
Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Matemáticas

# Fundamentos Matemáticos del Cifrado Completamente Homomórfico

Autor: Mario Mozo Ruiz
Tutor: José Ignacio Farrán Martín
Año 2025

# Resumen

Este trabajo trata la construcción matemática del cifrado completamente homomórfico (FHE), desde los primeros esquemas hasta los más recientes. Se analiza, desde un punto de vista matemático, la construcción, desarrollo, funcionamiento, problemas y limitaciones de cada uno de ellos. Finalmente, se presenta un breve repaso del estado del arte en la aplicación práctica e implementaciones actuales y futuras.

**Palabras clave:** Cifrado completamente homomórfico, Criptografía algebraica, Bootstrapping, Ruido criptográfico, Esquemas criptográficos, Seguridad matemática.

# Abstract

This work covers the mathematical construction of fully homomorphic encryption (FHE), from the earliest schemes to the most recent developments. We analyze, from a mathematical perspective, the construction, development, operation, challenges, and limitations of each scheme. Finally, we provide a brief overview of the current state of the art regarding practical applications and current and future implementations.

**Keywords:** Fully homomorphic encryption, Algebraic cryptography, Bootstrapping, Cryptographic noise, Cryptographic schemes, Mathematical security.

# Agradecimientos

*Este trabajo es el culmen de cinco largos años de esfuerzo y dedicación, pero también de experiencias, personas, etapas, vivencias y aprendizajes. Y no es la línea de meta, es una cumbre más en esta etapa de la carrera y precede a muchas otras subidas que vendrán, pero desde la que se ve todo el paisaje con mucha belleza. Merece la pena detenerse a disfrutar de las vistas. En este momento tan importante en mi vida pienso en todas aquellas personas que han estado a mi lado y que me han ayudado, impulsado, motivado a ser mejor y llegar más allá.*

*En primer lugar, agradezco enormemente la labor docente de mi tutor, José Ignacio Farrán, y de todos mis profesores a lo largo de estos años, que con su labor hacen posible una universidad pública y accesible para todos.*

*Gracias a mi madre, Tere, por educarme, entenderme y escucharme mejor que nadie.*
*A mi padre, Gabi, por motivarme, hacerme ser una persona íntegra y darme el mejor ejemplo a seguir que he podido tener.*
*A mis hermanos, Emma y Miguel, por acompañarme, quererme y ser un salvavidas y un soplo de aire fresco en todo momento.*
*A mis abuelos, por transmitirme sus conocimientos, su forma de vida y sus enseñanzas de un valor incalculable.*
*Y a todo el resto de mi familia y amigos, en especial a Lara, que siempre ha creído en mí, pero sin olvidarme de Isma y Pedro, Álvaro, Martín, Víctor, Marco, Marta, Dani y Mario, la gente de Traspinedo, de Parquesol, de Besana, de Segovia, de Perugia...*

*Gracias a todos. Cada uno de vosotros sois un trocito de mí.*

# Índice general

# Capítulo 1

# Introducción

El desarrollo de sistemas criptográficos seguros es, desde hace décadas, uno de los grandes retos matemáticos y tecnológicos de nuestra época. La necesidad de preservar la confidencialidad, integridad y privacidad de la información se ha vuelto aún más crucial con la proliferación de servicios en la nube, el crecimiento del procesamiento distribuido y la interconexión global de datos. Si se diera la posibilidad de poder trabajar con datos cifrados en la nube, sin necesidad de descifrarlos para operar con ellos, esto supondría una revolución en la criptografía y el intercambio de información tal y como lo conocemos. Frente a esta situación, surge un desafío fascinante desde el punto de vista matemático: diseñar esquemas de cifrado que permitan operar sobre datos cifrados sin renunciar a la seguridad, es decir, realizar cálculos significativos directamente en el dominio cifrado y sin necesidad de descifrar los datos en sistemas potencialmente vulnerables.

En este contexto se sitúa el cifrado homomórfico, cuyo objetivo es permitir la evaluación de funciones sobre textos cifrados, de modo que el resultado, tras el descifrado, coincida con el que se obtendría al aplicar la función sobre los datos originales. Aunque ciertos sistemas clásicos, como RSA, ElGamal o Goldwasser-Micali, ya presentan alguna propiedad homomórfica —permitiendo operar con suma o producto, pero nunca ambas operaciones a la vez de manera ilimitada—, la verdadera meta ha sido siempre alcanzar **esquemas completamente homomórficos (FHE)**, capaces de soportar operaciones aritméticas generales y repetidas sin restricciones prácticas.

