio, cifrado homofórfico, escalabilidad, agrupamiento.**

I. Introducción

El cifrado homomórfico (Homomorphic Encryption, por sus siglas en inglés) permite realizar operaciones directamente sobre datos cifrados sin necesidad de descifrarlos, garantizando la privacidad en escenarios como cómputo delegado a terceros [1], aprendizaje automático seguro [2] y minería de datos con preservación de la privacidad (MDPP) [3]. No obstante, a pesar de su potencial, su elevado costo computacional limita su aplicación práctica en entornos de big data, donde los volúmenes de datos y la complejidad de las tareas pueden provocar tiempos de procesamiento inaceptables [4], [5]. Dado que muchas tareas de mineria de datos como clustering y clasificación, requieren preservar la privacidad de los datos, es crucial mejorar los tiempos de cifrado sin sacrificar utilidad. Por ello, en este trabajo validamos nuestro servicio con los tres esquemas más comunes en MDPP: Liu, CKKS y Paillier, cada uno ofreciendo diferentes compromisos entre funcionalidad y costo computacional.

Liu [6] es un esquema simétrico completamente homomórfico que admite suma, multiplicación, resta y multiplicación por escalar con costo lineal en el tamaño de la clave y del texto cifrado. CKKS [7] implementa un cifrado aproximado sobre números reales codificados en polinomios, ideal para aplicaciones de aprendizaje automático tolerantes a ligeras imprecisiones. Paillier [8] es un esquema asimétrico parcialmente homomórfico aditivo, mejorado para operaciones de suma y multiplicación por escalar sobre enteros con textos relativamente compactos.

Proponemos un servicio de cifrado homomórfico, validado con el algoritmo de Liu, CKKS y Paillier pero extensible a otros esquemas. A partir de la configuración del usuario, el servicio segmenta el dataset por filas con un tamaño configurable, asigna los segmentos a procesos según la carga, cifra cada bloque y consolida los resultados junto con sus metadatos. La interfaz abstrae el backend criptográfico, lo que facilita intercambiar o incorporar esquemas. Este flujo reduce los tiempos de cifrado y mantiene la estructura por registro requerida por algoritmos de ciencia de datos con preservación de privacidad, como en aquellos para agrupamiento y clasificación.

La minería de datos con preservación de la privacidad se compone de dos etapas: primero, la preparación de los datos para preservar la privacidad, por ejemplo, usando cifrado homomórfico, y a continuación, la aplicación del algoritmo de minería de datos (clustering o clasificación). Nuestra propuesta aborda esa primera etapa, donde se segmenta el conjunto de datos, se cifra cada segmento de manera paralela y finalmente cada uno se entregan a la segunda etapa sin necesidad de trabajo adicional.

Este servicio ha sido validado mediante una evaluación en la que se define un conjunto de datos de 100,000 registros por 100 atributos, se varía el número de procesos y el esquema de cifrado. Medimos rendimiento y escalabilidad, y verificamos que los resultados cifrados conservan su utilidad en tareas de

agrupamiento. Los resultados muestran reducciones significativas en los tiempos de cifrado sin sacrificar la privacidad, lo que posiciona este servicio como una solución práctica para análisis escalable con preservación de la privacidad.

El resto del artículo se organiza de la siguiente manera. En la Sección II presentamos los conceptos y notación básicos de HE. En la Sección III revisamos los trabajos relacionados con modos de ejecución local y paralelo en HE. La Sección IV describe el diseño e implementación de nuestro servicio. La Sección V detalla el entorno experimental, las métricas y los resultados de rendimiento y utilidad. Finalmente, la Sección VI concluye el artículo y plantea líneas de trabajo futuras.

II. Preliminares

En este trabajo empleamos esquemas de cifrado homomórfico, que permiten realizar operaciones aritméticas directamente sobre datos cifrados sin necesidad de descifrarlos. Para ello, definimos las operaciones homomórficas mediante los operadores:

$$E_3 = E_1 \oplus E_2, \qquad E_3 = E_1 \otimes E_2, \qquad E' = c \odot E_1,$$

donde E_1 y E_2 son textos cifrados, y c es un escalar en claro. Un esquema se describe con los algoritmos

$$CT \leftarrow \mathsf{Enc}\big(K_{\text{enc}}, M\big), \qquad M \leftarrow \mathsf{Dec}\big(K_{\text{dec}}, CT\big),$$

donde M es el mensaje en claro y CT su correspondiente texto cifrado. K_{enc} es la clave de cifrado y K_{dec} la clave de descifrado. Si $K_{\text{enc}} = K_{\text{dec}}$, el esquema es simétrico; en caso contrario, es asimétrico. Siempre debe cumplirse

$$\mathsf{Dec}\big(K_{\text{dec}},\, \mathsf{Enc}(K_{\text{enc}}, M)\big) \;=\; M$$

