# Criptografía Homomórfica

*Cátedra ISDEFE UAH*

Noviembre/Diciembre 2020

## Resumen

Se conoce como criptografía homomórfica al conjunto de técnicas criptográficas que permiten operar con datos cifrados de tal forma que, al descifrarlos, los resultados de las operaciones se manifiesten sobre el texto en claro. Aunque su postulado data de 1978, hasta 2009 no se plantea un mecanismo que permita aplicarla a la computación. Durante la última década la capacidad de estas técnicas ha evolucionado vertiginosamente, pero, ¿qué límites y aplicaciones tiene realmente? Este trabajo pretende ilustrar la evolución sufrida por la criptografía homomórfica hasta su estado actual de una forma ordenada, permitiendo al lector comprender los principales elementos que conforman esta prometedora tecnología.


*Palabras clave*: criptografía homomórfica, computación segura, privacidad diferencial

# Índice

## Introducción

La criptografía es el ámbito de la criptología destinado a proteger la información mediante la aplicación de algoritmos y protocolos. Podemos describir de forma resumida los sistemas criptográficos de protección de la información como herramientas que ofrecen dos operaciones: cifrar (volver ilegible un texto plano, haciéndolo indistinguible de un texto aleatorio) y descifrar (recuperar el valor original de la información desde un texto cifrado).

Para que estas operaciones sean seguras desde el punto de vista de la autenticación de sus usuarios legítimos, será necesario establecer un valor secreto que haga de llave. En función de si este valor secreto (la clave) es el mismo en la operación de cifrado que en la de descifrado, hablaremos de sistemas de criptografía simétrica o asimétrica (respectivamente).

Estos protocolos de cifrado, que son esenciales para el funcionamiento de los sistemas de información tal y como los conocemos, permiten tanto proteger la información en tránsito como en reposo, pero adolecen de ser completamente ineficaces cuando los datos cifrados tienen que ser validados o procesados por una tercera entidad: no puede hacer nada con ellos sin descifrarlos (es decir, sin conocer su contenido). La criptografía homomórfica pretende resolver este problema.

Se conoce como sistemas de criptografía homomórfica a aquellos criptosistemas que permiten realizar operaciones (evaluar funciones) sobre un texto cifrado, y que estas (u otras equivalentes) se manifiesten en el texto plano tras descifrar. Por ejemplo, teniendo los siguientes elementos:

1. Una función de cifrado $Hom(m, K)$ y otra de descifrado $Hom^{-1}(c, K)$, con $m$ texto plano, $c$ texto cifrado y $K$ elemento secreto

2. Dos mensajes en claro $M_1$ y $M_2$
3. Dos mensajes cifrados $C_1 = Hom(M_1)$ y $C_2 = Hom(M_2)$
4. Una operación $+$ definida en el espacio *texto plano*, y otra operación $\times$ definida en el espacio *texto cifrado*.
5. El sistema $(Hom\ \ , Hom^{-1})$ será un sistema homomórfico si $Hom^{-1}(C_1 \times C_2) = Hom^{-1}(C_1) + Hom^{-1}(C_2)$

Cuando hablemos de criptografía homomórfica, por lo tanto, tendremos presente además de las operaciones de cifrado y descifrado una operación nueva: la operación de evaluación.

Serán de especial interés los sistemas homomórficos implementados con criptografía asimétrica, porque además de permitirnos operar entre textos cifrados, nos permitirán introducir valores a placer en la ecuación: nos permitirán generar nuevos textos cifrados para operar, sin poner en riesgo la información protegida.

Cuando estos sistemas nos permitan, además, realizar operaciones arbitrarias, hablaremos de sistemas completamente homomórficos (o FHE, siglas en inglés de Fully Homomorphic Encryption).

## Organizaciones implicadas

Para que las tecnologías de criptografía homomórfica puedan adquirir la solidez necesaria, además de como producto criptográfico, como productos tecnológicos implantables en organizaciones reales, se han desarrollado diversas iniciativas de estandarización y análisis.

Desde el NIST se estudia la evolución de los algoritmos de criptografía homomórfica desde dos grupos: principalmente, y por motivos obvios, desde el proyecto PEC [1]; y

colateralmente en el proyecto PQC [2], debido a que varios algoritmos post-cuánticos (como NewHope [3]) están basados en el problema LWE (del que hablaremos más adelante).

Pese a la aprobación en 2019 de una norma ISO sobre algoritmos de criptografía homomórfica [4, p. 180] (ya 10 años obsoleta al momento de ver la luz) el proceso de estandarización, y el seguimiento al estado del arte, van de la mano de la organización Homomorphic Encryption Standardization [5].

## Objetivo

El objetivo de este estudio será analizar el estado actual de la criptografía homomórfica, mostrando sus distintas fases de desarrollo desde sus primeros postulados, y prestando especial atención a la vertiginosa evolución que ha sufrido durante la última década. Se analizará tanto a nivel técnico (algoritmos e implementaciones) como a alto nivel, mostrando los usos que tendrá para la industria y para la resolución de problemas de privacidad que han supuesto siempre un bache en el desarrollo de tecnologías en la nube para sectores concretos (como la medicina o la seguridad nacional).

### Guía de lectura

Este documento tiene una estructura gradual, en cuanto a nivel técnico se refiere. Para ello, se ha dividido en tres grandes unidades: introducción, marco teórico y conclusiones.

Los tres primeros capítulos (Introducción, Contexto e Innovación) ofrecen una visión a alto nivel de la criptografía homomórfica. Si bien es inevitable tratar con elementos técnicos, cuando el nivel requerido para su comprensión pueda ser excesivo, se trata de suavizar para ser abordado de nuevo en capítulos posteriores.

El bloque abierto por el capítulo de base teórica persigue seguir la misma estructura incremental, para que el lector técnico pueda comprender elemento a elemento la construcción de los algoritmos de criptografía homomórfica. Este bloque, que abarca hasta computación, puede ser sorteado si resulta arduo y pasar directamente al bloque de conclusiones.

Para finalizar, se pondrá en perspectiva el estado del arte de la criptografía homomórfica (Retos de la criptografía homomórfica) y se extraerán conclusiones, principalmente, sobre su nivel de madurez.

## Contexto

Comenzaremos haciendo un repaso por los distintos avances que ha ido experimentando la criptografía homomórfica hasta hoy. En base a los distintos hitos alcanzados en el desarrollo de sistemas homomórficos se han definido varias generaciones, desde los sistemas más primitivos (que apenas permitían alguna operación, y de forma casual) hasta los tan ansiados sistemas completamente homomórficos.

Aunque, como veremos, los sistemas basados en criptografía homomórfica van a ser más garantistas y versátiles, conviene conocer qué otras aproximaciones hay para la protección de los datos a la hora de realizar cómputos con ellos.

### Enclaves

Existen tecnologías de seguridad como ARM TrustZone, Intel SGX o Secure Enclave Apple; que buscan proteger la información permitiendo el trabajo con la misma. La base de estas tecnologías son sistemas de aislamiento de información y procesos implantados en el hardware de la CPU, implementando una suerte de sistema operativo independiente. Su

utilización no asegura que se cumplan todas las propiedades deseables para el procesamiento de datos en un equipo externo, y sigue sin asegurar completamente la confidencialidad de nuestros datos [6].

Sin embargo, la criptografía homomórfica ofrece un valor diferenciador con respecto a estas tecnologías, y es que los datos no necesitan estar en claro en ningún momento para ser procesados.

**Secure multi-party computation**

Otras aproximaciones más fiables desde el punto de vista de proteger rigurosamente la información, pero que no llegan a ofrecer la versatilidad de cómputo de un sistema FHE, son todas aquellas recogidas bajo el paraguas de la computación multiparte que no están basadas en homomorfismos.

Uno de los problemas más conocidos que pueden resolverse mediante computación multiparte es el problema de los millonarios de Yao [7]. En este problema, dos millonarios quieren saber cuál de los dos es más rico sin que el otro pueda saber exáctamente cuánto dinero tiene. Aunque existen varias aproximaciones para su resolución, quizá la generalización más interesante es la de la intersección de conjuntos privados (o PSI por sus siglas en inglés).

Ilustraremos la evolución de los métodos de computación multiparte segura a través del problema de la intersección de conjuntos privados debido a que es aplicable a multitud de casos de uso, y está compuesto por varios elementos clave para la comprensión del campo. Este problema consiste en tratar de buscar la intersección entre dos conjuntos (por ejemplo, dos agendas de contactos) sin que ninguno de los participantes tenga que exponer sus datos a los demás, ni sea necesario un tercero de confianza.

Para resolver este problema serán necesarias 2 operaciones esenciales:

➔ Ordenación segura

Juntar ambos conjuntos y ordenarlos.

➔ Comparación segura

Comparar datos adyacentes. Si son iguales, están en ambos conjuntos.

➔ Desordenación segura

Introducir entropía en el resultado.

Una forma trivial de resolver el problema podría ser comparar los hashes de las entradas de cada conjunto. Buscando la intersección de dos conjuntos de datos hasheados, los datos originales serían invisibles y podría obtenerse una primera solución al problema. Sin embargo, este sistema tendría dos fallas:

➔ Podría exponer el tamaño de los conjuntos, por lo que no tendría seguridad semántica.
➔ En un conjunto de elementos con baja entropía, por ejemplo, números de teléfono, sería inseguro (puede atacarse fácilmente por fuerza bruta). Por lo tanto no sería un método generalizable.

Existen también aproximaciones basadas en la asunción Diffie-Hellman decisiva [8], como el protocolo de transferencia inconsciente [9]:

1. Teniendo un conjunto de datos $(x_1, \dots, x_n)$.
2. Procesando el mismo conjunto de datos con las claves privadas de $a$ y $b$:$(x_1, \dots, x_n)_a$ , $(x_1, \dots, x_n)_b$ , $(x_1, \dots, x_n)_{ab}$ .
3. No será posible obtener $a$, $b$ o $x_n$

4. a podrá encontrar intersecciones con el conjunto de datos generados por b sin revelar los datos originales, y viceversa.

### *Filtros de Bloom*

Por último, los Filtros de Bloom [10] son estructuras que permiten comprobar si un elemento pertenece a un conjunto sin revelarlo. Un filtro de bloom nunca dará un falso negativo (nunca dirá que un elemento no está en un conjunto si lo está), pero sí puede dar falsos positivos (indicar que está en el conjunto sin estarlo).

Un filtro de Bloom puede implementarse como un vector de valores binarios, inicializado con todas las entradas a 0. Cada vez que se desee introducir un valor, será procesado por varias funciones hash, y cada una de ellas devolverá un resultado entre cero y el tamaño del vector. Por cada resultado introducido se actualizará el filtro marcando las posiciones que indique su hash. Cuando se quiera comprobar si un valor está en el conjunto, sólo habrá que comprobar si estas posiciones están marcadas: si no existe, no lo estará; si lo están, puede existir, o no.

Puede verse un ejemplo del funcionamiento en el Anexo I

### *Criptografía homomórfica*

Finalmente, la criptografía homomórfica terminará simplificando estas construcciones, a la vez que dará versatilidad a la hora de modificarlas. En 2012, López-Alt et al. presentan un sistema destinado a la computación multiparte [11] basado en las propiedades homomórficas del criptosistema NTRU. Haciendo esta extensión por un lado rompen el modelo cliente servidor que ha imperado a la hora de conceptualizar la criptografía homomórfica, y por otro

condensan la computación multiparte en un único mecanismo casi omnipotente (desde el punto de vista de la computación multiparte).

## Generaciones

Hay múltiples sistemas que presentan homomorfismos con algunas operaciones (es decir: alguna operación sobre el texto cifrado se manifiesta al descifrar, pero no cualquier operación). Uno de los principales retos en los que se ha involucrado la comunidad científica en torno a la criptografía homomórfica es el de crear un criptosistema que permita aplicar las operaciones de suma y producto. Este objetivo no es caprichoso: la suma y el producto de números binarios equivalen a puertas lógicas AND y XOR (como se muestra en el anexo II), puertas que combinadas, permiten realizar cualquier operación computacional. Es decir, un criptosistema con homomorfismo en la suma y el producto permitirá construir cualquier circuito lógico para procesar el texto cifrado: tendrá compleción homomórfica.

Así, podríamos distinguir las siguientes generaciones:

### *Pre-HE*

Ya en 1978 Rivest, Adleman y Dertouzos [12] presentan un sistema criptográfico que permite realizar determinadas operaciones sobre texto cifrado, y el propio sistema RSA presenta un homomorfismo con respecto al producto.

Por otro lado, el criptosistema de Paillier [13] es homomórfico con respecto a la suma. Es decir, se pueden sumar dos textos cifrados y obtener la suma de los respectivos textos en claro al descifrar.

Estos homomorfismos, aunque son interesantes -especialmente por su capacidad para procesar y compartir información de forma oculta- tienen un recorrido muy corto si se desean utilizar para computar.

***Primera generación***

En 2005, y tras estudiar aproximaciones criptográficas basadas en retículos [14], Oded Regev presenta un sistema de cifrado cuya función trampa está basada en el problema del aprendizaje con errores [15].

Pese a existir otros criptosistemas con homomorfismos en la suma y el producto (como es el caso de NTRU), la aportación de Regev marca un hito en la criptografía homomórfica, y es quien marca el camino de las posteriores investigaciones destinadas a la computación con textos cifrados.

El sistema de Regev presenta una carencia para aplicarlo a la computación, y es que a medida que se opera, acumula un nivel de error que acaba corrompiendo el resultado. En 2009 Craig Gentry presenta una nueva técnica capaz de ir eliminando este error, conocida como bootstrapping [16] (más adelante detallaremos ambos sistemas).

***Segunda generación***

En la segunda generación se desarrollan algoritmos de criptografía homomórfica válidos para la computación, aunque con ciertas restricciones. La solución que aportan los autores de las técnicas empleadas en esta generación (eminentemente Brakerski, Vaikuntanathan, Vercauteren y Fan) consisten principalmente en trabajar sobre grandes espacios numéricos (en comparación con el tamaño del conjunto en el que se genera el error) para que, aunque el error crezca tras operar, se mantenga dentro de unas cotas que no

corrompan el resultado. De esta forma, y definiendo cuál es exáctamente el límite de integridad de los datos, se podrá calcular el número de operaciones realizables.

Aunque no permitirá el cálculo de forma arbitraria, abre la ventana a construir los primeros sistemas que quepan dentro de estos límites.

### *Tercera generación*

Finalmente Gentry, Sahai y Waters diseñan un algoritmo [17] capaz de computar el producto y la suma de una forma eficiente y aplicable a la computación arbitraria. Este algoritmo, ya de tipo FHE, permite prescindir de la clave pública del usuario para realizar.