El avance fundamental en este campo se produjo con la construcción de los primeros esquemas FHE, inaugurando así una nueva era en la criptografía algebraica. La aportación de Gentry, basada en la noción de *bootstrapping*, demostró que es posible refrescar los textos cifrados y mantener la corrección aun tras muchas operaciones, lo que abrió el camino a una serie de desarrollos posteriores tanto sobre los enteros como sobre estructuras algebraicas más ricas, como los anillos de polinomios y los módulos de alta dimensión. El interés matemático de estos esquemas se enfoca en su estrecha relación con problemas fundamentales de la teoría de números, la teoría de retículos y el álgebra computacional, que no solo proporcionan la base para la funcionalidad homomórfica, sino también las garantías de seguridad frente a posibles ataques, incluso en presencia de adversarios cuánticos.

El objetivo de este trabajo es presentar, desde una perspectiva matemática, los fundamentos y la evolución de los principales esquemas de cifrado homomórfico, haciendo especial hincapié en las ideas algebraicas, la formalización de los problemas difíciles subyacentes y el análisis de las propiedades y limitaciones de cada construcción. A lo largo del texto, abordamos en primer lugar los conceptos preliminares necesarios de álgebra, teoría de números y teoría de retículos, sentando así el marco formal que permite comprender la estructura y el funcionamiento de los esquemas estudiados.

Posteriormente, explicamos en detalle la idea del cifrado homomórfico y los diferentes tipos de esquemas previos. Seguidamente, repasamos los principales cifrados homomórficos clásicos y sus limitaciones, para después adentrarnos en la construcción matemática del FHE y los retos que implica la acumulación de ruido, la profundidad de los circuitos y la necesidad de técnicas como el *bootstrapping*. El texto dedica capítulos específicos tanto a los esquemas basados en los enteros (como DGHV) como a los construidos sobre problemas de retículos (LWE y RLWE), analizando tanto su estructura matemática como las implicaciones criptográficas y las diferencias fundamentales entre ambos enfoques.

El recorrido termina con un análisis de los esquemas más recientes y relevantes, basados en la teoría de retículos, y una discusión de las ventajas, retos y aplicaciones prácticas que abren los desarrollos actuales. Cerramos el trabajo con una reflexión sobre las tendencias más prometedoras del campo y las preguntas abiertas, subrayando el papel central que ocupa la matemática avanzada en la evolución de la criptografía moderna.

En definitiva, este trabajo busca ofrecer una visión ordenada, rigurosa y accesible de los cimientos matemáticos del cifrado completamente homomórfico, destacando su relevancia tanto teórica como práctica y mostrando cómo el diálogo entre la teoría algebraica y las necesidades criptográficas ha dado lugar a uno de los avances más notables de la criptografía moderna.

# Capítulo 2

# Preliminares Matemáticos

## 2.1. Álgebra básica: anillos, cuerpos y módulos

El estudio de la criptografía homomórfica requiere el manejo de ciertas estructuras algebraicas fundamentales. En este apartado, se repasarán las nociones más básicas, como los conceptos de **anillo**, **cuerpo** y **módulo**, con el objetivo de sentar las bases sobre las que se desarrollan los principales esquemas de cifrado homomórfico.
Los conceptos y notación algebraica de este apartado, a pesar de ser muy básicos y sencillos, se han decidido incluir para fijar contexto y facilitar la lectura a quienes no estén familiarizados con la terminología habitual en teoría algebraica y su uso en criptografía moderna.

### 2.1.1. Grupos

**Definición 2.1** (Grupo)**.** Un **grupo** es un conjunto $G$ junto con una operación binaria $*$ tal que:

1. Asociatividad: Para todo $a, b, c \in G$, se cumple $(a * b) * c = a * (b * c)$.
2. Elemento neutro: Existe un elemento $e \in G$ tal que $e * a = a * e = a$ para todo $a \in G$.
3. Inverso: Para cada $a \in G$, existe $b \in G$ tal que $a * b = b * a = e$.

Si, además, $a * b = b * a$ para todo $a, b \in G$, el grupo se dice *abeliano*.

**Definición 2.2** (Grupo cíclico)**.** Un **grupo cíclico** es un grupo $G$ tal que existe un elemento $g \in G$ con la propiedad de que, para todo $x \in G$, existe un entero $k$ tal que $x = g^k$ (si la operación es multiplicativa) o $x = k \cdot g$ (si es aditiva). A dicho elemento $g$ se le llama *generador* del grupo.

**Definición 2.3** (Logaritmo discreto)**.** Sea $G$ un grupo cíclico finito, escrito multiplicativamente, con generador $g \in G$. Para cualquier $h \in G$, el **logaritmo discreto** de $h$ en base $g$ es el entero $k$ tal que $g^k = h$ (si existe).