Los esquemas de cifrado homomórfico se distinguen por la cantidad y el tipo de operaciones que admiten antes de requerir un descifrado intermedio [9]. Los esquemas parcialmente homomórficos (PHE) soportan solo un tipo de operación, ya sea suma o multiplicación. Los esquemas algo homomórficos (SWHE) extienden esta capacidad permitiendo un número limitado de sumas y productos, mientras que los esquemas totalmente homomórficos (FHE) admiten un número arbitrario de ambas operaciones. En este artículo validamos el servicio con tres esquemas representativos de estas clases: Paillier (PHE aditivo), Liu (FHE simétrico con aritmética entera) y CKKS (FHE aproximado para datos decimales). Cada uno de estos esquemas se define mediante tres algoritmos básicos: 1) generación de llaves (`KeyGen`), 2) cifrado (`Encrypt`), y 3) descifrado (`Decrypt`). A continuación se presenta el pseudocodigo de cada esquema: el Algoritmo 1 resume `KeyGen`, `Encrypt` y `Decrypt` del esquema de Paillier, los Algoritmos 2 y 3 describen los mismos procedimientos para los esquemas de Liu y CKKS respectivamente, incluyendo la codificación/decodificación y el manejo de escala. Las operaciones homomórficas se expresan de forma uniforme con los operadores $\oplus$, $\otimes$ y $\odot$:

a) Paillier (PHE aditivo): Fundamentado en el problema del residuo cuadrático [8], con espacio de mensajes $\mathbb{Z}_n$.

Algorithm 1 Paillier — `KeyGen`, `Encrypt`, `Decrypt`

procedure KeyGen(k)
  Elegir primos p, q de k bits
  $n \leftarrow p \cdot q, \quad \lambda \leftarrow \mathrm{lcm}(p-1, q-1)$
  Elegir $g \in \mathbb{Z}^*_{n^2}$
  $\mu \leftarrow \big(L(g^\lambda \bmod n^2)\big)^{-1} \bmod n$, donde $L(u) = \frac{u-1}{n}$
  Salida: $K_{\text{enc}} = (n, g)$, $K_{\text{dec}} = (\lambda, \mu)$
end procedure
procedure Encrypt(K_{enc}, M)
  Elegir $r \in \mathbb{Z}^*_n$
  $CT \leftarrow g^M \cdot r^n \bmod n^2$
  Salida: CT
end procedure
procedure Decrypt(K_{dec}, CT)
  $M \leftarrow L\big(CT^\lambda \bmod n^2\big) \cdot \mu \bmod n$
  Salida: M
end procedure

Las operaciones homomórficas admitidas son $E_3 = E_1 \otimes E_2 = v_1 \cdot v_2 \bmod n^2$, $\quad E' = c \odot E = E^c \bmod n^2$, (donde v es un plaintext y c es un escalar en claro), lo que permite implementaciones de suma, multiplicación por escalar y resta cifrada.

b) Liu (FHE simétrico): $\mathbb{R}$ cifra un valor $v \in \mathbb{R}$ transformandolo en un vector $E_v \in \mathbb{R}^m$ [6].

Algorithm 2 Liu — `KeyGen`, `Encrypt`, `Decrypt`

procedure KeyGen(m)
  Generar $SK_m = \big[(k_1, s_1, t_1), (k_2, s_2, t_2), \cdots k_m, s_m, t_m)\big]$ con $(k_i, s_i, t_i) \in \mathbb{R}$
  Salida: $K_{\text{enc}} = K_{\text{dec}} = SK_m$
end procedure
procedure Encrypt(K_{enc}, v)
  Generar $r_1, \ldots, r_m \in \mathbb{R}$ aleatorios
  for $i = 1$ **to** $m-1$ **do**
    $e_i \leftarrow k_i t_i v + s_i r_m + k_i (r_i - r_{m-1})$
  end for
  $e_m \leftarrow (k_m + s_m + t_m)\, r_m$
  $CT \leftarrow E_v = [e_1, \ldots, e_m]$
  Salida: CT
end procedure
procedure Decrypt(K_{dec}, CT)
  $t = \sum_{i=1}^{m-1} t_i$
  $s = \dfrac{e_m}{k_m + s_m + t_m}$
  $v = \dfrac{\sum_{i=1}^{m-1} \big(e_i - s \cdot s_i\big)/k_i}{t}$
  Salida: v
end procedure

Las primitivas homomórficas se definen por componente: $E_3 = E_1 \oplus E_2, \quad E_3 = E_1 \otimes E_2, \quad E' = c \odot E, \quad E_3 = E_1 \ominus E_2$, lo que cubre suma, multiplicación, multiplicación por escalar y resta.