Tras GSW, el trabajo de los investigadores se ha centrado principalmente en la eficiencia de las soluciones y la usabilidad para su implementación.

## Implementaciones

En este capítulo se mostrarán implementaciones que permiten el uso de criptografía homomórfica “fuera del papel”. Veremos librerías, productos que las usan y, por último, abstracciones orientadas a simplificar la transformación de sistemas normales a sistemas homomórficos.

### *Librerías*

En el anexo III se muestran las principales librerías que pueden utilizarse para realizar implementaciones de criptografía homomórfica. Varias de ellas se encuentran descontinuadas, o han buscado servir como pruebas de concepto, pero los desarrolladores están trabajando activamente bien en:

- Su implantación en tecnologías existentes, como los desarrollos de OpenMined [18] tratando de mejorar la privacidad de los sistemas de inteligencia artificial con criptografía homomórfica.
- La adaptación de las librerías existentes a otros lenguajes, como *lattigo* [19].

***HE en la industria***

La empresa Inpher [20] desarrolla soluciones para aumentar la privacidad en implementaciones de machine learning. Además han participado de forma activa en el desarrollo de la librería TFHE.

En IXUP [21] desarrollan una plataforma de analítica de datos también orientada a preservar la privacidad de los mismos. De esta forma se puede, por ejemplo, permitir la operativa con datos especialmente protegidos por leyes de protección de datos asegurando el cumplimiento.

El producto ZeroReveal® desarrollado por Enveil [22] envuelve el procesado de información en la nube utilizando criptografía homomórfica para proteger no sólo la confidencialidad de los datos, sino también la confidencialidad de las operaciones que se realicen sobre los mismos.

Por último, Duality Tech desarrolla una plataforma que integra mecanismos de computación y consultas SQL utilizando criptografía homomórfica. Si dos participantes (e.g. dos organizaciones distintas) desean operar con datos privados de forma común, pueden hacerlo a través de un middleware instalado en sus instalaciones que cifra y gestiona la comunicación de las dos partes en la nube. Internamente utiliza la librería PALISADE (una librería SHE).

### *Toolchains*

El desarrollo de aplicaciones con criptografía homomórfica utilizando las librerías mencionadas requiere un perfil de desarrollador muy especializado, que además conozca el funcionamiento de los esquemas criptográficos, a riesgo de cometer fallos que pongan en riesgo la seguridad del sistema. Para lidiar con este problema se han desarrollado frameworks destinados a traducir código fuente "clásico" en código homomórfico de forma segura. Las principales implementaciones son Alchemy [23] y Cingulata [24].

El objetivo de Alchemy es reducir la complejidad de codificar un circuito lógico con operaciones en el dominio cifrado a la complejidad del circuito en texto plano. Busca convertir código normal en código homomórfico lidiando con problemas que los desarrolladores no tendrían por qué entender (reducción de ruido, gestión de claves, etc).

En la otra mano tenemos Cingulata, que implementa todas las funcionalidades de Alchemy (implementando el código en C++), pero además trabaja con esquemas FHE como TFHE. Su documentación es muy completa, con numerosos ejemplos.

Por último, cabe la mención de ABY [25]. Aunque no esté recomendado para aplicaciones en producción, implementa sistemas de computación segura entre dos, y ofrece una gran cantidad de circuitos lógicos que pueden ser útiles (como documentación) para implementaciones con TFHE.

## Innovación

En este capítulo se mostrará la amplitud de aplicaciones que se puede implementar mediante criptografía homomórfica, aportando un mayor grado de privacidad y confidencialidad a sistemas clásicos, o permitiendo la construcción de sistemas que serían

impensables sin ella. La importancia de esta tecnología radica en que solventan múltiples fallos en aplicaciones para producir ataques man-in-the-middle o el envío y almacenamiento de datos sensibles en texto plano. La criptografía homomórfica pretende dar una solución a problemas de una multitud de dominios y poder desarrollar e innovar aplicaciones de una manera más segura.

**Sistemas de voto electrónico**

En la actualidad el poder de los países democráticos reside en el pueblo, que es quien elige qué partido político gobernará los próximos años. Con el avance de la tecnología, la población se beneficia en múltiples aplicaciones de la vida cotidiana. Esto ha conllevado la aparición de un sistema de voto electrónico online, en el que no es necesario salir de casa para poder votar manteniendo el mismo anonimato, integridad, confidencialidad y disponibilidad.

La implementación de un sistema de voto electrónico online clásico mantiene una serie de debilidades debido a que es necesario descifrar cada voto para posteriormente poder sumarse al partido político que pertenece. La criptografía homomórfica ofrece una solución y respuesta a este tipo de problemas, debido a que es posible implementar una aplicación que permita operar, y por tanto sumar votos cifrados entre sí y de este modo mantener los pilares de la seguridad de la información, confidencialidad, integridad y disponibilidad.

El sistema de voto electrónico basado en criptografía homomórfica [26] se encuentra basado en la investigación de Yingming Zhao [27] que aporta anonimidad y privacidad, pero además mejora los problemas que plantea el trabajo de Zhao como por ejemplo cómo supervisar el proceso de voto para impedir conspiraciones.

Estos sistemas de voto electrónico basado en criptografía homomórfica permiten simplificar en gran medida los sistemas clásicos, lo que se traduce en un sistema que supone un ahorro económico manteniendo la integridad, confidencialidad y disponibilidad de igual manera que si se votara físicamente.

## Aplicaciones médicas

La medicina es un campo que requiere tratar con datos especialmente protegidos por lo que, una mala implementación puede llevar a brechas de seguridad con un impacto elevado al tratarse de datos identificativos [28]. En 2018 el proveedor de seguros de salud Anthem Inc sufrió una de las mayores brechas de seguridad de la historia, exponiendo información sobre la salud de casi 80 millones de personas, la multa de la HIPAA (Ley de Contabilidad y Transferencia del Seguro de Salud [29]) de 16 millones de dólares se suman a los 5,55 millones de dólares pagados en 2016 [30]. Las nuevas reglas de la HIPAA estiman que las próximas brechas superen en multas más de 1 billón de dólares, por lo que es necesario implementar aplicaciones debidamente protegidas y con datos de clientes cifrados correctamente.

### *Aplicaciones en genomas*

Para el campo de la genómica se acumulan miles de cantidades de secuencias de ADN y ARN, muchos estudios de enfermedades complejas requieren de estas secuencias para poder detectar patrones, estos datos genéticos pueden revelar información significativa como el posible riesgo que aparezca una enfermedad o no [31]. Sin embargo, como se ha comentado anteriormente, es necesario tener especial cuidado con este tipo de datos especialmente identificativos, donde el Reglamento General de Protección de Datos (RGPD) establece un fuerte hincapié en su protección y salvaguardado.

En 2018, alrededor de 92 millones de cuentas con contraseñas, datos genéticos y biométricos se filtraron del servicio *MyHeritage* [32], una de las compañías más grandes del mercado para construir la descendencia genealógica de donde procede una persona. Estas filtraciones pueden traer graves consecuencias como pérdidas financieras, daños a la empresa y asumir la responsabilidad legal de la filtración, en vista de ello los primeros beneficiarios en este tipo de aplicaciones son los seguros médicos, ya que se aumentará la seguridad de sus aplicaciones para estar protegidos frente a cualquier filtración y posterior multa.

La identificación de qué causa genética conlleva una enfermedad puede conducir muchas veces a un tratamiento mucho mejor, esto se conoce como *Match Maker* [33]. El uso de la criptografía homomórfica facilita la subida de datos especialmente protegidos a la nube con seguridad, de tal manera que sea posible ofrecer una mejora en la personalización médica y con ello mejorar el bienestar de los pacientes.

La criptografía homomórfica orientada a aplicaciones que tratan datos sobre genomas humanos surge principalmente para abordar las causas genéticas de los pacientes con enfermedades poco frecuentes.

Las operaciones que se utilizan para la compartición de datos genómicos son sencillas, por lo que es viable de implementar mediante criptografía homomórfica. Es importante retransmitir datos de manera fiable para asegurar su disponibilidad e integridad, de esta manera se facilita entender las variantes genéticas y su evolución de manera segura.

Una de las ventajas de este tipo de aplicaciones se debe a que no es necesario que la velocidad de cómputo se realice en tiempo real, es decir, la latencia del servicio de la operación puede tardar varias horas sin que afecta al rendimiento del sistema.

A bajo nivel los problemas técnicos que plantean son:

- Comparación

  Comparar datos adyacentes y si son iguales, están en ambos conjuntos.

- Ordenación

  Que incluyen, por definición, sentencias de comparación de datos. Esta ordenación se realiza mediante redes de ordenamiento y siempre y cuando se conozca el tamaño (en bits) de los datos.

Como veremos en próximos capítulos son problemas que ya están siendo abordados y resueltos.

***Búsquedas en bases de datos médicos cifrados***

En el área del cáncer, los tumores suelen ser muy diferentes unos de otros. La respuesta de cada paciente a un tratamiento es difícil que se sepa a priori, depende de múltiples factores genéticos. Es necesario no sólo estratificar los tratamientos en función de la sensibilidad a los fármacos, sino también evitar el sobretratamiento y predecir los posibles efectos adversos para la salud [34].

La utilización de la criptografía homomórfica permite acceder a registros médicos sin que se consulten ni envíen en claro. También permite que las consultas que se realizan sobre los datos sean cifradas y que devuelvan una respuesta igualmente cifrada al que lo solicita. La respuesta se descifra y se generan las operaciones correspondientes a tratar los datos de una manera clásica, pero aportando una capa de seguridad sobre las búsquedas. Como se ha explicado anteriormente la aproximación de la criptografía homomórfica se puede extender a varios campos del cuidado de la salud [35].

Los principales interesados en este tipo de aplicaciones son los servicios que salvaguardan este tipo de datos especialmente identificables, en este caso los hospitales y clínicas. La privacidad de los datos y las necesidades de las organizaciones de salud exigen compensaciones caras en el caso de filtraciones de datos médicos y de pacientes, pudiendo dar lugar a grandes daños económicos y sociales tanto para las organizaciones como para los pacientes.

Las operaciones que son necesarias implementar este sistema de manera segura con criptografía homomórfica son operaciones CRUD (Create, Read, Update, Delete). Algunas de ellas dependen de un elemento que haga de selector. El selector realiza una operación para elegir entre un valor y otro en base a una condición, operación que será posible siempre que el sistema cuente con la capacidad de realizar cualquier operación booleana mediante puertas lógicas.

En términos de eficiencia no es estrictamente necesario que la latencia de los servicios permita el trabajo en tiempo real, sino que es posible que se ejecute por horas. Computacionalmente hablando es una ventaja a la hora de realizar las operaciones debido a que no es crítica la optimización de tiempo en este tipo de aplicaciones.

***Uso de aplicaciones móviles para la realización de diagnósticos médicos***

Mediante tecnología vestible un hospital es capaz de recoger datos médicos como el ritmo cardíaco, la quema de calorías, los pasos dados, presión arterial, etc [36]; los datos enviados a la nube se encuentran expuestos a sufrir ataques contra la disponibilidad e integridad de los sistemas debido al envío y almacenamiento en el servidor en texto plano. Una de las primeras aproximaciones para frenar este tipo de ataques aumentando la seguridad fue Boxcryptor [37] y SpiderOak [38], proveedores en la nube que cifran los datos antes de enviarse

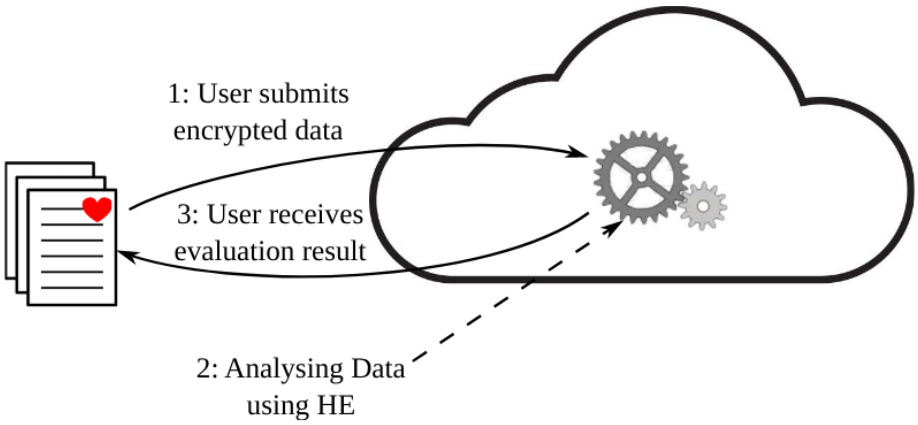
asegurando la privacidad de los datos enviados, mientras la clave de descifrado la posee el cliente, o un nuevo esquema de índice mediante búsqueda por palabra clave sobre datos cifrados denominado P-index o índice de posición [39].

Una de las desventajas de los servicios nombrados anteriormente es la incapacidad de operar sobre los datos cifrados en el propio servidor. La criptografía homomórfica ofrece una solución frente a este tipo de aplicaciones ligeras, implementando aplicaciones que envían los datos médicos de usuarios cifrados preservando la privacidad, analizando los datos cifrados en la nube.

La implementación de la aplicación móvil pretende enviar datos médicos a la nube mediante un algoritmo de criptografía homomórfica. Un hospital puede recoger datos médicos de clientes y se envían a la nube (previamente cifrados) para ser tratados. Un caso de uso puede ser el factor de riesgo cardiovascular, donde el propio servidor procesa y envía datos a la aplicación del cliente para conocer su situación actual [40].

La criptografía homomórfica aporta seguridad semántica debido al ruido añadido. Esto se traduce que a partir de un texto plano es posible obtener múltiples textos cifrados [41]. Por otro lado también puede suponer un peligro si sobrepasa cierto número de operaciones debido a que es imposible descifrar dicho mensaje debido al ruido excesivo añadido, por ello muchos autores consideran evaluar circuitos hasta un nivel de profundidad máxima.

La aplicación móvil consta de dos partes: el usuario (dueño de la clave privada) y el servidor (dueño del algoritmo y encargado de operar con los datos cifrados, de esta parte depende la confidencialidad de la implementación). El servidor ejecuta el diagnóstico sobre el texto cifrado y se lo devuelve a la aplicación móvil. La aplicación del cliente que tiene la clave privada es capaz de descifrar la información devuelta por el servidor e interpretar el análisis.