### 2.1.2. Anillos

**Definición 2.4** (Anillo)**.** Un **anillo** es una estructura algebraica formada por un conjunto $R$ provisto de dos operaciones binarias, generalmente llamadas suma $(+)$ y producto $(\cdot)$, que cumplen una serie de propiedades:

1. $(R, +)$ es un grupo abeliano.
2. El producto $\cdot$ es asociativo: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ para todo $a, b, c \in R$.
3. La distributividad del producto $\cdot$ respecto de la suma $+$ se cumple:

$$a \cdot (b + c) = a \cdot b + a \cdot c \quad \text{y} \quad (a + b) \cdot c = a \cdot c + b \cdot c$$

   para todo $a, b, c \in R$.

Si existe un elemento $1 \in R$ tal que $1 \cdot a = a \cdot 1 = a$ para todo $a \in R$, el anillo se dice *con unidad*.

**Ejemplo 2.1. Anillos relevantes en criptografía:**

- El anillo de restos módulo $n$, $(\mathbb{Z}/n\mathbb{Z}, +, \cdot)$, de especial importancia en esquemas como Paillier, RSA o DGHV.
- El anillo de polinomios $\mathbb{F}_p[x]$, imprescindible para la criptografía basada en retículos y esquemas avanzados de FHE.

En criptografía, la mayoría de anillos usados son conmutativos (tanto para la suma como para el producto) con unidad. Más específicamente, el contexto del cifrado homomórfico, los anillos proporcionan el marco en el que se definen las operaciones cifradas. La posibilidad de realizar sumas y productos en el espacio cifrado depende de la estructura del anillo subyacente.

### 2.1.3. Cuerpos

**Definición 2.5** (Cuerpo)**.** Un **cuerpo** es un anillo conmutativo con unidad en el que todo elemento distinto de cero tiene inverso multiplicativo.

Formalmente, un cuerpo es una estructura $(K, +, \cdot)$ tal que:

- $(K, +)$ es un grupo abeliano, con elemento neutro $0_K$ e inverso $-a$.
- $(K \setminus \{0_K\}, \cdot)$ es un grupo abeliano, con elemento neutro $1_K$ e inverso $a^{-1}$.
- El producto es distributivo respecto de la suma.

**Ejemplo 2.2.** El cuerpo finito $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$, donde $p$ es primo.

Los cuerpos son esenciales cuando se trabaja con operaciones modulares. Por ejemplo, en el caso de Goldwasser–Micali o en algunos pasos de la construcción de LWE, los cuerpos finitos se utilizan para garantizar la existencia de inversos.

### 2.1.4. Módulos

**Definición 2.6** (Módulo)**.** El concepto de **módulo** generaliza el de espacio vectorial, permitiendo que los escalares provengan de un anillo en vez de un cuerpo. Formalmente, dado un anillo $R$, un *módulo* sobre $R$ es un conjunto $M$ con una operación de suma, de forma que $(M, +)$ es un grupo abeliano, y una acción de $R$ sobre $M$ que satisface las propiedades habituales de compatibilidad y distributividad.

Para todo $m, n \in M$ y $r, s \in R$:

- $r \cdot (m + n) = r \cdot m + r \cdot n$
- $(r + s) \cdot m = r \cdot m + s \cdot m$
- $(r \cdot s) \cdot m = r \cdot (s \cdot m)$
- Si $R$ tiene unidad, $1 \cdot m = m$

**Ejemplo 2.3.** $\mathbb{Z}^n$ como módulo sobre $\mathbb{Z}$ es esencial a la hora de tratar con retículos (por ejemplo, en esquemas FHE basados en LWE).

El uso de módulos es especialmente relevante en la criptografía basada en retículos, donde los retículos se interpretan como módulos discretos sobre $\mathbb{Z}$ o sobre otros anillos.

## 2.2. Elementos fundamentales de teoría de números

La teoría de números proporciona la base aritmética sobre la que se construyen muchos esquemas criptográficos, incluidos los de cifrado homomórfico. En esta sección se repasan la notación y los conceptos básicos que se utilizarán a lo largo del trabajo.

### 2.2.1. Aritmética modular

**Definición 2.7.** Dados dos enteros $a, n$ con $n > 1$, se dice que $a$ **es congruente con** $b$ **módulo** $n$ si $n$ divide a $a - b$, y se denota $a \equiv b \pmod{n}$.

La clase de equivalencia de $a$ módulo $n$ se representa por $\overline{a}$ o $[a]_n$. El conjunto de las clases de equivalencia módulo $n$ se denota $\mathbb{Z}/n\mathbb{Z}$, que es un anillo con las operaciones suma y producto inducidas.

La aritmética modular es una herramienta fundamental en criptografía, como veremos en los próximos capítulos.

### 2.2.2. Primos y propiedades fundamentales

**Definición 2.8.** Un número **primo** es un entero $p > 1$ cuyos únicos divisores positivos son 1 y $p$. Los primos son esenciales en muchos algoritmos criptográficos por sus propiedades aritméticas, como la imposibilidad de factorizar productos de grandes primos de forma eficiente.