Algorithm 3 CKKS — `KeyGen`, `Encrypt`, `Decrypt`

procedure KEYGEN(n, q, σ)
  Generar secreto $K_{\text{dec}}(x)$ (polinomio pequeño) en $\mathbb{Z}_q[x]/(x^n+1)$
  Elegir $a(x)$ uniforme; muestrear ruido $e(x)$ gaussiano
  $b(x) \leftarrow \big(-a(x)\,K_{\text{dec}}(x) + e(x)\big) \bmod q$
  Salida: $K_{\text{enc}} = (a(x), b(x))$, $K_{\text{dec}}(x)$
end procedure
procedure ENCRYPT(K_{enc}, v, scale)
  Codificar $v \mapsto m(x)$ con *scale*
  Muestrear $u(x)$, $e_1(x)$, $e_2(x)$
  $c_1(x) \leftarrow \big(m(x) + b(x)u(x) + e_1(x)\big) \bmod q$
  $c_2(x) \leftarrow \big(a(x)u(x) + e_2(x)\big) \bmod q$
  Salida: $CT = (c_1(x), c_2(x))$
end procedure
procedure DECRYPT(K_{dec}, CT, scale)
  $m^*(x) \leftarrow \big(c_1(x) + c_2(x)\,K_{\text{dec}}(x)\big) \bmod q$
  Decodificar $m^*(x) \mapsto \tilde{v}$ (aproximación de v)
  Salida: $\tilde{v}$
end procedure

c) CKKS (FHE aproximado): Realiza cifrado de numeros reales basado en el problema RLWE [7]. Opera en el anillo $\mathbb{Z}_q[x]/(x^n+1)$ y permite empaquetar vectores de reales (batching). Admite operaciones de suma, multiplicación, rotación y relinearización, lo que lo hace apto para aprendizaje automático con tolerancia a pequeñas imprecisiones.

Cada uno de estos esquemas se asocia a diferentes costos computacionales y tamaños de texto cifrado. Paillier genera textos cifrados compactos, pero solo admite sumas y multiplicaciones por un escalar. En cambio, Liu y CKKS son completamente homomórficos. Liu produce vectores cifrados de tamaño lineal en m y ofrece aritmética exacta, mientras que CKKS empaqueta múltiples valores en un único polinomio de grado n, lo que aumenta el tamaño del cifrado y la complejidad de controlar el ruido a cambio de permitir operaciones aproximadas sobre lotes de gran volumen.

Aunque el foco de este artículo son los esquemas de cifrado, estos pueden emplearse para soportar tareas de minería de datos con preservación de la privacidad, como clustering y clasificación [10]–[12].

III. Estado del arte

Se llevó a cabo un estudio del estado del arte centrado en trabajos publicados en los últimos cinco años, que abordan cifrado homomórfico aplicado a tareas de minería de datos con preservación de la privacidad (MDPP), enfocado en modos de ejecución *local* y *paralelo* en un único nodo. Se priorizaron propuestas con evaluación experimental y una descripción clara del modo de ejecución, y se excluyeron soluciones que no emplean HE. La Tabla I resume las propuestas seleccionadas y su clasificación por modo de ejecución.

En el modo de ejecución local, tanto la tarea de cifrado como la tarea de minería de datos se ejecutan en una única máquina sin recurrir a hilos ni procesos adicionales. Yunlu Cai *et al.* [10] presentan un protocolo de dos partes que ejecuta k-means homomórfico en un único servidor, operando sobre datos cifrados con el esquema de Liu y sin paralelización. Lekshmy *et al.* [11] proponen un enfoque híbrido para acelerar k-means cifrado con Paillier mediante mini-batches, reduciendo el número de operaciones homomórficas por iteración en una sola instancia de cómputo; sin embargo, la tarea de cifrado se mantiene de manera secuencial. Rovida *et al.* [17] estudian una versión aproximada de k-means basada en CKKS y técnicas de enmascaramiento, en la que el servidor ejecuta la mayoría de las operaciones de agrupamiento cifrado y el cliente solo aplica una máscara al resultado, lo que mejora el rendimiento sin emplear paralelismo explícito. Sokhonn *et al.* [12] describen un método cooperativo de agrupamiento jerárquico sobre CKKS, en el cual el propietario de los datos asiste al servidor durante el ordenamiento de distancias cifradas para completar el proceso en una única máquina. El esquema propuesto por Jiasen *et al.* [15] implementa k-nn seguro sobre datos cifrados con CKKS en un unico servidor en la nube. Asimismo, Zorarpacı *et al.* [16] presentan una estrategia híbrida que combina el cifrado aditivo de Paillier con técnicas de privacidad diferencial, aplicada a clasificadores como One Rule y Naïve Bayes, en este caso, tanto el cifrado como la aplicación de ruido se realizan en un único entorno.