*Análisis de datos mediante criptografía homomórfica como un servicio en la nube* [39]

La aplicación mostrada en [39] se ejecuta en un tiempo razonable razonable con un rendimiento de 4 segundos. La implementación de criptografía homomórfica resulta viable para aplicaciones ligeras si observamos el aumento de la confidencialidad en relación con el aumento del tiempo de computación.

## Aplicaciones en sistemas de control

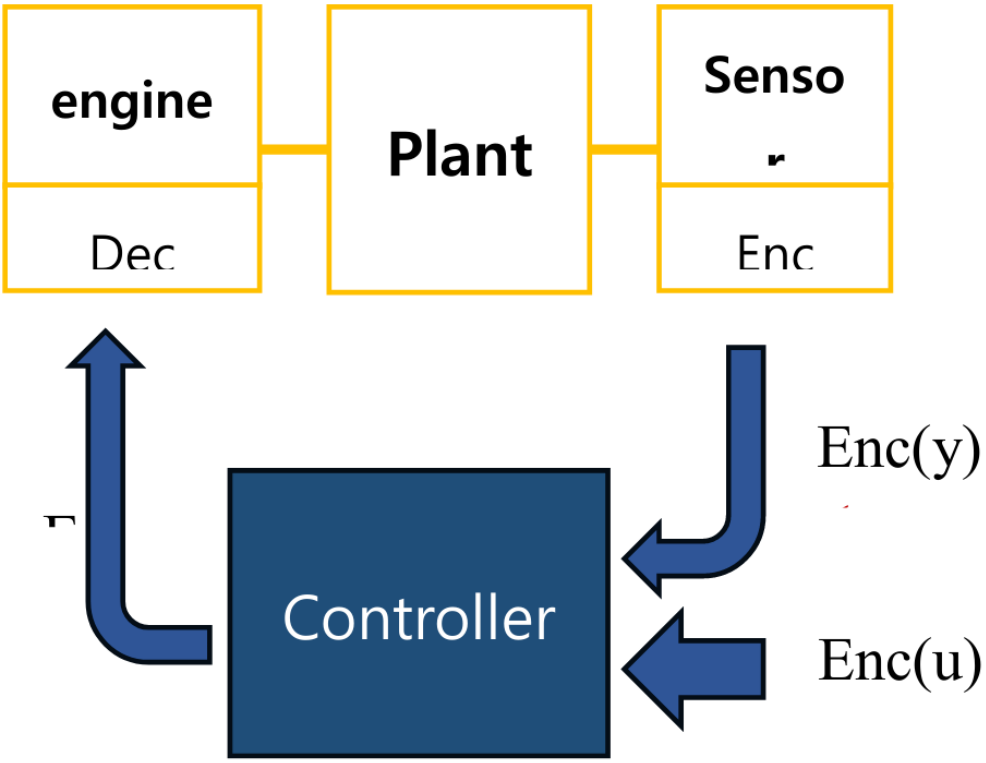
Un sistema de control es cualquier aplicación que dispone de sensores, que reciben datos del exterior, lo envíen al controlador para procesarlo, y en función de los resultados envían información a los actuadores para que respondan de una manera u otra. Las aplicaciones que utilizan sistemas de control son prácticamente cualquier sistema, como coches inteligentes, detectores de metales, centrales nucleares, electrodomésticos, etc.

### *Sistemas de control*

En un sistema de control clásico un cibercriminal puede realizar un ataque man-in-the-middle para capturar los datos que se envían del sensor al controlador o del controlador al actuador. Debido a ello, se recomienda que los sensores envíen datos al controlador cifrados

[42], y que este procese y envíe los datos al actuador cifrados también. Cifrando estas comunicaciones se evita que los cibercriminales secuestren los datos de control, impidiendoles afectar a la confidencialidad del sistema, ni manipularlos afectando a la integridad. Sin no fuese así, y sólo se aplicase cifrado en tránsito, no sería posible evitar que los datos fuesen descubiertos en texto plano si existiera un malware dentro del controlador.

La principal dificultad para aplicar criptografía homomórfica en este ámbito es que los datos deben ser tratados en tiempo real, y no es viable que la duración de la operación dure minutos. Cuanto más sensible sea el sistema de control más preciso debe ser el tratamiento de las operaciones, y es necesario que la velocidad de reacción se acote a unos límites según la aplicación desarrollada. La velocidad de reacción de un airbag en un accidente de coche no se debe de retrasar más de un determinado slot de tiempo, o los sistemas de control de una central nuclear podrían no detenerse a tiempo (y seguir enriqueciendo uranio hasta fallar) si no evalúan las operaciones con la suficiente rapidez. Por lo tanto, es necesario analizar la viabilidad de las operaciones (en cuanto a su eficiencia) para conocer en qué sistemas es posible implementar criptografía homomórfica.




*Configuración de un sistema de control mediante un controlador que soporte cifrado homomórfico* [42]

***Redes eléctricas inteligentes***

Las ciudades del siglo XXI están experimentando una época de expansión tecnológica, y esta expansión se puede aprovechar para mejorar la calidad de la vida y de los servicios de una ciudad. Los sistemas de control se pueden aplicar a numerosas aplicaciones como es el caso de las redes eléctricas inteligentes.

Realizando un análisis, es viable desarrollar redes eléctricas inteligentes que implementen criptografía homomórfica, cada nodo esclavo del cluster debe ser monitorizado y enviar sus datos cifrados al nodo maestro. Los dispositivos involucrados en este tipo de

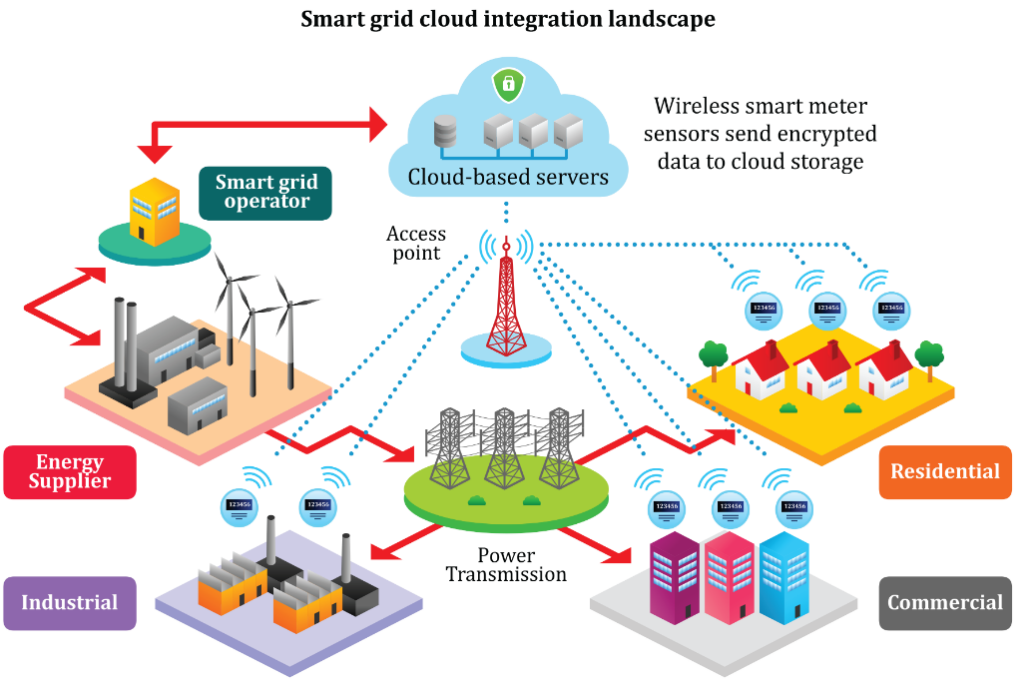
aplicaciones son los contadores eléctricos, contadores de agua y contadores de gas inteligentes. Los contadores tradicionales no dotan de funcionalidades tales como entrega de potencia, información del consumo, utilización de uso de energía, gestión de errores, de uso, monitorización de generación de electricidad.

Los ataques hacia este tipo de aplicaciones afectan a la privacidad y seguridad [43], pudiendo manipular la energía que se transmite, y con ello poder reducir los costes que supone la utilización de energía, además es posible falsear el consumo de energía, atacar al sistema y a su base de datos. Es posible controlar dispositivos eléctricos ajenos como demostró la empresa de ciberseguridad Tarlogic en Marzo de 2020 [44].

La solución que se ofrece es poder segmentar la red de una manera distribuida, de tal manera que se proteja la privacidad y además se envíen datos cifrados manteniendo la eficiencia. Este esquema de cifrado aporta seguridad semántica y las operaciones disponibles sobre mensajes cifrados permitidos son funciones de agregación [43].

Como se ha comentado el nodo maestro debe ser capaz de leer las mediciones cifradas y responder a ellas, poder analizar consumos inusuales y avisar al responsable para analizar la anomalía, de este modo si existe un malware en un nodo y el consumo de electricidad aumenta se podría detectar dicho virus.

Una de las ventajas de aplicar criptografía homomórfica a las redes eléctricas inteligentes es la utilización de un único servidor seguro en la nube, por lo que es más económico que la implementación con otros métodos de cifrado como MPC (Computación multiparte) que requieren de más capacidad tecnológica y dinero para poder implementar el mismo sistema.




*Implementación de un Smart Grid con técnicas de criptografía homomórfica* [45]

***Monitorización de coches policías, ambulancias y bomberos desde la nube***

Otra de las maneras para aunar la expansión tecnológica y el aumento de la calidad de vida es mediante la creación de las Smart Cities, un sector transversal y en auge que requiere de compartición de información sensible hacia distintos medios.

Una de las aplicaciones que se plantean cuando se implanten Smart Cities es la compartición de información entre las ambulancias y las víctimas de un altercado. Se entiende como compartición de información a la ruta más efectiva que deben tomar las ambulancias, policías, bomberos, etc para llegar al destino.

Un criminal puede realizar un ataque para situarse en el medio de las comunicaciones para interceptar el tráfico, esto afecta directamente a la disponibilidad de los datos pudiendo no reenviarlos desde los cibercriminales hasta el nodo final (coches de policía, ambulancias,

transporte, etc), y a la integridad por la modificación de ellos pudiendo dirigirse a otro lugar [42].

La criptografía homomórfica puede solventar las brechas de información en este tipo de aplicaciones ofreciendo una capa de protección a la hora de enviar los datos. Desde el momento en el que se detecta un accidente, el servidor de la Smart City crea un nodo que envía información de la ruta a los departamentos correspondientes.

Sin embargo uno de los problemas a los que se somete la criptografía homomórfica es la retransmisión de los datos con una latencia mínima, los datos deben computarse en tiempo real en función de la ubicación del vehículo, el tráfico existente en la calle y demás factores.

Las operaciones que se necesitan computar con criptografía homomórfica son sencillas, principalmente calculo de estadísticas descriptivas. Sin embargo unas de las operaciones más costosas es la comparación. Esta comparación se puede hacer "a ciegas", siempre que se conozca el tamaño (en bits) de los datos, mediante redes de ordenamiento. Ordenando una mezcla de dos conjuntos se puede saber si tienen un dato en común porque, tras la ordenación, los datos repetidos serán adyacentes.

Actualmente las compañías que están desarrollando Smart Cities son los principales clientes en este tipo de aplicaciones en países como Dubai, Amsterdam o Viena.

***Primitivas homomórficas para operaciones sobre sistemas ciber físicos***

Una de las múltiples aplicaciones que tienen los sistemas de control en la época actual con el aumento de la tecnología y las Smart Cities son los sistemas ciber físicos: sistemas que integran capacidades de computación, almacenamiento y comunicación junto con capacidades

de seguimiento y/o control de objetos en el mundo físicos, orientados a las redes eléctricas inteligentes o a la asistencia sanitaria.

Actualmente las aplicaciones sobre esquemas FHE (Fully Homomorphic Encryption) están restringidas por problemas de rendimiento y el gran tamaño de texto cifrado. Por ello es necesario diseñar y desarrollar nuevos esquemas sobre SHE (Somewhat Homomorphic Encryption) debido a su gran rendimiento, teniendo siempre en cuenta que soporta realizar un determinado número de operaciones hasta un límite (y que si se sobrepasa se vuelve indescifrable).

Sobre los esquemas que se desarrolla y analiza el rendimiento se encuentran NLV, FV y YASHE (Yet Another Somewhat Homomorphic Encryption) [46]. De los esquemas descritos, YASHE supera los dos anteriores en número de operaciones. YASHE es una mejor opción cuando el tamaño del texto cifrado importa, como muestra la evaluación del rendimiento realizada con la librería NTL [47].

Como conclusión, los esquemas SHE ofrecen esquemas rápidos con un rendimiento favorable sobre las principales operaciones homomórficas preservando la confidencialidad e integridad sobre los datos tratados.

**Análisis predictivo**

Cuando hablamos de análisis predictivo, machine learning, clasificación, etc; estamos hablando principalmente de grandes volúmenes de datos que permiten definir el comportamiento de algo. Es decir, estamos hablando de un sistema que, si tiene un fallo de seguridad, puede poner en grave riesgo la confidencialidad de los elementos que modela.

Si a esto le sumamos que estos datos pueden ser biométricos, de salud, o de sistemas críticos, incrementar las medidas de seguridad al máximo se convierte en un requisito del sistema. La criptografía homomórfica encaja a la perfección en este modelo, y aunque aún puede que no haya alcanzado el nivel de madurez óptimo, hay varias investigaciones muy esperanzadoras.

En el contexto del congreso IDASH [48] se desarrollan varias competiciones relacionadas con el análisis seguro de información genómica. En cada track de la competición se insta a los participantes a resolver una serie de problemas, y se les entrega un conjunto de datos para que prueben sus soluciones. Navegar por los resultados de estas pruebas no sólo nos permite obtener una fotografía del estado de la criptografía homomórfica en cada momento (puede verse la evolución que ha habido entre la costosa solución implementada en 2017 por investigadores de Microsoft [49] y la implementación ganadora en 2019 basada en TFHE-Chimera [50]. También certifica que es una solución cada vez más viable para ser aplicada en escenarios reales cuando el riesgo al que estén expuestos los activos de información así lo requieran.

Por último, en [51] muestran cómo se puede utilizar criptografía homomórfica para entrenar y probar modelos de machine learning. Aunque los modelos que utilizan son relativamente simples, la prueba de concepto validan la criptografía homomórfica, al menos en algunos casos, como una alternativa que aporta privacidad al procesamiento de datos.

## Base teórica

A continuación mostraremos someramente los elementos matemáticos que sustentan la criptografía homomórfica. Se ha tratado de simplificar estos conceptos lo máximo posible, sin

que por ello sean inexactos, con el objetivo de que cualquier lector sea capaz de interpretarlos teniendo una pequeña base previa.