Dos enteros $a$ y $b$ se dicen **coprimos** (o primos entre sí) si su máximo común divisor es 1, es decir, $\gcd(a, b) = 1$.

Esta propiedad es fundamental para la existencia de **inversos modulares**: dado $a \in \mathbb{Z}$ y $n > 1$, si $\gcd(a, n) = 1$ (es decir, son coprimos), entonces existe un inverso multiplicativo de $a$ módulo $n$, es decir, existe $x \in \mathbb{Z}$ tal que $ax \equiv 1 \pmod{n}$.

El cálculo efectivo de este inverso se realiza mediante el **algoritmo de Euclides extendido**, una herramienta de criptografía que permite encontrar de manera eficiente el inverso modular siempre que $a$ y $n$ sean coprimos, lo que resulta crucial en la generación de claves y el descifrado de muchos esquemas criptográficos.

### 2.2.3. Residuos cuadráticos, símbolo de Legendre y símbolo de Jacobi

**Definición 2.9** (Residuo cuadrático)**.** Sea $n$ un entero compuesto y $x \in \mathbb{Z}_n^*$. Se dice que $x$ es un **residuo cuadrático** módulo $n$ si existe $y \in \mathbb{Z}_n^*$ tal que $x \equiv y^2 \pmod{n}$. En caso contrario, $x$ se denomina **no residuo cuadrático** módulo $n$.

Si $n$ es primo, exactamente la mitad de los elementos de $\mathbb{Z}_n^*$ son residuos cuadráticos. Para $n$ compuesto, la caracterización es más compleja y, en particular, es computacionalmente difícil determinar si $x$ es un residuo cuadrático cuando la factorización de $n$ es desconocida.

**Definición 2.10** (Símbolo de Legendre)**.** Sea $p$ un número primo impar y $a \in \mathbb{Z}$. El **símbolo de Legendre** se define como:

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{si } p \mid a \\ 1 & \text{si } a \not\equiv 0 \pmod{p} \text{ y existe } x \in \mathbb{Z} \text{ tal que } x^2 \equiv a \pmod{p} \\ -1 & \text{si no existe tal } x \end{cases}$$

Es decir, $\left(\frac{a}{p}\right) = 1$ si $a$ es residuo cuadrático módulo $p$, $-1$ si no lo es, y 0 si $p$ divide a $a$.

**Definición 2.11** (Símbolo de Jacobi)**.** Sean $n \geq 3$ un entero impar y $a \in \mathbb{Z}$. El **símbolo de Jacobi** $\left(\frac{a}{n}\right)$ se define como el producto de los símbolos de Legendre para cada uno de los factores primos $p_i$ de $n$:

$$\left(\frac{a}{n}\right) = \prod_{i=1}^{k} \left(\frac{a}{p_i}\right)$$

donde $n = p_1^{e_1} \cdots p_k^{e_k}$ es la descomposición en primos de $n$, y $\left(\frac{a}{p_i}\right)$ es el símbolo de Legendre.

**Ejemplo 2.4.** Sea $n = 15 = 3 \cdot 5$ y $a = 2$. Entonces:

$$\left(\frac{2}{15}\right) = \left(\frac{2}{3}\right) \cdot \left(\frac{2}{5}\right) = (-1) \cdot (-1) = 1$$

Obsérvese que aunque el símbolo de Jacobi es 1, 2 no es residuo cuadrático módulo 15.

El símbolo de Jacobi se puede calcular de manera eficiente sin conocer la factorización de $n$, pero no caracteriza completamente los residuos cuadráticos cuando $n$ es compuesto: si $\left(\frac{a}{n}\right) = -1$, entonces $a$ es un no residuo cuadrático módulo $n$; si $\left(\frac{a}{n}\right) = 1$, $a$ puede ser residuo o no residuo cuadrático módulo $n$.

Este hecho se utiliza en criptografía para construir esquemas donde la dificultad de distinguir residuos de no residuos asegura la seguridad, como en el esquema de Goldwasser–Micali.

### 2.2.4. Teoremas clásicos

Algunos resultados de la teoría de números que se emplean con frecuencia en criptografía son los siguientes:

**Teorema 2.1. Pequeño teorema de Fermat**: Si $p$ es primo y $a$ es un entero no divisible por $p$, entonces

$$a^{p-1} \equiv 1 \pmod{p}$$

**Teorema 2.2. Teorema de Euler**: Si $n$ y $a$ son enteros coprimos,

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

donde $\varphi(n)$ es la función phi de Euler, que cuenta los enteros positivos menores que $n$ y coprimos con $n$.

### Ejemplos relevantes

- En RSA, la seguridad se basa en la dificultad de factorizar $n = pq$ siendo $p$ y $q$ primos grandes.
- En esquemas como DGHV, el uso de la aritmética modular y la gestión de restos es esencial.
- La función phi de Euler aparece directamente en la generación de claves y el cálculo de inversos.