Tabla I
TRABAJOS RELACIONADOS CON EL MODO DE EJECUCIÓN LOCAL Y PARALELO PARA CIFRADO HOMOMÓRFICO.

Autor	Año	Esquema HE	Modo de ejecución
Yunlu Cai *et al.* [10]	2019	Liu	Local
Almutairi *et al.* [13]	2020	Liu	Paralelo
Lekshmy *et al.* [11]	2020	Paillier	Local
Xuancheng Guo *et al.* [14]	2020	Paillier	Paralelo
Jiasen *et al.* [15]	2021	CKKS	Local
Zorarpacı *et al.* [16]	2022	Paillier	Local
Rovida *et al.* [17]	2023	CKKS	Local
Sokhonn *et al.* [12]	2024	CKKS	Local

En el modo de ejecución paralelo se emplean múltiples hilos o procesos dentro de la misma máquina o servicio para distribuir la carga de trabajo. Almutairi *et. al* [13] diseñan un servicio DMaaS en la nube que orquesta contenedores encargados de procesar una matriz de distancia global con el esquema de Liu y MUOPE, escalando dinámicamente el número de procesos de ejecución según la demanda. De manera similar, Xuancheng Guo *et. al* [14] implementan una plataforma MLaaS basada en microservicios y un bus MQTT, donde varios procesos colaboran en el clustering homomórfico utilizando Paillier y RSA para mejorar la latencia extremo a extremo y la sobrecarga de comunicación. Estas propuestas muestran beneficios del paralelismo en un nodo, pero no exponen una interfaz unificada independiente del esquema HE ni detallan una segmentación por filas con balanceo de carga explícito en la fase de *cifrado* previa a la MDPP.

A. Discusión

Los trabajos revisados cubren adecuadamente los modos local y paralelo, pero ninguno ofrece un servicio de cifrado homomórfico agnóstico al esquema que segmente el dataset y distribuya los segmentos de manera balanceada entre los procesos disponibles. Nuestra propuesta aborda este desafío mediante la propuesta de un servicio que aplica patrones de

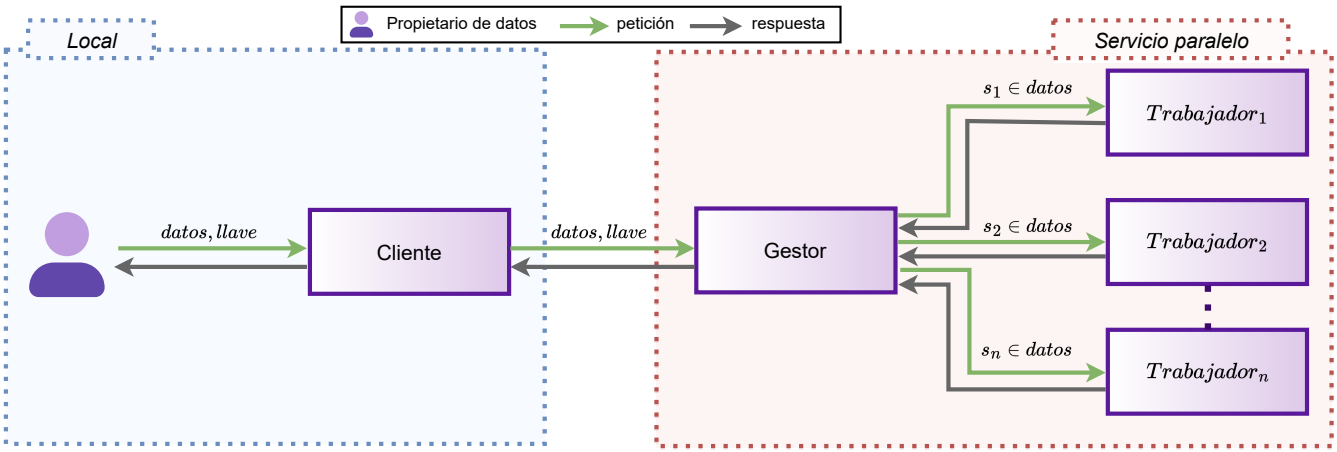

Figura 1. Arquitectura conceptual del servicio paralelo.

paralelismo y balanceo de carga, asignando a cada proceso segmentos proporcionales a su capacidad. Este servicio expone una interfaz única desacoplada del backend criptográfico. El flujo se organiza con el patrón gestor–trabajador para cifrar en paralelo dentro de un único nodo y se valida con tres esquemas representativos (Liu, CKKS y Paillier), facilitando su integración en flujos de MDPP.