## ¿Qué es un homomorfismo?

Se conoce como homomorfismo a la relación que existe entre dos espacios matemáticos, que permite la correspondencia entre sus elementos tras aplicarles una función. Es decir: que el resultado de aplicar una función entre dos o más elementos de un grupo operados entre sí, equivale a aplicar la función a sus elementos correspondientes del otro grupo y operar posteriormente los resultados.

El hecho de que las funciones de cifrado no sean más que funciones matemáticas (destinadas a proteger la confidencialidad de un código numérico) permite aplicar este principio algebraico a la criptografía, obteniendo como resultado la criptografía homomórfica.

No todas las funciones van a contener homomorfismos con cualquier operación. A continuación mostraremos una función que lo ofrece con el producto, y otra que lo ofrece con la suma.

### *Homomorfismo en el logaritmo discreto*

El logaritmo discreto es uno de los principales problemas utilizados como función trampa a la hora de cifrar. Su fuerza reside en la complejidad computacional que tiene obtener el logaritmo de un número para una base concreta.

El algoritmo de cifrado de RSA presenta homomorfismo en el producto precisamente por estar basado en este problema:

- No nos detendremos mucho en la generación de la clave: RSA construye sus claves en base a $p$ y $q$ primos, $n = p * q$, $\varphi(n) = (p-1)(q-1)$, $e$ coprimo con $\varphi(n)$ y $d$ congruente con $e$ en base $\varphi(n)$
- Se cifra el mensaje $m$, generando el texto cifrado$c = m^e \ mod \ n$
- Se descifra el criptograma$c$, generando de nuevo el mensaje $m = c^d \ mod \ n$
- Con un $m_1, m_2$ cifrados con el mismo par de claves:
  - $c_1 = {m_1}^e \ mod \ n$
  - $c_2 = {m_2}^e \ mod \ n$
- Al multiplicar c1*c2, tenemos c3 tal que:
  - $c_3 = c_1 * c_2$
  - $c_3 = ({m_1}^e \ mod \ n) * ({m_2}^e \ mod \ n)$
  - $c_3 = (m_1 * m_2)^e \ mod \ n$
- Y obtendríamos $m_3$ (descifrando $c_3$):
  - $m_3 = {c_3}^d \ \ mod \ n$
  - $m_3 = ((m_1 * m_2)^e)^d \ \ \ \ mod \ n$
  - $m_3 = m_1 * m_2$
- Por lo tanto:
  - $desc(c_1 * c_2) = decs(c_1) * desc(c_2)$

***Homomorfismo en la asunción del residuo compuesto***

Otro gran problema es el de la residualidad compuesta. Este problema se basa (a grandes rasgos) en la dificultad de encontrar residuos compuestos (i.e. un número $z$ tal que, para cualquier $y$, $z = y^n \ mod \ n^2$)[52], [53].

El criptosistema de Paillier (basado en este problema) es homomórfico en la suma:

- Teniendo una clave pública $(n, g)$, una clave privada $(l, u)$ y un valor aleatorio $r$, el cifrado se define como
    - $c(m) = g^m * r^n \, mod \, n^2$
- El producto de dos elementos $c_1$,$c_2$ cifrados con Paillier
    - $c_3 = c_1 * c_2$
    - $c_3 = (g^{m_1} * r_1^n \, mod \, n^2) * (g^{m_2} * r_2^n \, mod \, n^2)$
    - $c_3 = (g^{m_1+m_2}) * (r_1 * r_2)^n mod \, n^2$
- Por lo tanto:
    - $desc(c_1, c_2) = desc(c_1) + desc(c_2)$

El producto entre elementos del espacio de valores generados por el criptosistema de Paillier, equivale a la suma dentro del espacio de enteros en base $n$.

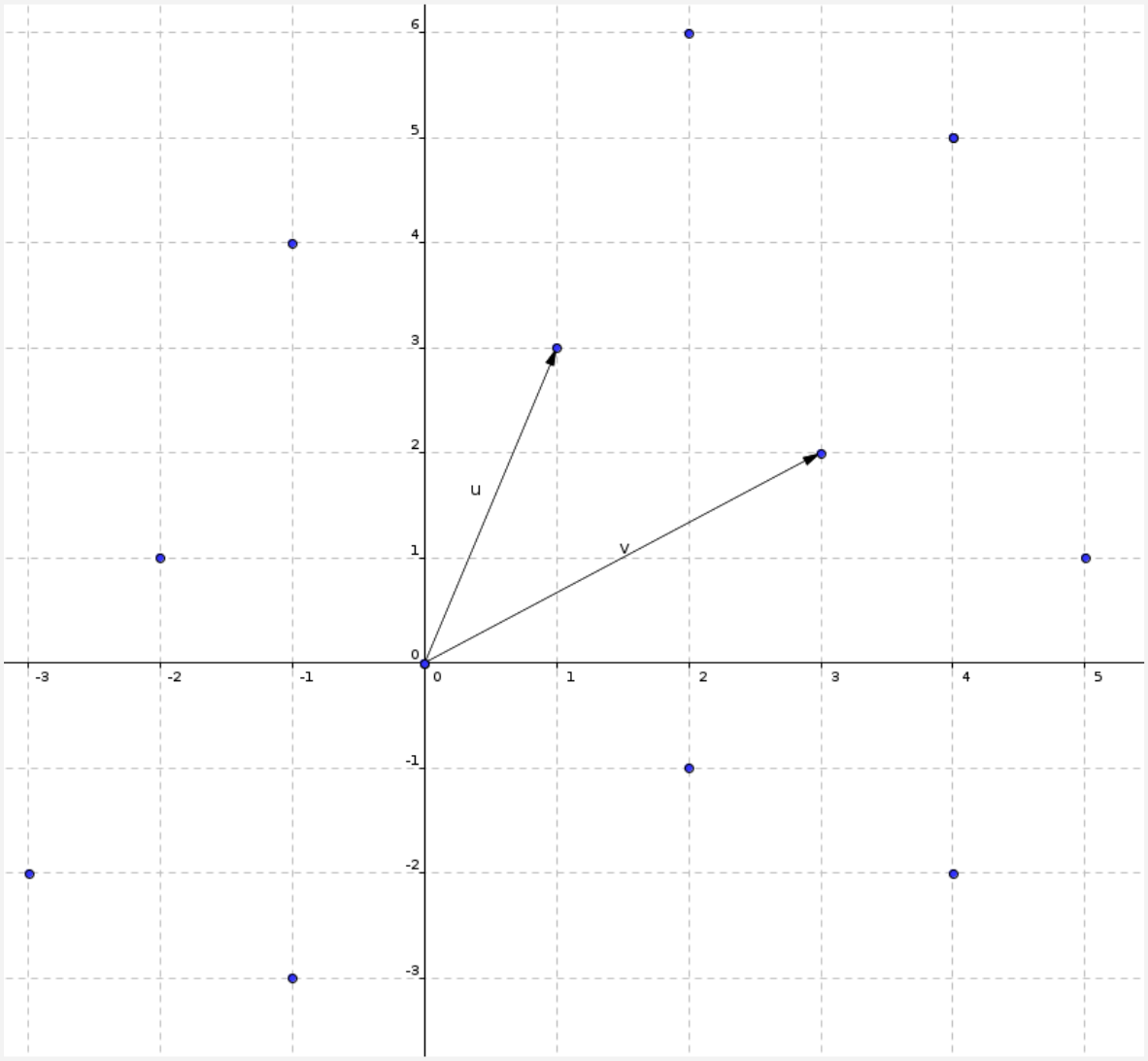
**Lattice based encryption**

Como hemos visto, la seguridad de los criptosistemas se basa en que hay determinados problemas muy complejos para resolver por un ordenador. Es decir, que a medida que aumenta linealmente el tamaño de la clave, su dificultad aumenta mucho más rápido. En términos computacionales, para que un problema sea válido como función trampa en criptografía [54], debe ser de clase NP (no resoluble en tiempo polinómico)[55].

La última gran familia de problemas de este tipo que veremos, y en la que se van a basar la mayor parte de los algoritmos de criptografía homomórfica, serán los problemas basados en retículos (lattices en inglés).

Los retículos son estructuras algebraicas similares a espacios vectoriales discretos (no continuos) que pueden ser generados con una base (un conjunto finito de sus elementos que,

combinados, pueden representar todos los elementos del espacio). Así por ejemplo, con una base formada por los elementos $u = (1, 3)$ y $v = (3,2)$ podrían formarse los siguientes valores: $w = (4,5)$ (resultado de sumar $u$ y $v$), $x = (-7, 0)$ (suma de $2u$ y $-3v$), etc.




*Retículo generado por los vectores u y v*

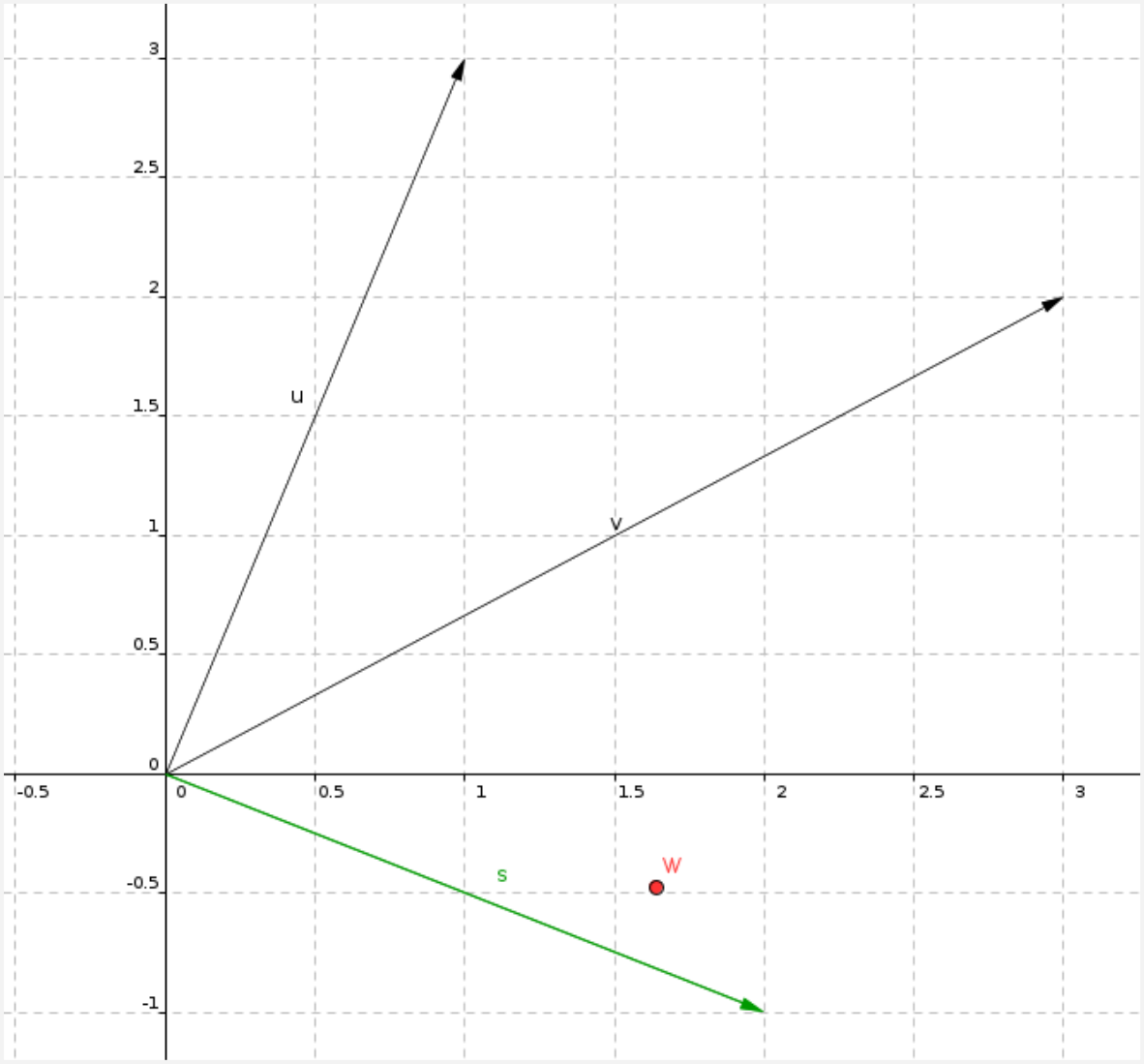
Los dos problemas más representativos son el problema del vector más corto (SVP por sus siglas en inglés) y el problema del vector más cercano (CVP).

***SVP***

Consiste en encontrar un vector cercano al origen del espacio (i.e. el punto que equivale a 0) dada una base larga. Para encontrarlo, por ejemplo, podrían realizarse combinaciones de la base y quedarse con el valor más cercano a cero, sin ser cero.

***CVP***

Buscar la combinación de elementos de una base larga que resulte en el punto del espacio más cercano a un punto dado.




*Ejemplo de CVP (combinación de u y v más cercana al punto w)*

Pese a que la resolución de ambos problemas (SVP y CVP) puede parecer trivial, computacionalmente son muy complejos, pues el tiempo de resolución aumenta drásticamente al aumentar el espacio (e.g. con vectores muy grandes, muchas dimensiones, etc).

**LWE**

Un caso particular de problemas de retículos es el problema del aprendizaje con errores (Learning With Errors, LWE). Este problema plantea la dificultad de determinar qué variables se han introducido en una función polinómica si se introduce un error aleatorio.

El ingenioso método de Regev es un criptosistema de clave pública (cuya forma más básica es relativamente simple) que utiliza este problema para "envolver" los bits del mensaje, de tal forma que sólo pueda revelarse conociendo la clave secreta.

El procedimiento, utilizando el anillo $\mathbb{Z}_q$ (enteros en base $q$) sería el siguiente.