## 2.3. Retículos y problemas clásicos

Los retículos juegan un papel fundamental en criptografía moderna y en el estudio de problemas algorítmicos difíciles. A continuación presentamos las definiciones y resultados matemáticos que serán necesarios para el desarrollo de esquemas criptográficos basados en retículos.

### 2.3.1. Definición y elementos de un retículo

**Definición 2.12.** Un *retículo* $\mathcal{L}$ de dimensión $n$ en $\mathbb{R}^m$ es el conjunto

$$\mathcal{L} = \left\{ \sum_{i=1}^{n} \lambda_i \mathbf{b}_i : \lambda_i \in \mathbb{Z} \right\}$$

donde los vectores $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$ son linealmente independientes. El conjunto $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ se denomina *base* del retículo $\mathcal{L}$ y $n$ es la *dimensión* o *rango* del retículo.

**Definición 2.13.** Alternativamente, un *retículo* en $\mathbb{R}^n$ es cualquier subgrupo discreto de $(\mathbb{R}^n, +)$, es decir, un subconjunto $\mathcal{L} \subset \mathbb{R}^n$ tal que:

- $0 \in \mathcal{L}$, si $x \in \mathcal{L}$ entonces $-x \in \mathcal{L}$, y $x + y \in \mathcal{L}$ para todo $x, y \in \mathcal{L}$.
- Para cada $x \in \mathcal{L}$ existe un entorno de $x$ en $\mathbb{R}^n$ que no contiene otros puntos de $\mathcal{L}$.

### Determinante de un retículo

**Definición 2.14.** Sea $\mathcal{L}$ un retículo con base $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, el *determinante* de $\mathcal{L}$ se define como

$$\det(\mathcal{L}) = |\det(\mathbf{B})|$$

donde $\mathbf{B}$ es la matriz cuyas columnas son los vectores de la base.

### Paralelepípedo fundamental de un retículo

**Definición 2.15.** Dada una base $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ de un retículo $\mathcal{L}$, se llama *paralelepípedo fundamental* al conjunto

$$\mathcal{P} = \left\{ \sum_{i=1}^{n} \alpha_i \mathbf{b}_i : 0 \leq \alpha_i < 1 \right\}$$

**Proposición 2.3.** Sea $\mathcal{L}$ un retículo de dimensión $n$ con base $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$. El volumen del paralelepípedo fundamental $\mathcal{P}$ asociado a la base es igual a $|\det(\mathbf{B})|$, es decir,

$$\mathrm{vol}(\mathcal{P}) = |\det(\mathbf{B})| = \det(\mathcal{L}).$$

*Demostración.* La demostración es directa a partir de la teoría clásica de álgebra lineal: el volumen del paralelepípedo generado por los vectores $\mathbf{b}_1, \ldots, \mathbf{b}_n$ es precisamente el valor absoluto del determinante de la matriz formada por estos vectores como columnas. □

Con estos conceptos, podemos comprobar que las dos definiciones de retículo dadas son equivalentes:

**Proposición 2.4.** Sea $\mathcal{L} \subset \mathbb{R}^n$. Entonces las siguientes afirmaciones son equivalentes:

1. $\mathcal{L}$ es un retículo, es decir, existe un conjunto de $n$ vectores linealmente independientes $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^n$ tal que
$$\mathcal{L} = \left\{ \sum_{i=1}^{n} \lambda_i \mathbf{b}_i : \lambda_i \in \mathbb{Z} \right\}$$
2. $\mathcal{L}$ es un subgrupo aditivo discreto de $\mathbb{R}^n$ de rango finito.

*Demostración.* (1 $\implies$ 2) Sea $\mathcal{L}$ como en (1). Claramente, $\mathcal{L}$ es un subgrupo de $(\mathbb{R}^n, +)$ porque la suma y los inversos de combinaciones lineales enteras siguen siendo combinaciones lineales enteras. Además, $\mathcal{L}$ es discreto: dado que los vectores de la base son linealmente independientes, el paralelepípedo fundamental asociado tiene volumen positivo, y existe un entorno alrededor de cada punto en el que no hay más puntos del retículo.

(2 $\implies$ 1) Sea $\mathcal{L}$ un subgrupo aditivo discreto de $\mathbb{R}^n$ de rango finito. Entonces, por resultados clásicos de álgebra lineal, existe un conjunto finito de generadores $\mathbf{b}_1, \ldots, \mathbf{b}_n$ linealmente independientes tal que cada elemento de $\mathcal{L}$ es combinación lineal entera de estos vectores, es decir,

$$\mathcal{L} = \left\{ \sum_{i=1}^{n} \lambda_i \mathbf{b}_i : \lambda_i \in \mathbb{Z} \right\}.$$

□

**Ejemplo 2.5.** El conjunto $\mathbb{Z}^n = \{(x_1, \ldots, x_n) : x_i \in \mathbb{Z}\}$ es un retículo de dimensión $n$ en $\mathbb{R}^n$ cuya base es la canónica de los vectores estándar.