IV. Cifrado homomórfico como servicio

El servicio de cifrado homomórfico ofrece una interfaz unificada para distintos esquemas HE. Está diseñado para operar de manera paralela, por lo que la carga de trabajo se divide entre múltiples procesos en el mismo nodo, aprovechando al máximo los recursos de CPU disponibles para reducir los tiempos de cifrado.

El servicio divide en segmentos el conjunto de datos a cifrar, lanza los procesos de cifrado para cada segmento en paralelo, recolectando los resultados de forma transparente.

Consideramos un dataset $D \in \mathbb{R}^{n \times a}$, donde cada fila representa un registro con a atributos. El cifrado conserva la estructura por registro, generando una matriz D', de manera que cada elemento $r_{i,j} \in D$ se transforma en un texto cifrado $E_{i,j} \in D'$:

$$D = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,a} \\ \vdots & \vdots & \ddots & \vdots \\ r_{k,1} & r_{k,2} & \cdots & r_{k,a} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n,1} & r_{n,2} & \cdots & r_{n,a} \end{bmatrix} \xrightarrow{\text{Encrypt}} D' = \begin{bmatrix} E_{1,1} & E_{1,2} & \cdots & E_{1,a} \\ \vdots & \vdots & \ddots & \vdots \\ E_{k,1} & E_{k,2} & \cdots & E_{k,a} \\ \vdots & \vdots & \ddots & \vdots \\ E_{n,1} & E_{n,2} & \cdots & E_{n,a} \end{bmatrix}$$

Para aprovechar el paralelismo, se segmenta por filas usando un tamaño de segmento TS y el número de segmentos $S = \lceil n/TS \rceil$. Definimos el dataset segmentado como:

$$D = \begin{bmatrix} d^{(1)} \\ d^{(2)} \\ \vdots \\ d^{(s)} \end{bmatrix}$$

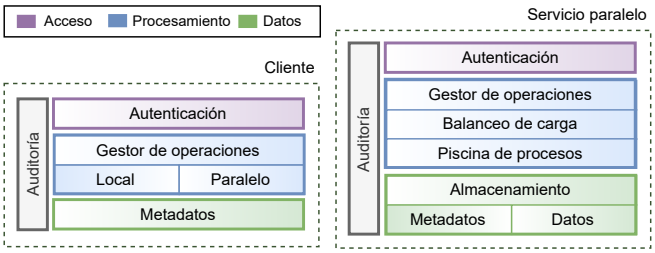

Figura 2. Arquitectura en capas de los componentes del servicio

se realiza el proceso en paralelo

$$d'^{(j)} = \texttt{Encrypt}(K_{enc}, d^{(j)})$$

y se integra para entregar un resultado final al propietario de datos:

$$D' = [d'^{(1)}; d'^{(2)}; \ldots; d'^{(s)}]$$

Este diseño modular, centrado en patrones de paralelismo y balanceo, facilita la integración de cifrado homomórfico en flujos de análisis de datos, mejorando la privacidad sin sacrificar el rendimiento.

La Figura 1 ilustra de manera conceptual el flujo de interacción con el servicio. El proceso inicia del lado del propietario de los datos, que dispone de un cliente. Este cliente facilita la comunicación con el servicio, permitiendo configurar distintos parámetros (cantidad de segmentos y procesos disponibles). Además, ofrece operaciones básicas como generación de llaves, cifrado, descifrado y segmentación. La petición enviada de parte del cliente es recibida por un *gestor* que segmenta el conjunto de datos y lanza múltiples procesos de cifrado. Cada proceso cifra su correspondiente segmento y devuelve el resultado al *gestor*, que agrupa las salidas y entrega al *cliente* los datos cifrados junto con los metadatos asociados. Esta separación permite adaptar el flujo interno para mejorar la eficiencia del cifrado homomórfico.

La Figura 2 presenta la solución organizada en tres capas para dos componentes principales: cliente y servicio paralelo. En ambos componentes, la capa de acceso valida credenciales y parámetros de ejecución antes de aceptar una petición.

En el componente *cliente*, la capa de procesamiento maneja el flujo según la configuración indicada por el usuario: selecciona el esquema de cifrado, determina el tamaño de segmento y el número de procesos, aplica la segmentación por filas y prepara las tareas

para su ejecución en el nodo. Para fines de comparación, es posible ejecutar en modo local, reutilizando el mismo flujo, o bien en modo paralelo, coordinando el envío de segmentos al *servicio paralelo* y la integración en el orden original al finalizar. La capa de datos del componente cliente administra los metadatos de la solicitud y de cada segmento.

En el componente *servicio paralelo*, la capa de procesamiento recibe los segmentos y los distribuye a un pool de procesos en el mismo nodo aplicando balanceo de carga. Cada proceso ejecuta el cifrado del segmento con el esquema indicado, aplica políticas de reintento ante fallos o tiempo excedido, y devuelve resultados parciales al gestor para su agregación, preservando el orden lógico de los segmentos y la integración transparente con el cliente. La capa de datos gestiona los metadatos de ejecución a nivel de proceso y segmento (estado, número de reintentos, tiempos parciales).