***Generación de claves***

- Se elige un conjuntos de valores aleatorios $A$ y un valor aleatorio $s$
- Se calculan los valores $b_i$ de otro conjunto, al que llamaremos $B$, de la siguiente forma:
    - $b_i = (a_i * s + e) \, mod \, q$
- Siendo $e$ un valor de error aleatorio (escogido de una distribución aleatoria) muy pequeño en comparación con $q$, y distinto para cada $b_i$.
- La clave pública será $(A, B)$, y la clave privada será $(S)$

***Cifrar un bit***

- Para cifrar un bit $x$ con la clave pública se elegirá un subconjunto de valores de $A$ (e.g. $[a_3, a_5, a_9]$), y su correspondiente conjunto de valores de $B$ (i.e. para los valores del ejemplo de $A$, serían. $[b_3, b_5, b_9]$).
- Se calculan los valores $u$ y $v$:
  - $u = Suma(A) = a_3 + a_5 + a_9$
  - $v = Suma(B) = b_3 + b_5 + b_9$
- Se calcula el cifrado de un bit $x$ (con valor 0 o 1) de la siguiente forma:
  - $c_1 = u \; mod \; q$
  - $c_2 = v + (x * (q/2)) \; mod \; q$
- El texto cifrado sería el par $(c_1, c_2)$

***Descifrar***

- Ahora, sólo quien tenga la clave secreta $s$ podrá descifrar el texto:
  - $d = c_2 - c_1 * s$
  - $c_2 = (b_3 + b_5 + b_9) + x * (\frac{q}{2})$
  - $b_n = a_n * s + e$
  - $c_2 = (a_3 * s + a_5 * s + a_9 * s) + 3e + x * (\frac{q}{2})$
  - $c_1 * s = (a_3 + a_5 + a_9) * s = (a_3 * s + a_5 * s + a_9 * s)$
  - $d = x * (\frac{q}{2}) + 3e \simeq x * (\frac{q}{2}) + e$
- Al ser el valor de $e$ despreciable en comparación con q, el resultado de este cálculo sería
  - $d \simeq \frac{q}{2} \rightarrow x = 1$
  - $d \ll \frac{q}{2} \rightarrow x = 0$

De esta forma tendríamos un criptosistema de clave pública basado en el problema LWE, que además cumple con las propiedades homomórficas necesarias para computar. La clave para que estas propiedades sean fructíferas (en lo que a criptografía homomórfica se refiere) residirá en la elección del parámetro A, concretamente en la elección del tipo de parámetro. Optar por un entero, una matriz diagonalizable o elementos con propiedades de anillo algebraico (como un anillo de polinomios, de ahí la R de Ring cuando hablamos de RLWE) determinará el recorrido que podremos darle al algoritmo, y sus propiedades.

En [56], [57] puede verse un ejemplo dinámico del funcionamiento del sistema.

**Bootstrapping**

La propiedad que da tanto valor al cifrado mediante aprendizaje con errores es que presenta homomorfismo tanto en la suma como en el producto. A grandes rasgos, los resultados de aplicar estas operaciones serían:

- En la suma:
    a. $c_3 = c_1 + c_2 = (c_{11} + c_{21}, c_{21} + c_{22})$
    b. $c_{32} - c_{31} * s \simeq x_1 * (\frac{q}{2}) + x_2 * (\frac{q}{2}) + 2 * e$
    c. $(x_1 + x_2) * (\frac{q}{2}) + 2 * e$
- Bajo determinadas condiciones, y teniendo siempre en cuenta que la operación en el espacio del texto cifrado, no va a ser siempre literalmente la misma operación que en espacio del texto plano, podríamos ver el producto así:
    a. $c_3 = c_1 * c_2 = (c_{11} * c_{21}, c_{21} * c_{22})$
    b. $c_{32} - c_{31} * s \simeq x_1 * (\frac{q}{2}) * x_2 * (\frac{q}{2}) + e^2$
    c. $(x_1 * x_2) * (\frac{q}{2}) + e^2$

Como se puede apreciar, la operación entre valores cifrados con LWE hace que el ruido aumente (notándose especialmente en el producto). Al basar su funcionamiento en que, en el último paso, podremos deshacernos fácilmente del error porque es despreciable con respecto a $q$, si este ruido aumenta hasta cierto punto no nos será posible descifrar.

La técnica de bootstrapping de Gentry trata de eliminar este problema introduciendo la operación *Recrypt*. Esta operación utiliza unas claves especiales (derivadas de la clave privada) para eliminar el ruido cuando se opera. Así, una operación (×), pasaría a ser algo así como:

- $\times\_boostrapped = encrypt(decrypt(c_1 \times c_2))$

Al hacer ese descifrado intermedio (se codifica como un proceso interno de la operación, realmente en ningún momento se llega a descifrar), se elimina el ruido de la operación (a x b), y la siguiente operación comenzaría limpia. Pero con la operación de bootstrapping se introduce un nuevo cálculo, que además tiene un gran coste de eficiencia.

***Ejemplo de bootstrapping***

Para realizar esta técnica, con un algoritmo de cifrado $f$, se podrían seguir los siguientes pasos:

1. Junto con la clave pública ($pk_1$), se distribuye una versión cifrada (con criptografía homomórfica) de la clave privada $sk_{1x}$tal que $sk_{1x} = f(sk_1, pk_1)$.
2. Al operar $+$ entre dos textos cifrados con $pk_1$a los que llamaremos $c_1$y $c_2$ (resultado de cifrar los mensajes en claro $m_1$y $m_2$respectivamente), obtendríamos un texto cifrado $c_3$, tal que $c_3 = f(m_1 + m_2 + e)$.

Para eliminar el error $e$, podría implementarse la operación de *bootstrapping* de la siguiente manera:

3. Ciframos $c_3$(que en este caso sería igual que $f(m_1 + m_2 + e, pk_1)$), obteniendo $c_{3x} = f(\, f(m_1 + m_2 + e, pk_1), pk_1\,)$
4. Aquí es donde entra en juego la clave secreta cifrada $sk_{1x}$. Aplicamos la operación de descifrado entre $c_{3x}$y $sk_{1x}$, de tal forma que obtendríamos $c_4$tal que:
    a. $c_4 = f^{-1}(c_{3x}, sk_{1x})$
    b. $f^{-1}(c_{3x}, sk_{1x}) = f(\, f^{-1}(c_3, sk_1)\,) = f(\, m_1 + m_2\,)$

La operación de descifrado intrínsecamente elimina el error que haya podido acumular el texto cifrado, por lo tanto $c_4$sería equivalente a $c_3$pero con el contador de error reseteado.

Se puede apreciar que, para poder implementar esta técnica, es necesario que el esquema permita realizar más de una operación con el texto cifrado. Cuando un esquema de criptografía homomórfica cumpla esta condición, podremos hablar de un algoritmo bootstrapable, por ende perteneciente a la familia de los algoritmos FHE.

## Algoritmos

En base a la capacidad que tengan los algoritmos para la computación (tipo de operaciones que puedan implementarse, profundidad computacional, etc) los dividiremos en tres categorías, modelo que guarda cierta relación con las generaciones que hemos visto anteriormente:

- Partially Homomorphic Encryption
- Somewhat Homomorphic Encryption
- Fully Homomorphic Encryption

### Partially Homomorphic Encryption

Conocemos como esquemas Partially Homomorphic Encryption (PHE) a aquellos que bien por casualidad o de forma intencionada, presentan homomorfismos, pero estos no son lo suficientemente versátiles como para realizar cómputos. Tal es el caso del algoritmo RSA: es homomórfico con respecto al producto (ver demostración), pero no con respecto a la suma. El sistema de Paillier nos ofrecería el ejemplo contrario, siendo homomórfico en la suma, pero no en el producto.

Finalmente, el sistema implementado por Bonneh, Goh y Nissim [58] trata de ir más allá, pero se queda a medio camino. En su sistema, implementado como un sistema homomórfico ex profeso, pueden evaluarse tanto la suma como el producto, pero si esta última operación se repite más de una vez, el resultado se corrompe. Aunque sus limitaciones no permiten categorizarlo junto a los sistemas que veremos a continuación, este algoritmo sienta un precedente esperanzador en el campo de la criptografía homomórfica.

**Somewhat Homomorphic Encryption**

Los algoritmos Somewhat Homomorphic Encryption (SHE) permiten la realización de cualquier operación lógica (es decir, permiten la suma y el producto bit a bit del texto cifrado) pero acotan la capacidad de cómputo de alguna forma a saber:

- Impidiendo establecer arbitrariamente el circuito, porque genere incongruencias en los datos; como las relacionadas con órdenes de magnitud.
- Limitando el número de operaciones realizables, porque la computación genere ruido en los datos y el esquema no sea capaz de eliminarlo.

Debido a esta cota, en la literatura podrá encontrarse ocasionalmente que estos sistemas son categorizados como esquemas de cifrado nivelado (Leveled Encryption Schemes, o LES),

manteniéndose la categoría SHE exclusivamente para algoritmos como el de Bonneh, Goh y Nissim [58].

***Sistemas basados en control de ruido***

Se puede operar con esquemas como BGV [59] y BFV [60] determinando el umbral, y calculando el aumento de error que supone cada operación. De esta forma se tendrá presente en todo momento cuántas operaciones se podrán realizar antes de que este error sea inasumible.

Además, como mencionan en [46], existen otros esquemas SHE como NLC, FV y YASHE que se basan en el esquema BGV.

Con estos primeros sistemas se busca sortear la operación de bootstrapping (que es extremadamente lenta) a costa de limitar la profundidad de cómputo. Algunos esquemas también incluyen una nueva operación para reducir el ruido, que aunque no es tan eficaz con el bootstrapping, es mucho más eficiente: la realinearización.

***Sistemas de aproximación y truncado***

El sistema CKKS ofrece una aproximación alternativa a la de la monitorización del ruido [61]. Mediante la división por lotes de los textos cifrados, controla el ruido aplicando una operación similar al truncado que elimina los valores menos significativos (en los que el ruido se manifiesta más notablemente). De esta forma, tras cada operación, el ruido se reduce pero se pierde precisión hasta corromper el resultado.

**Fully homomorphic encryption**

Finalmente, los algoritmos que permiten desarrollar cualquier cómputo con el texto cifrado, sin limitar el número de operaciones, son conocidos como Fully Homomorphic

Encryption (FHE). Más allá de las particularidades que pueda haber con respecto a la eficiencia del esquema, o de la forma de operar con ellos, un esquema es FHE cuando a efectos prácticos la capacidad de cómputo no tiene ningún límite que no tenga la computación sobre el texto plano: es decir, se pueden hacer exactamente las mismas operaciones con el texto sin cifrar, y con el texto cifrado.

El primer sistema FHE eficiente viene de nuevo de la mano de Gentry, que junto a Sahai y Waters diseñan un sistema (GSW [17], por sus apellidos) en el que los datos son codificados en matrices que se comportan como autovectores, de forma que se puede reducir la complejidad a operaciones aritméticas mucho más simples sin reducir los niveles de seguridad.

Tras la aparición de este algoritmo, conceptualizado por la comunidad científica como el santo grial de la criptografía homomórfica, los límites a la computación dejan de ser un problema, y los investigadores comienzan a invertir sus esfuerzos en la búsqueda de técnicas más eficientes.

Actualmente el culmen de estas investigaciones lo alcanza TFHE [62] con capacidad de evaluar una operación lógica en 13 milisegundos (frente a los 6 minutos que podía tardar inicialmente la operación de bootstrapping de Gentry [63]), pero sigue quedando muy lejos de las $1{,}417e9$ (nueve órdenes de magnitud por encima, más de un millón de operaciones) operaciones de coma flotante que podría realizar un procesador actual en el mismo tiempo [64].

Debido a que el coste computacional del bootstrapping sigue siendo elevadísimo, y no es necesaria en una infinidad de casos se está trabajando en el desarrollo de soluciones que combinen esquemas SHE y apliquen el bootstrapping cuando sea necesario. Varios miembros del equipo de desarrollo de TFHE llevan la vanguardia en este ámbito, y trabajan en un framework conocido como Chimera [65].

## Seguridad

Ante todo estamos hablando de criptosistemas, es decir, sistemas destinados a proteger la seguridad de la información (concretamente, en nuestro caso, la confidencialidad). Si estos algoritmos permitiesen operar en un dominio cifrado, pero solo diesen una falsa sensación de seguridad, nos toparíamos con que estamos trabajando en una forma estupenda de minar la eficiencia de nuestros sistemas sin aportar nada.

A continuación veremos las características que validan la tecnología homomórfica, y cuales son sus principales carencias.

### Asunciones de LWE

Regev demuestra que la dureza del problema de LWE es similar a la de SVP [15]. Por lo tanto la asunción en la que se basa la seguridad de los algoritmos sería correcta (SVP es un problema NP complejo) [66]. Sin embargo, la seguridad de un criptosistema contiene numerosos eslabones.

Asumiendo que el principio de la cadena (la base matemática) va a ser seguro, y que el final (el usuario) va a ser como mucho, ligeramente confiable, mostraremos cuales son las fortalezas y debilidades de los elementos intermedios.

### Seguridad semántica

#### *IND-CPA*

Los algoritmos comentados son seguros en cuanto a lo que ataques de texto plano elegido se refiere. Es decir: un adversario [67] que enviase dos textos planos diferentes a un

oráculo de cifrado, no podría distinguir cuál de los dos cifrados resultantes pertenece a cada mensaje.

Sin embargo, hay que ser muy cuidadosos con las implementaciones. En [42] analizan esta propiedad sobre un algoritmo de ordenación basado en pivotes y el algoritmo Quick Sort, y muestran cómo pese a que el segundo es seguro, el primero filtra información ante determinadas entradas.

***IND-CCA***

Parece que la propiedad homomórfica de los algoritmos basados en LWE tiene conflicto con respecto a ataques de texto cifrado elegido. Como veíamos en la demostración, la operación de descifrado consiste (a grandes rasgos) en multiplicar el texto cifrado por la clave privada. Si en este escenario académico el texto cifrado fuese igual a 1, esta operación devolvería la clave privada. Si un atacante pudiese enviar textos cifrados al sistema y tener acceso a un solo texto descifrado, podría averiguar la clave.

Por lo tanto, es muy importante que nunca se permita a usuarios externos descifrar, o acceder a textos descifrados, bajo el riesgo de filtrar la clave secreta. Un ejemplo de caso aparentemente seguro, que evidencia la necesidad de implantar controles estrictos en el flujo de datos es el siguiente:

1. El usuario cifra un vector y la manda a un servidor para que lo ordene
2. El servidor procesa la información
3. El servidor devuelve la información procesada y el usuario lo descifra
4. El usuario utiliza este vector para ordenar otra lista de datos que sube al mismo servidor, esta vez en claro

Pese a que parece una operación inocente, si en el tercer paso el servidor devolviese un texto cifrado determinado, podría ser capaz de inferir la clave cuando obtuviese la información (otra información, que en principio sería completamente distinta) del cuarto.

En [68] demuestran que, por ejemplo, BFV no es seguro IND-CCA.

### Maleabilidad

Por definición, en criptografía homomórfica trabajaremos con cifrados maleables (es decir, que permiten manipular el mensaje desde un texto cifrado). Por lo tanto, esquemas de cifrado autenticados no parecen tener mucho sentido en este paradigma.