### 2.3.2. Otros conceptos

Aunque en este trabajo no se desarrollan en detalle, conviene mencionar dos resultados importantes de la teoría de retículos:
Por un lado, la **forma normal de Hermite** [1] proporciona una manera ordenada y única de expresar una base del retículo con vectores lo más cortos y ortogonales posible, lo que facilita mucho algunos cálculos y comparaciones entre retículos.
Por otro lado, el **lema de Minkowski** [2] es un teorema clásico que garantiza que, bajo ciertas condiciones de simetría y volumen, cualquier retículo contiene puntos suficientemente "cortos". Esta es una propiedad clave en la seguridad y en el diseño de esquemas criptográficos basados en retículos.

### 2.3.3. Problemas fundamentales

- Shortest Vector Problem (SVP): Dado un retículo $\mathcal{L}$, consiste en encontrar un vector no nulo $\mathbf{v} \in \mathcal{L} \setminus \{0\}$ tal que:

$$\|\mathbf{v}\| = \lambda_1(\mathcal{L}).$$

- Closest Vector Problem (CVP): Dado un retículo $\mathcal{L}$ y un vector $\mathbf{t} \in \mathbb{R}^n$, consiste en encontrar $\mathbf{v} \in \mathcal{L}$ que minimice la distancia $\|\mathbf{t} - \mathbf{v}\|$.

- Short Integer Solution (SIS): Dada una matriz $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, consiste en encontrar un vector $\mathbf{x} \in \mathbb{Z}^m \setminus \{0\}$ tal que, para $\|\mathbf{x}\|$ pequeño,

$$\mathbf{A}\mathbf{x} = 0 \pmod{q}$$

- Learning With Errors (LWE): Consiste en distinguir entre muestras $(\mathbf{a}, b)$ con $b = \langle \mathbf{a}, \mathbf{s} \rangle + e$ (donde $e$ es un error pequeño, $\mathbf{a} \in \mathbb{Z}_q^n$ es aleatorio y $\mathbf{s} \in \mathbb{Z}_q^n$ secreto) y muestras completamente aleatorias.

## 2.4. Lógica matemática y funciones

En el estudio de la criptografía homomórfica, las funciones juegan un papel fundamental al formalizar los algoritmos de cifrado, descifrado y evaluación a partir de funciones entre conjuntos.

### 2.4.1. Funciones y composición

**Definición 2.16.** Una **función** entre dos conjuntos $A$ y $B$ es una regla que asigna a cada elemento $a \in A$ un único elemento $b \in B$, y se denota $f(a) = b$. En este trabajo, la mayoría de las funciones consideradas son aplicaciones entre conjuntos finitos de números enteros o elementos de anillos y cuerpos.

La **composición de funciones** funciona de la siguiente forma: dada $f : A \to B$ y $g : B \to C$, la función compuesta $g \circ f : A \to C$ se define como $g \circ f(a) = g(f(a))$. Esta herramienta se utiliza habitualmente en la descripción de esquemas criptográficos.

#### Notación y propiedades básicas

A lo largo del trabajo se emplea la notación estándar de funciones: $\mathrm{Im}(f)$ como la imagen de $f$, $\mathrm{Ker}(f)$ como el núcleo de $f$ (si $f$ es un homomorfismo), y $f^{-1}$ como la función inversa, cuando existe.

La distinción entre **función inyectiva** ($f(a_1) = f(a_2)$ implica $a_1 = a_2$), **sobreyectiva** ($\mathrm{Im}(f) = B$) y **biyectiva** (inyectiva y sobreyectiva) también será relevante.

### 2.4.2. Álgebra de Boole y funciones booleanas

El marco matemático fundamental para el estudio de la lógica y la computación sobre bits es la **álgebra de Boole**. Una álgebra de Boole es una estructura algebraica en la que se definen operaciones binarias (como la conjunción y la disyunción) y una operación unaria (la negación), que satisfacen ciertos axiomas básicos de conmutatividad, asociatividad, distributividad, neutro y complemento.

En particular, el conjunto $\{0, 1\}$, con las operaciones AND ($\wedge$), OR ($\vee$) y NOT ($\neg$), constituye una álgebra de Boole finita, que es la base para la definición de **funciones booleanas** y su manipulación algebraica. Cualquier función booleana se puede entender, por tanto, como una aplicación definida sobre una álgebra de Boole, es decir, $f : \{0, 1\}^n \to \{0, 1\}^m$.