De manera transversal, un módulo de auditoría registra eventos y métricas (tiempos de respuesta, uso de CPU/RAM, reintentos y estado) para asegurar trazabilidad y diagnóstico durante toda la ejecución.

V. Evaluación

Con el objetivo de analizar la eficiencia del servicio de cifrado homomórfico propuesto, realizamos tres experimentos que miden el tiempo de respuesta en función del algoritmo y del nivel de seguridad. En todos los casos se emplearon los siguientes backends criptográficos: CKKS con Pyfhel, Paillier con la librería phe y Liu con una implementación propia. El servicio es agnóstico al backend, pues el motor criptográfico se desacopla mediante interfaces; por ello, la evaluación se centra en la ganancia del paralelismo frente a una version secuencial. Primero, evaluamos la escalabilidad con el número de procesos en tres niveles de seguridad (Fig. 3–5), después, comparamos los modos local y paralelo con 16 procesos fijos (Fig.6), por último, verificamos la utilidad comparando el agrupamiento cifrado frente al agrupamiento estándar (Fig. 7).

A. Evaluación 1: Escalabilidad con número de procesos de cifrado

En esta evaluación se midió el tiempo de respuesta del proceso de cifrado empleando los esquemas Liu, Paillier y CKKS en tres niveles de seguridad: 128, 192 y 256 bits. En cada caso, se procesó un conjunto de datos de 100,000 registros por 100 atributos y se varió el número de procesos en cinco configuraciones (1, 2, 4, 8 y 16).

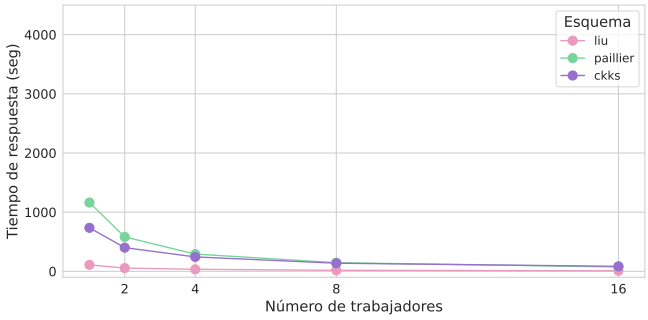

Figura 3. Tiempo de respuesta para distintos esquemas de cifrado al variar la cantidad de procesos con un nivel de seguridad de 128 bits.

La Figura 3 presenta los resultados para 128 bits de seguridad, donde se observa una reducción notable del tiempo de cifrado al incrementar los procesos disponibles. Liu y Paillier alcanzan tiempos de respuesta de 10 y 73 segundos con 16 procesos, mientras que CKKS, más costoso, cae de 735 segundos a 85 segundos.

En la Figura 4, correspondiente a 192 bits, se mantiene la tendencia de mejora, con tiempos máximos iniciales mayores (hasta 2100 segundos para el caso de Paillier) pero un escalado proporcional similar.

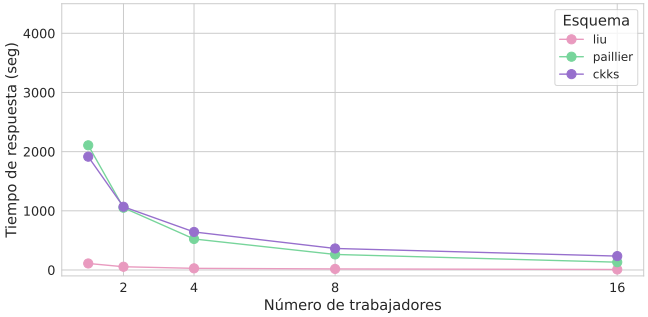

Figura 4. Tiempo de respuesta para distintos esquemas de cifrado al variar la cantidad de procesos con un nivel de seguridad de 192 bits.

Finalmente, la Figura 5 muestra que para 256 bits, el paralelismo sigue siendo efectivo: aunque los tiempos para un proceso aumentan (4207 segundos para CKKS), al usar 16 procesos todos los esquemas reducen sus tiempos de manera significativa (Liu 9 segundos, Paillier 251 segundos y CKKS 509 segundos), confirmando el impacto positivo del balanceo de carga.