Cuando sin embargo se precise mantener la autenticación del mensaje, porque por ejemplo, vaya a estar en tránsito, se pueden aplicar capas adicionales de cifrado (como por ejemplo, TLS).

### Parámetros de seguridad

Por último, para determinar los parámetros de seguridad que debe tener nuestra implementación, podemos acudir a la documentación generada por el estándar. En [5] muestran una referencia de cuáles deben ser estos parámetros en función del nivel (en bits) de seguridad deseado, documentando incluso un apartado destinado a modificar dichos parámetros ante un eventual panorama post-cuántico.

## Computación

En un paradigma como el que estamos tratando, en el que no se pueden comparar los datos ni modificar el flujo de ejecución, porque las variables están cifradas, deben desarrollarse mecanismos lógicos que permitan implementar los algoritmos clásicos.

Para ello se partirá de las instrucciones más básicas (instrucciones lógicas a nivel de bit [69]), y se implementarán bloques operacionales que permitan construir un sistema computacional completo. El enfoque introducido por Rass [70] nos muestra la computación en criptografía homomórfica como el diseño de programas para una máquina de Turing ciega.

Cuando comenzamos hablando de que el éxito de la criptografía homomórfica residía en poder sumar y multiplicar, estábamos introduciendo que todas las construcciones que se desarrollen partirán de cero: de puertas lógicas XOR y AND.

Existen sólo algunos estudios centrados en la traducción de los algoritmos clásicos al paradigma homomórfico, como [71]. En este trabajo desarrollaremos la forma de construir elementos para las principales aplicaciones de la computación y trataremos de identificar las principales referencias que permitan profundizar posteriormente al lector interesado en ello.

**Ordenación**

Una de las operaciones más interesantes, por subyacer en la mayoría de las implementaciones, es la ordenación. Los algoritmos de ordenación incluyen, por definición, sentencias de comparación datos. Esta comparación se puede hacer “a ciegas”, siempre que se conozca el tamaño (en bits) de los datos, mediante redes de ordenamiento [72].

Estas redes de ordenamiento son circuitos diseñados de forma que el paso de los valores de entrada a través de puertas lógicas genere un resultado equivalente al de ordenar los datos mediante sistemas con estructuras de control iterativas.

***Ordenación bubble sort***

En [73] por ejemplo, indican que es posible implementar el algoritmo Bubble Sort siempre que se pueda implementar una circuito de resta y un circuito de transposición de valores.

Para ello, y siguiendo el principio expuesto anteriormente de trabajo por bloques, desarrollaríamos:

- Circuito de resta:
    - La resta se compone por una operación de suma y la operación de complemento a 2.
    - La suma, bit a bit, es trivial. Simplemente se necesita la puerta lógica XOR (equivalente a suma en $\mathbb{Z}_2$) y una puerta AND para tratar el acarreo. El sumador completo puede desarrollarse para un tamaño fijo de palabra concatenando sumadores.
    - El complemento a dos a su vez se construye como una negación (XOR de todos los valores de la entrada con unos) a la que se suma 1.
        - $Resta\ (a, b)\ =\ Suma(a, comp_2(b))$
- Circuito de transposición:
    - Un circuito de transposición de un bit, consiste simplemente un par de circuitos multiplexores que reciben como entrada dos bits $(a, b)$. El valor selector será el bit más significativo de la resta $(a - b)$ para el primero, y el mismo bit negado para el segundo.

- Así, si $a >= b$ el bit más significativo será 0 (este bit indica el signo en complemento a 2) y el resultado a la salida será $(a, b)$; y si $b > a$, al ser $a - b < 0$, obtendremos $(b, a)$

Con estos elementos, la implementación del algoritmo bubble sort es trivial.

En [74] puede encontrarse un estudio sobre la eficiencia de los algoritmos de ordenación, junto con una propuesta llamada *Lazy sort*.

El anexo III muestra ejemplos de operaciones implementadas en circuitos lógicos.

**Bases de datos**

A la hora de utilizar criptografía homomórfica en servicios en la nube otro gran reto es el del almacenamiento de la información estructurada. Aunque para una gran parte de los consumidores de estos servicios el riesgo de una brecha de seguridad o una filtración por parte del prestador de servicios pueden ser riesgos asumibles, para otros la delegación del control de los datos a este tipo de arquitecturas no puede basarse en la confianza. Para estos segundos la criptografía homomórfica es una buena solución, pues permitirá mantener estos datos almacenados de forma segura y funcional.

Estando los datos cifrados no será necesaria la implementación de medidas de auditoría (pruebas de posesión, de borrado, etc) [75], y se podrá confiar plenamente en la confidencialidad de la información siempre que los algoritmos de cifrado sean seguros. Utilizando cifrado homomórfico, además, también se podrán realizar búsquedas, actualizaciones de los datos, o incluso borrados selectivos.

Ante la diversidad de sistemas de bases de datos existente hoy en día, centraremos nuestro estudio en bases de datos de tipo relacional manejadas mediante consultas SQL.

### *Antecedentes*

Las primeras implementaciones para el uso de bases de datos cifradas estaban basadas en algoritmos propios de la criptografía clásica, combinando el almacenamiento en el servidor con la ejecución de la lógica en el cliente. Algunas aproximaciones como CryptDB mantienen en el servidor los datos cifrados celda a celda (con nombres de tablas y columnas en claro), y es en el cliente donde se cifran los parámetros de las consultas y se procesan los resultados intermedios. Una sola query normalmente se compone de varias operaciones distintas basadas en operaciones algebraicas, así que tras cada pequeña operación se devuelve el resultado intermedio al cliente para que lo procese y realimente el sistema del servidor.

Aunque en estas aproximaciones los nombres de tablas y columnas puedan estar cifrados, hablamos de textos en claro porque realmente este cifrado actúa como método de ofuscación, pero si el atacante averigua el nombre cifrado de una columna podrá utilizarlo como diccionario para inferir qué operación se está realizando en el futuro.

A continuación mostraremos la forma de implementar las principales operaciones que se realizan en una base de datos.

### *INSERT*

La operación de inserción será trivial, consistiendo exclusivamente en la anexión de datos cifrados a la base de datos.

### *SELECT*

Mientras que la operación de selección no condicional ($SELECT$ ... $FROM$ ...) es trivial, introducir condiciones en la selección ($WHERE$ ...) no lo es. La selección condicional requiere de los siguientes elementos:

- Implementar operación $MUX$

  Como hemos comentado anteriormente, es posible realizar una operación para elegir entre un valor u otro, en base a una condición. La resolución de la condición en este caso también es trivial, porque podemos implementar cualquier operación booleana mediante puertas lógicas, pero los operandos de entrada dejan de serlo de nuevo. Uno de ellos, como es lógico, será la entrada de la base de datos; pero para el otro tendremos que codificar un nuevo valor: el valor *null*.

- Valor null

  Este valor null debe codificarse como un valor único con significado propio, para evitar que los datos que se introduzcan (y que no se podrán validar, porque estarán cifrados) colisionen con él.

- Ejecución

  Mediante la utilización de valores *null*, la selección condicional devolverá una copia de la tabla en la que se ha realizado la búsqueda con los valores que no cumplan la condición anulados, y será el cliente quien deba desarrollar la lógica de borrado de elementos no válidos.

***UPDATE***

La operación de actualización se realizará también haciendo uso de la operación MUX. Para ello se recorrerán las entradas de la tabla y su nuevo valor será el mismo si no cumple la condición, o el nuevo sí la cumple.

Para realizar esta operación de forma segura, también será necesario que el resultado de la operación MUX incluya alguna medida de aleatorización que impida la detección de cambios en los datos.

***DELETE***

La operación de borrado será simplemente una operación de actualización en la que el valor introducido en caso de cumplir la condición sea null.

Este sistema plantea el problema de que, al no eliminarse las entradas, la base de datos pueda crecer indefinidamente. Para lidiar con él, puede definirse una rutina periódica de compactación en la que el cliente descargue todo su contenido, elimine los valores nulos y vuelva a subir la información (de nuevo aplicando medidas de aleatorización de resultados).

La selección de tabla y columna serán siempre una parte intrínseca de la condición introducida en la consulta. Es decir, no se recorrerán tablas concretas cuando se opere, sino datos completamente cifrados, introduciendo el nombre de la tabla y la columna dentro de los parámetros de la operación MUX, para poder mantener la confidencialidad de estos datos.

**AES HE Gentry**

Como hemos repetido en varias ocasiones, una de las principales causas por las que puede ser especialmente relevante la criptografía homomórfica es el cambio de modelo hacia la nube. Especialmente dentro de los entornos corporativos, es habitual el uso de herramientas

de compartición de datos bien entre empleados o entre sistemas (comunicación entre APIs, almacenamiento de ficheros, etc).

El envío de datos cifrados entre sistemas en la nube supone dos nuevos retos: el cifrado en la nube, de forma que se asegure que el contenido de la información nunca sea revelado; y el almacenamiento seguro de contraseñas, de forma que no se puedan utilizar sin un secreto adicional.

Los estándares de seguridad para este tipo de operaciones operan bajo el paradigma KEM-DEM [76], utilizando una sistema de criptografía simétrica para cifrar la información de forma eficiente, y un sistema de criptografía asimétrica para proteger la clave de una forma más segura. De esta forma también se evita la necesidad de almacenar las múltiples claves de cifrado de la información, almacenando únicamente las claves asimétricas.

Para la gestión de claves, actualmente, predomina el uso de dos tecnologías: HSM y KMS [77]. Utilizar este tipo de sistemas en línea requiere permitir el acceso a la red interna de la organización desde redes no controladas por la organización (como Internet), o establecerlos directamente en la nube y confiar en que el gestor en que se delega la administración del sistema actúe correctamente en todo momento. Cuando además, se utilizan para cifrar información, los datos tienen que quedar expuestos en algún momento, y el sistema sustituye toda la robustez de los procedimientos criptográficos por un modelo basado en control de accesos. En 2012 aparece una propuesta que puede poner solución a estos problemas: el modelo híbrido de Gentry.

Gentry diseña un sistema de evaluación de un circuito AES [78] empleando criptografía homomórfica. Este sistema tiene un enorme potencial, por ejemplo, para el establecimiento de

almacenes de claves en la nube de forma segura. Este sistema podría funcionar de la siguiente forma:

1. Un usuario con un texto plano m1 lo cifra utilizando criptografía homomórfica (con un par de claves $< hpk, hsk >$) y genera $hm_1$. También genera una clave aleatoria $s$, y la cifra con criptografía homomórfica generando $hs_1$.

2. Sube hm1 y hs1 a su sistema de cifrado en la nube. Este sistema cifra hm1 con AES, utilizando como clave $hs_1$. Lo devuelve ($hc_1$), y almacena en una base de datos $hs1$.

3. El usuario descifra la capa de criptografía homomórfica de $hc_1$ y obtiene el texto $c_1$ (m1 cifrado con AES). El usuario olvida la clave $s$.

Cuando el usuario desee descifrar su mensaje seguirá el siguiente procedimiento:

1. Cifra c1 con criptografía homomórfica (usando $hpk$), generando $hc_1$ y lo sube a la nube.

2. La nube descifra $hc_1$ con la clave $hs_1$ que tiene almacenada, y devuelve $hm_1$

3. El usuario descifra $hm_1$ usando $hsk$, y obtiene $m_1$

El usuario podría repetir este proceso una infinidad de veces, y mientras fuese capaz de almacenar el par $< hpk, hsk >$, podría cifrar y descifrar sus mensajes delegando la gestión y almacenamiento de las claves simétricas en la nube, sin que esta conociese en ningún momento la clave ni el contenido de los mensajes.

**Criptografía homomórfica en CPU**

Por último, si hablamos de computación, estamos en la obligación de "descender hasta el hierro" y documentar las propuestas quizá menos maduras, pero también quizá con mayor potencial: las relacionadas con la construcción de un computador homomórfico.

Existen distintas aproximaciones orientadas a trabajar cifrando los registros, las direcciones de memoria, etc; pero la parte más compleja de este trabajo es la construcción de una unidad aritmético-lógica capaz de procesar las instrucciones cifradas. Este es el caso de FURISC.

FURISC [79] es una propuesta de arquitectura en la que los programas están completamente cifrados con criptografía FHE. La propuesta está basada en el modelo URISC, un modelo con instrucciones reducidas (RISC) hasta el mínimo: una sola instrucción. En URISC sólo existe un código de operación que se aplica a registros y direcciones similar a la resta, que combinada es capaz de realizar todas las demás. De esta forma, aunque aumente el tamaño de los programas, se reduce el problema de la implementación a un problema ya resuelto (el del manejo de datos cifrados), salvando el principal escollo para la construcción de computadores homomórficos.

## Retos de la criptografía homomórfica

Una vez vistos sus componentes, y viendo sus potentes aplicaciones: ¿por qué no hemos convertido todos nuestros sistemas para que funcionen con criptografía homomórfica? Pues bien, veremos cómo todavía hay varios retos que afrontar antes de que estas tecnologías lleguen a un público masivo.

### Eficiencia

En [80] estudian la eficiencia de distintos algoritmos a bajo nivel, implementando un juego de instrucciones y evaluando el coste computacional que tiene desarrollar cada una de ellas con varios tamaños de palabra. Su análisis, si bien no es concluyente, pretende sentar las bases para un futuro estudio más completo.

Un estudio a alto nivel sí que permite ver el contraste con respecto a una implementación sin criptografía homomórfica. La investigación desarrollada en [81] busca probar la validez de las librerías de criptografía homomórfica desde el punto de vista de los límites computacionales, la eficiencia y el coste de implementación. Se desarrolla un sistema que implementa una librería de segunda generación y otra de tercera, y se analiza su viabilidad de cara la utilización de estas tecnologías en sistemas en producción. Las diferencias de eficiencia entre esquemas SHE y FHE son notables, pero las diferencias entre ambos y la computación en texto plano son abrumadoras.

Si bien se suele hacer referencia a los problemas de eficiencia de la criptografía homomórfica aludiendo a la lentitud de sus operaciones (los 6 minutos del bootstrapping, el medio segundo de la evaluación en FHEW, o los 13ms en TFHE) la realidad es que los problemas de eficiencia de la criptografía homomórfica vienen dados de la complejidad computacional: no poder controlar los flujos de ejecución puede hacer, por ejemplo, que la complejidad del mejor caso de una búsqueda lineal sea $O(N)$.

**Usabilidad**

Aunque existan iniciativas que tratan de acercar la criptografía homomórfica a un público más amplio (como *lattigo*), lo cierto es que actualmente son sistemas demasiado complejos para ofrecerlos como una solución más para el desarrollo. Teniendo en cuenta que además su objetivo es mejorar la seguridad del sistema, y una mala implementación puede

ponerlo en grave riesgo (por ejemplo, si algún texto descifrado puede acabar en manos de un atacante, como comentamos en el análisis de IND-CCA).