Estas funciones pueden representarse mediante circuitos lógicos (combinaciones de puertas lógicas) o como polinomios sobre cuerpos finitos, especialmente sobre $\mathbb{F}_2$. Esta correspondencia permite traducir cualquier circuito lógico en una expresión polinómica, y viceversa. Algunas de estas aplicaciones son las siguientes:

- $\mathbf{NOT}(x) = 1 - x$
- $\mathbf{AND}(x, y) = x \cdot y$
- $\mathbf{OR}(x, y) = x + y - x \cdot y$
- $\mathbf{XOR}(x, y) = x + y$ mód 2
- $\mathbf{NAND}(x, y) = 1 - x \cdot y$

De este modo, cualquier función booleana puede representarse como un polinomio (en general multilineal) en las variables de entrada. Esto es la base para la evaluación homomórfica de circuitos: las operaciones algebraicas sobre los textos cifrados corresponden a las operaciones lógicas sobre los datos originales.

## Cosideraciones

Los conceptos y resultados expuestos forman la base algebraica sobre la que se construyen los esquemas de cifrado homomórfico, y serán empleados en los capítulos siguientes para describir y analizar las operaciones y la seguridad de los distintos esquemas.

# Capítulo 3

# Cifrado Homomórfico

## 3.1. Definición formal

### 3.1.1. Esquema y algoritmos de cifrado homomórfico

Un esquema de cifrado homomórfico es un sistema criptográfico que permite realizar ciertas operaciones algebraicas directamente sobre datos cifrados, de modo que el resultado, al descifrarse, coincide con el resultado de aplicar la operación correspondiente a los datos sin cifrar. Esta propiedad se denomina *homomorfismo* y es la base para la computación segura sobre datos cifrados.

Estos esquemas operan sobre dos espacios principales: el *espacio de mensajes* $M$, que contiene los datos originales (por ejemplo, bits, enteros módulo $q$, o polinomios), y el *espacio de cifrados* $C$, donde residen los datos cifrados. La relación entre ambos espacios la determinan los algoritmos del esquema, que definen cómo los elementos de $M$ se transforman y manipulan en $C$ preservando las operaciones algebraicas especificadas por la propiedad homomórfica específica.

Formalmente, un esquema de cifrado homomórfico se compone de cuatro algoritmos principales, todos ellos eficientes (es decir, de complejidad polinómica en el parámetro de seguridad):

- **KeyGen**: algoritmo de generación de claves. Dado un parámetro de seguridad $\lambda$ (y, en ciertos esquemas, un parámetro $\alpha$), genera una clave pública ($pk$), una clave secreta ($sk$) y, en algunos esquemas, una clave de evaluación ($evk$):

  $$(\mathrm{pk}, \mathrm{sk}, \mathrm{evk}) \leftarrow \mathrm{KeyGen}(1^{\lambda}, \alpha)$$

  La clave pública se puede distribuir libremente; la clave secreta permanece en posesión del propietario de los datos.

- **Enc**: algoritmo de cifrado. Toma la clave pública y un mensaje $m$ del espacio de mensajes y devuelve un cifrado $c$.

  $$\mathrm{Enc}\colon M \to C; \quad c \leftarrow \mathrm{Enc}(\mathrm{pk}, m)$$

Este cifrado se puede transmitir o almacenar en sistemas no confiables sin exponer el mensaje original.

- **Eval**: algoritmo de evaluación. Dada la clave de evaluación, una función (o circuito) $f$ y una colección de cifrados $c_1, \ldots, c_n$, produce un nuevo cifrado $c_f$ que corresponde a $f(m_1, \ldots, m_n)$, donde cada $m_i$ es el mensaje original correspondiente a $c_i$.

$$c_f \leftarrow \text{Eval}(\text{evk}, f, c_1, \ldots, c_n)$$

  Lo importante es que este cálculo se realiza sin necesidad de descifrar los datos.

- **Dec**: algoritmo de descifrado. Utiliza la clave secreta para recuperar el mensaje original (o el resultado de la operación) a partir del cifrado correspondiente.

$$\text{Dec}\colon C \to M; \quad m \leftarrow \text{Dec}(\text{sk}, c)$$

El funcionamiento típico de un sistema de cifrado homomórfico es el siguiente: el cliente o propietario de los datos genera las claves y cifra los mensajes con la clave pública. Los datos cifrados pueden ser enviados a un servidor o a un tercero, que realiza operaciones sobre ellos (por ejemplo, evaluando funciones o circuitos) mediante el algoritmo de evaluación. El resultado, aún cifrado, es devuelto al cliente, quien lo descifra con su clave secreta y obtiene el valor que habría resultado de operar directamente sobre los datos originales. De este modo, la confidencialidad de los datos se mantiene durante todo el proceso.