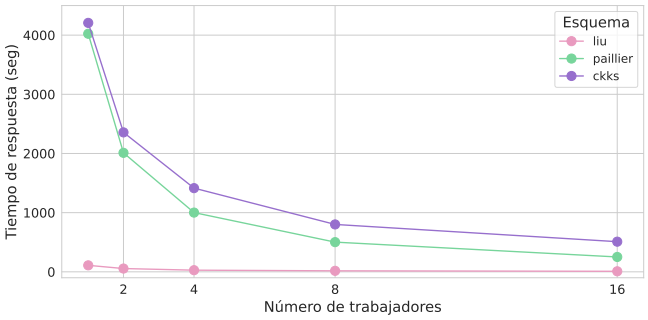

Figura 5. Tiempo de respuesta para distintos esquemas de cifrado al variar la cantidad de procesos con un nivel de seguridad de 256 bits.

Estos resultados validan que el servicio aprovecha eficientemente los recursos computacionales en modo paralelo, atenuando la sobrecarga de cifrado homomórfico incluso en niveles de seguridad mayores.

B. Evaluación 2: Comparativa de modos de ejecución

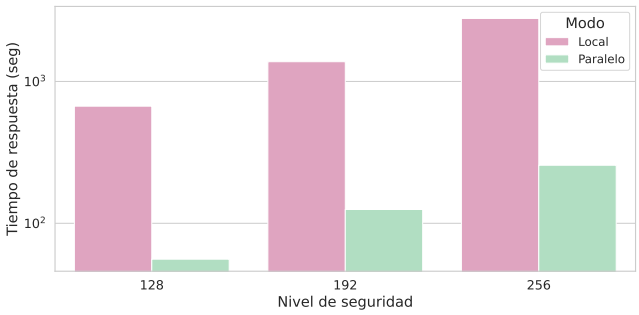

Figura 6. Comparativa de tiempo de respuesta entre modos de ejecución para diferentes niveles de seguridad.

En esta evaluación se comparan los dos modos de ejecución: local y paralelo. Para esta prueba, en el modo paralelo se mantuvo constante el número de procesos (dieciséis). Los tiempos de respuesta se presentan en la Figura 6. En todos los casos, el modo paralelo reduce el tiempo de cifrado de manera sustancial frente al modo local.

C. Evaluación 3: Validación de utilidad

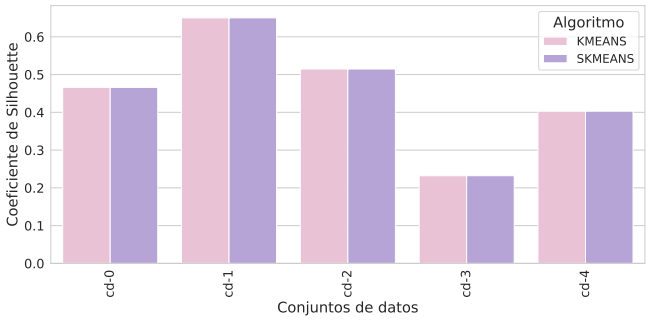

Figura 7. Comparativa de coeficiente de silhouette para los algoritmos k-means y sk-means con el esquema de Liu.

Para comprobar que el cifrado homomórfico no degrada la calidad de los resultados en minería de datos, realizamos una prueba de concepto sobre cinco conjuntos del repositorio UCI (Ecoli, Iris, Breast Cancer, Audit data y Fertility Diagnosis). Se aplicaron los algoritmos kmeans para texto en claro y skmeans para textos cifrados, para este segundo caso, cada conjunto de datos fue segmentado y cifrado usando el esquema de Liu con 128 bits, (en modo paralelo) o cifrado de forma completa (modo local). La Figura 7 muestra los coeficientes de silhouette obtenidos por ambos métodos, con valores que oscilan entre 0.60 y 0.75. En todas las pruebas, k-means y sk-means alcanzan valores idénticos de silhouette , lo que confirma que la fase de cifrado, incluso segmentada y paralelizada, no introduce pérdida de utilidad y preserva la misma calidad de agrupamiento que el algoritmo sin cifrado

VI. Conclusión

En este trabajo se presentó un servicio de cifrado homomórfico, validado con los esquemas Liu, CKKS y Paillier, que ofrece una interfaz unificada respaldada por una arquitectura modular basada en patrones de paralelismo y balanceo de carga. Las pruebas realizadas muestran que aplicar el modo paralelo reduce significativamente los tiempos de cifrado frente a una ejecución secuencial, logrando mejoras de hasta un 75 % en CKKS y entre un 60 % y un 70 % en Liu y Paillier.

Estos resultados confirman que segmentar y distribuir las tareas de cifrado entre procesos mejora el uso de los recursos dentro de un único nodo, mitigando así su elevado costo computacional. Además, la prueba de utilidad con k-means y sk-means demostró que la versión cifrada conserva idénticos coeficientes de silhouette en cinco conjuntos de datos, validando que no hay pérdida de calidad en el agrupamiento.