Ha sido una de los principales centros de atención en las reuniones de estandarización (organizadas por *Homomorphic Encryption Standardization*[1]). Pero precisamente, la falta de un estándar concreto (hay varias generaciones, varios esquemas por generación, etc) puede confundir a los desarrolladores, y mina la viabilidad de una posible inversión en estas tecnologías.

### Seguridad

Desde el punto de vista de la seguridad, más allá de los problemas ya analizados, los algoritmos son robustos. El principal problema es que (y aunque esto sea algo común a toda la criptografía) es fácil realizar una mala implementación que ponga en riesgo el sistema.

Por otra parte al ser sistemas mucho más complejos (con muchos más componentes matemáticos implicados) y modernos, han sido menos escrutados, por lo que es más fácil que puedan plantear fallas que, simplemente, no se conocen aún.

## Conclusiones

La vertiginosa evolución que ha tenido la criptografía homomórfica, y el alto interés que ha despertado en la comunidad científica (especialmente en el ámbito de las ciencias de la computación), sugiere no sólo que vaya a seguir desarrollándose a través de la innovación; sino

---

1 Standards Meetings. https://homomorphicencryption.org/standards-meetings/

que en cualquier momento puede dar un salto cualitativo importante (e.g. mediante la aparición de un esquema más eficiente).

En su estado actual, ya es aplicable a algunas soluciones. Si bien, tal vez carezca de la madurez necesaria como para sustituir completamente algunos sistemas, sí puede servir como un complemento estratégico. La decisión en cuanto a cómo implantarlo suele seguir la siguiente pauta:

- Suele utilizarse SHE para operaciones que requieran cierta eficiencia, siempre que se puedan desarrollar dentro de sus límites computacionales.
- Cuando la resolución del problema requiere cierta profundidad operacional, se utilizan soluciones FHE.

Sistemas como Chimera [65] pueden ser clave a la hora de realizar implementaciones comerciales, cuando no se consiga determinar el compromiso necesario entre eficiencia y complejidad; y la utilización de *toolchains*, un mecanismo para viabilizar su adopción.

En cualquier caso, el uso o no de la criptografía homomórfica debe valorarse a través del cristal de la eficacia: tiene que implantarse allí donde sea útil. Debe servir para solucionar problemas que eran irresolubles antes, no a hacer más complejos problemas que ya estaban solucionados.

Sin olvidar que debe primar la seguridad. En todos sus aspectos.

## Referencias


[1] I. T. L. Computer Security Division, «Privacy-Enhancing Cryptography | CSRC», *CSRC | NIST*, ene. 03, 2017. https://csrc.nist.gov/Projects/privacy-enhancing-cryptography (accedido abr. 29, 2020).

[2] I. T. L. Computer Security Division, «Post-Quantum Cryptography Standardization - Post-Quantum Cryptography | CSRC», *CSRC | NIST*, ene. 03, 2017. https://csrc.nist.gov/Projects/post-quantum-cryptography/Post-Quantum-Cryptography-Standardization (accedido mar. 20, 2020).

[3] P. Schwabe, «NewHope». https://newhopecrypto.org/ (accedido abr. 29, 2020).

[4] 14:00-17:00, «ISO/IEC 18033-6:2019», *ISO*. https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/77/67740.html (accedido abr. 29, 2020).

[5] M. Albrecht *et al.*, «Homomorphic Encryption Security Standard», HomomorphicEncryption.org, Toronto, Canada, nov. 2018.

[6] T. Mandt, M. Solnik, y D. Wang, «Demystifying the Secure Enclave Processor», Black Hat USA 2016.

[7] «Yao's Millionaires' Problem», *Wikipedia*. abr. 04, 2020, Accedido: abr. 07, 2020. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Yao%27s_Millionaires%27_Problem&oldid=949138359.

[8] D. Boneh, «The Decision Diffie-Hellman problem», en *Algorithmic Number Theory*, Berlin, Heidelberg, 1998, pp. 48-63, doi: 10.1007/BFb0054851.

[9] M. Naor y B. Pinkas, «Efficient oblivious transfer protocols», en *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, Washington, D.C., USA, ene. 2001, pp. 448–457, Accedido: mar. 30, 2020. [En línea].

[10] B. H. Bloom, «Space/time trade-offs in hash coding with allowable errors». Association for Computing Machinery, jul. 01, 1970, Accedido: mar. 28, 2020. [En línea]. Disponible en: https://doi.org/10.1145/362686.362692.

[11] A. López-Alt, E. Tromer, y V. Vaikuntanathan, «On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption», en *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, New York, New York, USA, may 2012, pp. 1219–1234, doi: 10.1145/2213977.2214086.

[12] R. L. Rivest y M. L. Dertouzos, «ON DATA BANKS AND PRIVACY HOMOMORPHISMS», 1978. /paper/ON-DATA-BANKS-AND-PRIVACY-HOMOMORPHISMS-Rivest-Dertouzos/c365f01d330b2211e74069120e88cff37eacbcf5 (accedido mar. 20, 2020).

[13] P. Paillier, «Public-Key Cryptosystems Based on Composite Degree Residuosity Classes», en *Advances in Cryptology — EUROCRYPT '99*, Berlin, Heidelberg, 1999, pp. 223-238, doi: 10.1007/3-540-48910-X_16.

[14] O. Regev, «New lattice-based cryptographic constructions». Association for Computing Machinery, nov. 01, 2004, Accedido: mar. 26, 2020. [En línea]. Disponible en: https://doi.org/10.1145/1039488.1039490.

[15] O. Regev, «On lattices, learning with errors, random linear codes, and cryptography», en *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, Baltimore, MD, USA, may 2005, pp. 84–93, doi: 10.1145/1060590.1060603.

[16] C. Gentry, «Fully Homomorphic Encryption Using Ideal Lattices», en *Proceedings of*

*the Forty-first Annual ACM Symposium on Theory of Computing*, New York, NY, USA, 2009, pp. 169–178, doi: 10.1145/1536414.1536440.
[17] C. Gentry, A. Sahai, y B. Waters, *Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based*. 2013.
[18] «OpenMined». https://www.openmined.org/ (accedido abr. 29, 2020).
[19] *Lattigo 1.0*. 2019.
[20] «Inpher | Secret Computing® & Privacy-Preserving Analytics», *inpher*. https://www.inpher.io (accedido abr. 30, 2020).
[21] «IXUP | Secure Data Collaboration», *IXUP*. https://ixup.com/ (accedido abr. 30, 2020).
[22] «Enveil | Encrypted Veil», *Enveil | Encrypted Veil*. https://www.enveil.com (accedido abr. 30, 2020).
[23] E. Crockett, C. Peikert, y C. Sharp, «ALCHEMY: A Language and Compiler for Homomorphic Encryption Made easY», en *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, Toronto Canada, ene. 2018, pp. 1020-1037, doi: 10.1145/3243734.3243828.
[24] *CEA-LIST/Cingulata*. CEA LIST, 2020.
[25] *encryptogroup/ABY*. ENCRYPTO, 2020.
[26] A. Azougaghe, M. Hedabou, y M. Belkasmi, «An electronic voting system based on homomorphic encryption and prime numbers», en *2015 11th International Conference on Information Assurance and Security (IAS)*, dic. 2015, pp. 140-145, doi: 10.1109/ISIAS.2015.7492759.
[27] Y. Zhao, Y. Pan, S. Wang, y J. Zhang, «An anonymous voting system based on homomorphic encryption», en *2014 10th International Conference on Communications (COMM)*, may 2014, pp. 1-4, doi: 10.1109/ICComm.2014.6866682.
[28] «REGLAMENTO (UE) 2016/ 679 DEL PARLAMENTO EUROPEO Y DEL C ONSEJO - de 27 de abril de 2016 - relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/ 46/ CE (Reglamento general de protección de datos)», p. 88.
[29] «Health Information Privacy», *HHS.gov*, ago. 26, 2015. https://www.hhs.gov/hipaa/index.html (accedido abr. 29, 2020).
[30] O. for C. Rights (OCR), «Anthem Pays OCR $16 Million in Record HIPAA Settlement Following Largest U.S. Health Data Breach in History», *HHS.gov*, oct. 15, 2018. https://www.hhs.gov/about/news/2018/10/15/anthem-pays-ocr-16-million-record-hipaa-settlement-following-largest-health-data-breach-history.html (accedido abr. 29, 2020).
[31] «Disease Gene Identification - an overview | ScienceDirect Topics». https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/disease-gene-identification (accedido abr. 29, 2020).
[32] «Security breach at MyHeritage website leaks details of over 92 million users», *Reuters*, jun. 05, 2018.
[33] «Matchmaker Exchange». https://irdirc.org/activities/task-forces/matchmaker-exchange/ (accedido abr. 29, 2020).
[34] J. W. Bos, K. Lauter, y M. Naehrig, «Private predictive analysis on encrypted medical data», *J. Biomed. Inform.*, vol. 50, pp. 234-243, ago. 2014, doi: 10.1016/j.jbi.2014.04.003.
[35] G. S. Çetin, W. Dai, Y. Doröz, W. J. Martin, y B. Sunar, «Blind Web Search: How far

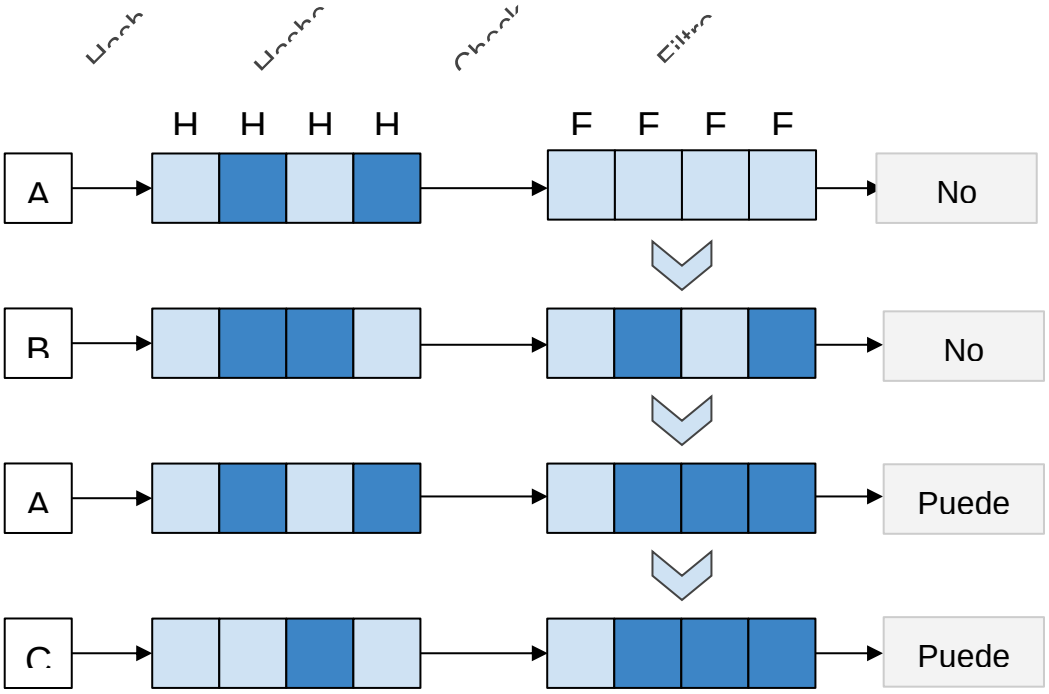
are we from a privacy preserving search engine?», 801, 2016. Accedido: abr. 29, 2020. [En línea]. Disponible en: https://eprint.iacr.org/2016/801.
[36] A. Phaneuf, «Latest trends in medical monitoring devices and wearable health technology», *Business Insider*. https://www.businessinsider.com/wearable-technology-healthcare-medical-devices (accedido abr. 29, 2020).
[37] «Boxcryptor | Seguridad para su Nube». https://www.boxcryptor.com/ (accedido abr. 29, 2020).
[38] «SpiderOak Secure Software», *SpiderOak*. https://spideroak.com/ (accedido abr. 29, 2020).
[39] Y.-C. Chen, G. Horng, Y.-J. Lin, y K.-C. Chen, «Privacy Preserving Index for Encrypted Electronic Medical Records», *J. Med. Syst.*, vol. 37, n.º 6, p. 9992, oct. 2013, doi: 10.1007/s10916-013-9992-x.
[40] S. Carpov, T. H. Nguyen, R. Sirdey, G. Constantino, y F. Martinelli, «Practical Privacy-Preserving Medical Diagnosis Using Homomorphic Encryption», en *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, jun. 2016, pp. 593-599, doi: 10.1109/CLOUD.2016.0084.
[41] «random oracle model - How can homomorphic encryption be probabilistic while allowing for math to be conducted?», *Cryptography Stack Exchange*. https://crypto.stackexchange.com/questions/61022/how-can-homomorphic-encryption-be-probabilistic-while-allowing-for-math-to-be-co (accedido abr. 29, 2020).
[42] D. Archer *et al.*, «Applications of Homomorphic Encryption», HomomorphicEncryption.org, Redmond WA, USA, jul. 2017.
[43] F. Li, B. Luo, y P. Liu, «Secure Information Aggregation for Smart Grids Using Homomorphic Encryption», en *2010 First IEEE International Conference on Smart Grid Communications*, Gaithersburg, MD, USA, oct. 2010, pp. 327-332, doi: 10.1109/SMARTGRID.2010.5622064.
[44] «Descubren un fallo en contadores de luz de toda España que puede dejarte a oscuras», *El Confidencial*, mar. 06, 2020. https://www.elconfidencial.com/tecnologia/2020-03-06/hacking-tarlogic-ciberseguridad-startup-infraestructuras-criticas_2478599/ (accedido abr. 29, 2020).
[45] A. Abdulatif, I. Khalil, H. Kumarage, y M. Atiquzzaman, «Privacy-preserving cloud-based billing with lightweight homomorphic encryption for sensor-enabled smart grid infrastructure», *IET Wirel. Sens. Syst.*, vol. 7, jul. 2017, doi: 10.1049/iet-wss.2017.0061.
[46] P. Hu, T. Mukherjee, A. Valliappan, y S. Radziszowski, «Evaluation of homomorphic primitives for computations on encrypted data for CPS systems», en *2016 Smart City Security and Privacy Workshop (SCSP-W)*, abr. 2016, pp. 1-5, doi: 10.1109/SCSPW.2016.7509556.
[47] «NTL: A Library for doing Number Theory». https://www.shoup.net/ntl/ (accedido abr. 29, 2020).
[48] «IDASH PRIVACY & SECURITY WORKSHOP 2019 - secure genome analysis competition - Home». http://www.humangenomeprivacy.org/2019/ (accedido abr. 23, 2020).
[49] H. Chen *et al.*, «Logistic regression over encrypted data from fully homomorphic encryption», *BMC Med. Genomics*, vol. 11, n.º 4, p. 81, oct. 2018, doi: 10.1186/s12920-018-0397-z.
[50] I. Chillotti, «iDASH 2019 – KU Leuven among the winners | COSIC». https://www.esat.kuleuven.be/cosic/blog/idash-2019-ku-leuven-among-the-winners/ (accedido abr. 29, 2020).