En conjunto, el servicio facilita la incorporación del cifrado homomórfico en flujos de análisis de datos con preservación de la privacidad, combinando rendimiento y facilidad de uso. Como trabajo futuro, contemplamos: 1) su extensión a entornos distribuidos multinodo, manteniendo la misma interfaz y el patrón de segmentación y ejecución en paralelo, y 2) la integración y evaluación de aceleración por GPU en los núcleos más costosos (por ejemplo, multiplicaciones y rotaciones polinomiales en CKKS, operaciones vectorizadas en Liu y exponenciaciones modulares en Paillier) con el fin de cuantificar su impacto en tiempos de cifrado.

References

[1] M. Ileas Pramanik, Raymond Y.K. Lau, Md. Sakir Hossain, Md. Mizanur Rahoman, Sumon Kumar Debnath, Md. Golam Rashed, and Md. Zasim Uddin. Privacy-preserving big data analytics: A critical analysis of state-of-the-art. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(1):e1387, 2021.

[2] Kathrin Grosse, Lukas Bieringer, Tarek R. Besold, Battista Biggio, and Katharina Krombholz. Machine learning security in industry: A quantitative survey. *IEEE Transactions on Information Forensics and Security*, 18:1749–1762, 2023.

[3] A. Asaduzzaman and ... D'Souza. Increase security by analyzing password strength using machine learning. In *Proceedings of the 2024 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTIDAMT & NCON)*. IEEE, 2024.

[4] Bhomik M. Gandhi, Shruti B. Vaghadia, Malaram Kumhar, Rajesh Gupta, Nilesh Kumar Jadav, Jitendra Bhatia, Sudeep Tanwar, and Abdulatif Alabdulatif. Homomorphic encryption and collaborative machine learning for secure healthcare analytics. *Security & Privacy*, 8(1), 2024. Published September 13, 2024.

[5] Fabian Schmid, Shibam Mukherjee, Stjepan Picek, Marc Stöttinger, Fabrizio De Santis, and Christian Rechberger. Towards private deep learning-based side-channel analysis using homomorphic encryption: Opportunities and limitations. In *Constructive Side-Channel Analysis and Secure Design (COSADE 2024)*, volume 14595 of *Lecture Notes in Computer Science*, pages 133–154, 2024.

[6] Dongxi Liu. Practical fully homomorphic encryption without noise reduction. Cryptology ePrint Archive, Paper 2015/468, 2015. https://eprint.iacr.org/2015/468.

[7] Andrey Kim, Antonis Papadimitriou, and Yuriy Polyakov. Approximate homomorphic encryption with reduced approximation error. In *Cryptographers' Track at the RSA Conference*, pages 120–144. Springer, 2022.

[8] Sudhansu Bala Das, Sugyan Kumar Mishra, and Anup Kumar Sahu. A new modified version of standard rsa cryptography algorithm. In *Smart Computing Paradigms: New Progresses and Challenges: Proceedings of ICACNI 2018, Volume 2*, pages 281–287. Springer, 2020.

[9] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.*, 51(4), jul 2018.

[10] Yunlu Cai and Chunming Tang. Privacy of outsourced two-party k-means clustering. *Concurrency and Computation: Practice and Experience*, 33(8):e5473, 2021.

[11] PL Lekshmy and M Abdul Rahiman. Hybrid approach to speed-up the privacy preserving kernel k-means clustering and its application in social distributed environment. *Journal of Network and Systems Management*, 28(2):398–422, 2020.

[12] Lynin Sokhonn, Yun-Soo Park, and Mun-Kyu Lee. Hierarchical clustering via single and complete linkage using fully homomorphic encryption. *Sensors*, 24(15):4826, 2024.

[13] Nawal Almutairi, Frans Coenen, and Keith Dures. A cryptographic ensemble for secure third party data analysis: collaborative data clustering without data owner participation. *Data & Knowledge Engineering*, 126:101734, 2020.

[14] Xuancheng Guo, Hui Lin, Yulei Wu, and Min Peng. A new data clustering strategy for enhancing mutual privacy in healthcare iot systems. *Future Generation Computer Systems*, 113:407–417, 2020.

[15] Jiasen Liu, Chao Wang, Zheng Tu, Xu An Wang, Chuan Lin, and Zhihu Li. Secure knn classification scheme based on homomorphic encryption for cyberspace. *Security and communication networks*, 2021(1):8759922, 2021.

[16] Ezgi Zorarpacı and Selma Ayşe Özel. A hybrid approach of homomorphic encryption and differential privacy for privacy preserving classification. *International Journal of Applied Mathematics Electronics and Computers*, 8(4):138–147, 2020.

[17] Lorenzo Rovida. Fast but approximate homomorphic k-means based on masking technique. *International Journal of Information Security*, 22(6):1605–1619, 2023.