[51] T. Graepel, K. Lauter, y M. Naehrig, «ML Confidential: Machine Learning on Encrypted Data», en *Information Security and Cryptology – ICISC 2012*, Berlin, Heidelberg, 2013, pp. 1-21, doi: 10.1007/978-3-642-37682-5_1.
[52] D. A. Steffen, «The Paillier Cryptosystem», p. 20.
[53] T. Adachi, «Composite residuosity and its application to cryptography», p. 6.
[54] «Trapdoor function», *Wikipedia*. ene. 30, 2020, Accedido: abr. 23, 2020. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Trapdoor_function&oldid=938358105.
[55] «Computational hardness assumption», *Wikipedia*. ene. 29, 2020, Accedido: abr. 23, 2020. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Computational_hardness_assumption&oldid=938082463.
[56] «Public Key Encryption with Learning With Errors (LWE)». https://asecuritysite.com/encryption/lwe2 (accedido abr. 23, 2020).
[57] P. B. B. OBE, «Public Key Encryption with Learning With Errors (LWE)», *Medium*, abr. 07, 2019. https://medium.com/asecuritysite-when-bob-met-alice/public-key-encryption-with-learning-with-errors-lwe-1aa6dd1df4e (accedido ago. 19, 2019).
[58] D. Boneh, E.-J. Goh, y K. Nissim, «Evaluating 2-DNF Formulas on Ciphertexts», en *Theory of Cryptography*, vol. 3378, J. Kilian, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 325-341.
[59] Z. Brakerski, C. Gentry, y V. Vaikuntanathan, «Fully Homomorphic Encryption without Bootstrapping», 277, 2011. Accedido: abr. 07, 2020. [En línea]. Disponible en: https://eprint.iacr.org/2011/277.
[60] J. Fan y F. Vercauteren, «Somewhat Practical Fully Homomorphic Encryption», *IACR Cryptol. EPrint Arch.*, vol. 2012, p. 144, 2012.
[61] J. H. Cheon, A. Kim, M. Kim, y Y. Song, «Homomorphic Encryption for Arithmetic of Approximate Numbers», en *Advances in Cryptology – ASIACRYPT 2017*, 2017, pp. 409-437.
[62] I. Chillotti, N. Gama, M. Georgieva, y M. Izabachène, «Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds», en *Advances in Cryptology – ASIACRYPT 2016*, vol. 10031, J. H. Cheon y T. Takagi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 3-33.
[63] L. Ducas y D. Micciancio, «FHEW: Bootstrapping Homomorphic Encryption in less than a second», 816, 2014. Accedido: ago. 27, 2019. [En línea]. Disponible en: https://eprint.iacr.org/2014/816.
[64] «Procesador Intel® Core™ i7-960 (caché de 8M, 3,20 GHz, 4,80 GT/s Intel® QPI) Especificaciones de productos». https://ark.intel.com/content/www/es/es/ark/products/37151/intel-core-i7-960-processor-8m-cache-3-20-ghz-4-80-gt-s-intel-qpi.html (accedido abr. 29, 2020).
[65] C. Boura, N. Gama, M. Georgieva, y D. Jetchev, «CHIMERA: Combining Ring-LWE-based Fully Homomorphic Encryption Schemes», 758, 2018. Accedido: ago. 27, 2019. [En línea]. Disponible en: https://eprint.iacr.org/2018/758.
[66] M. R. Albrecht, R. Fitzpatrick, y F. Göpfert, «On the Efficacy of Solving LWE by Reduction to Unique-SVP», en *Information Security and Cryptology -- ICISC 2013*, vol. 8565, H.-S. Lee y D.-G. Han, Eds. Cham: Springer International Publishing, 2014, pp. 293-310.
[67] O. Goldreich, «On Expected Probabilistic Polynomial-Time Adversaries: A Suggestion for Restricted Definitions and Their Benefits», en *Theory of Cryptography*, Berlin,

Heidelberg, 2007, pp. 174-193, doi: 10.1007/978-3-540-70936-7_10.
[68] Z. Peng, «Danger of using fully homomorphic encryption: A look at Microsoft SEAL», *ArXiv190607127 Cs*, jun. 2019, Accedido: jul. 18, 2019. [En línea]. Disponible en: http://arxiv.org/abs/1906.07127.
[69] S. Rass y D. Slamanig, *Cryptography for Security and Privacy in Cloud Computing*. Artech House Publishers, 2013.
[70] S. Rass, «Blind Turing-Machines: Arbitrary Private Computations from Group Homomorphic Encryption», *ArXiv13123146 Cs*, dic. 2013, Accedido: mar. 20, 2020. [En línea]. Disponible en: http://arxiv.org/abs/1312.3146.
[71] A. Chatterjee y K. M. M. Aung, «Translating Algorithms to Handle Fully Homomorphic Encrypted Data», en *Fully Homomorphic Encryption in Real World Applications*, A. Chatterjee y K. M. M. Aung, Eds. Singapore: Springer, 2019, pp. 49-70.
[72] «Red de ordenamiento», *Wikipedia, la enciclopedia libre*. mar. 19, 2020, Accedido: mar. 23, 2020. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Red_de_ordenamiento&oldid=124382636.
[73] A. Chatterjee y K. M. M. Aung, *Fully Homomorphic Encryption in Real World Applications*. Springer Singapore, 2019.
[74] A. Chatterjee y I. SenGupta, «Sorting of Fully Homomorphic Encrypted Cloud Data: Can Partitioning be effective?», *IEEE Trans. Serv. Comput.*, pp. 1-1, 2017, doi: 10.1109/TSC.2017.2711018.
[75] S. Han, S. Liu, K. Chen, y D. Gu, «Proofs of Data Possession and Retrievability Based on MRD Codes», p. 23.
[76] J. Herranz, D. Hofheinz, y E. Kiltz, «KEM/DEM: Necessary and Sufficient Conditions for Secure Hybrid Encryption», p. 31.
[77] «Hardware Security Module (HSM) vs. Key Management Service (KMS)», *Equinix*, jun. 19, 2018. https://blog.equinix.com/blog/2018/06/19/hardware-security-module-hsm-vs-key-management-service-kms/ (accedido abr. 30, 2020).
[78] C. Gentry, S. Halevi, y N. P. Smart, «Homomorphic Evaluation of the AES Circuit», en *Advances in Cryptology – CRYPTO 2012*, Berlin, Heidelberg, 2012, pp. 850-867, doi: 10.1007/978-3-642-32009-5_49.
[79] A. Chatterjee y K. M. M. Aung, «FURISC: FHE Encrypted URISC Design», en *Fully Homomorphic Encryption in Real World Applications*, A. Chatterjee y K. M. M. Aung, Eds. Singapore: Springer, 2019, pp. 87-115.
[80] M. Brenner, H. Perl, y M. Smith, «How Practical is Homomorphically Encrypted Program Execution? An Implementation and Performance Evaluation», en *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, jun. 2012, pp. 375-382, doi: 10.1109/TrustCom.2012.174.
[81] J. Junquera Sánchez, «Estudio de viabilidad del uso de librerías de criptografía homomórfica en procesos reales», Universidad Europea de Madrid, Madrid, 2019.
[82] «Performing Addition on IBMs Quantum Computers», *Quantum Computing UK*. https://quantumcomputinguk.org/tutorials/performing-addition-on-ibms-quantum-computers (accedido mar. 23, 2020).

# Anexos

## Anexo I - Filtro de Bloom

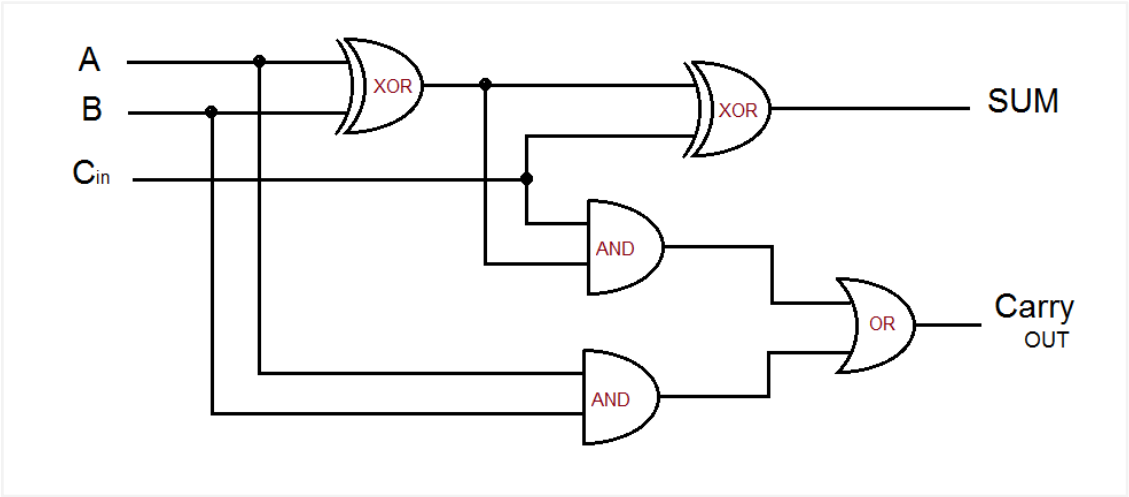



*Funcionamiento de filtro de Bloom*

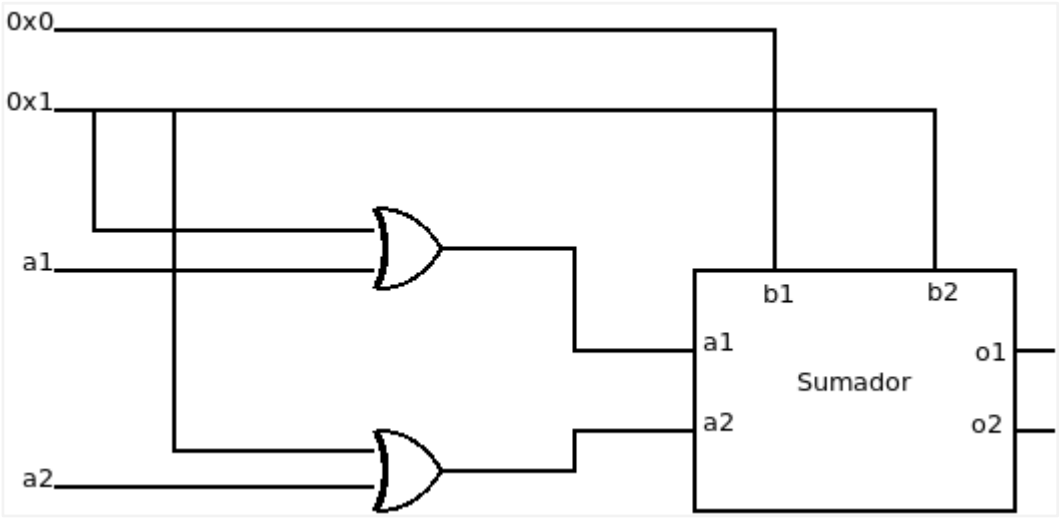
## Anexo II - Aritmética/Lógica

| A | B | A+B | A*B | A xor B | A and B |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 (*y acarreo*) | 1 | 0 | 1 |

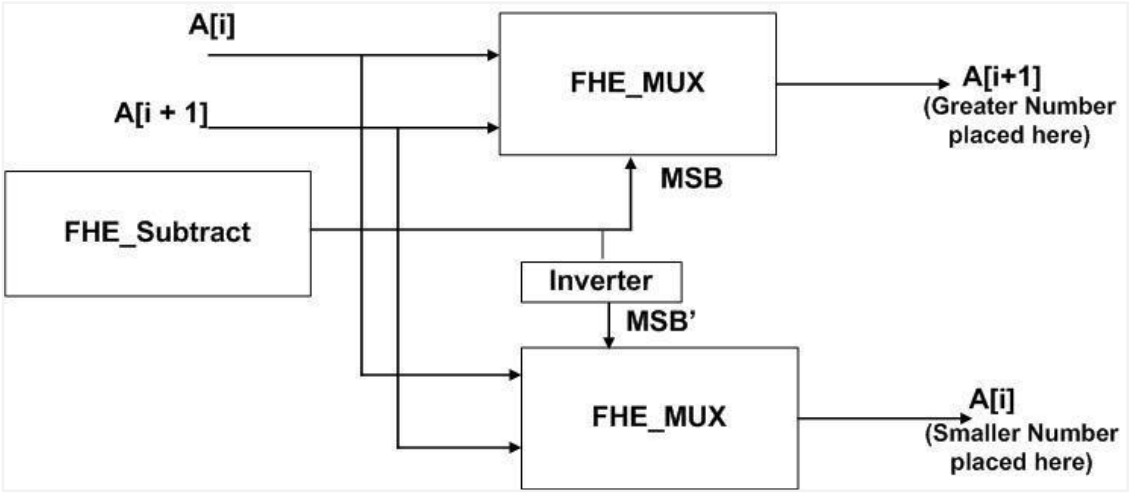
## Anexo III - Circuitos lógicos




*Circuito sumador* [82]




*Circuito restador*




*Circuito permutador [73]*

## Anexo IV - Principales librerías de criptografía homomórfica

| Librería | Esquemas | Lenguaje | Enlace |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **HELib** | BGV | C++ | https://github.com/shaih/HElib |
| **PALISADE** | BGV, BFV, CKKS, FHEW | C++ | https://palisade-crypto.org/ |
| **Microsoft SEAL** | BFV, CKKS | C++ / C# | https://www.microsoft.com/en-us/research/project/microsoft-seal/ |
| **TFHE** | TFHE | C++ | https://tfhe.github.io/tfhe/ |
| **cuHE** | DHS (derivado de LTV) | | https://github.com/vernamlab/cuHE |
| **Lattigo** | BFV, CKKS | GO | https://github.com/ldsec/lattigo |
| **PySyft** | - | Python | https://github.com/OpenMined/PySyft |