### 3.1.2. Propiedades de los esquemas de cifrado homomórfico

Para que un esquema de cifrado homomórfico sea práctico y seguro, debe satisfacer tres propiedades fundamentales:

- **Corrección:** El descifrado del resultado de la evaluación de cualquier función permitida sobre los cifrados debe coincidir (con alta probabilidad) con el resultado de aplicar dicha función directamente a los mensajes originales. Esta es la propiedad fundamental de estos esquemas, y formalmente podemos expresarlo tal que así:

  Para toda función $f$ permitida por el esquema y para cualquier conjunto de mensajes $m_1, \ldots, m_n$,

$$\text{Dec}\left(\text{sk}, \text{Eval}(\text{evk}, f, \text{Enc}(\text{pk}, m_1), \ldots, \text{Enc}(\text{pk}, m_n))\right) = f(m_1, \ldots, m_n)$$

  Esta propiedad garantiza que el descifrado del resultado cifrado de la operación da efectivamente el valor esperado.

- **Seguridad:** Los cifrados no deben revelar información relevante sobre los mensajes originales, ni siquiera después de múltiples evaluaciones. Formalmente, esto se traduce en la propiedad de indistinguibilidad bajo ataque de texto claro

elegido (IND-CPA): se exige que los cifrados de dos posibles mensajes elegidos por el atacante sean indistinguibles para cualquier adversario eficiente, incluso aunque este pueda elegir los mensajes a cifrar y observar los correspondientes cifrados.

- **Compacidad:** El tamaño del cifrado resultante tras cualquier evaluación y el coste del descifrado no deben crecer con el tamaño del circuito evaluado, sino que permanecen acotados por una función polinómica del parámetro de seguridad. Esta propiedad es esencial para que el cifrado sea útil en la práctica: garantiza que, aunque se evalúen operaciones complejas o circuitos largos, los cifrados no se vuelven incontrolablemente grandes ni el descifrado inviable.

  Esta propiedad, como veremos en el capítulo 5, es la que distingue realmente a los esquemas completamente homomórficos (FHE) de los simplemente algo homomórficos (SHE).

- **Otras propiedades:** Existen otras propiedades de interés, como la privacidad del circuito (*circuit privacy*) o la robustez frente a ataques activos, que se analizarán en detalle en capítulos posteriores.

En definitiva, el objetivo de los esquemas FHE es permitir la evaluación de cualquier función computable sobre datos cifrados, garantizando siempre corrección, seguridad y compacidad. Estas propiedades constituyen el fundamento sobre el que se apoyan los desarrollos y aplicaciones que se estudiarán en los siguientes capítulos.

## 3.2. Tipos de cifrado homomórfico

La clasificación de los esquemas de cifrado homomórfico es esencial para entender el desarrollo de este tipo de criptografía moderna y las posibilidades reales de computación segura sobre datos cifrados. Dependiendo del conjunto de operaciones o circuitos que permiten evaluar sobre los cifrados, se distinguen varias clases fundamentales.

### 3.2.1. Cifrado parcialmente homomórfico (PHE)

Un esquema es **parcialmente homomórfico** si admite únicamente una operación algebraica (de forma ilimitada) sobre los datos cifrados, típicamente la suma o el producto, pero no ambas. Clásicos ejemplos de este tipo son:

- El cifrado de Paillier, que es aditivo: dados dos mensajes cifrados $c_1 = \text{Enc}(m_1)$ y $c_2 = \text{Enc}(m_2)$, existe una operación sobre los cifrados tal que $\text{Dec}(c_1 \circ c_2) = m_1 + m_2$.

- El cifrado de RSA o ElGamal, que son multiplicativos: la operación sobre los cifrados corresponde a la multiplicación de los mensajes.

Estos esquemas son útiles en aplicaciones donde solo se necesita una operación (por ejemplo, agregación de votos o sumas de datos), pero no permiten la evaluación general de circuitos.

### 3.2.2. Cifrado algo homomórfico (SHE) y esquemas nivelados

Un esquema es **algo homomórfico** (Somewhat Homomorphic Encryption, SHE) si permite la evaluación de ciertos circuitos compuestos por sumas y productos, pero solo hasta una profundidad limitada, a partir de la cual el cifrado se vuelve ilegible. La principal limitación surge porque en cada operación homomórfica, los cifrados suelen acumular un "ruido" interno; una vez que este ruido supera cierto umbral, el descifrado ya no es posible.

Existen variantes dentro de los esquemas SHE:

- **Nivelado (leveled):** El esquema permite evaluar todos los circuitos de profundidad acotada $d$ (que puede elegirse al generar las claves), garantizando que el descifrado será correcto mientras no se exceda esa profundidad.
- **No nivelado:** El límite en el número de operaciones permitidas depende de los parámetros del esquema, pero no necesariamente es configurable.

En ambos casos, SHE fue el punto de partida para el desarrollo de FHE, ya que permitió entender cómo gestionar y eliminar el ruido acumulado.

Como ejemplo relevante tenemos el primer esquema de Gentry, Halevi y Smart (2010), que permitió realizar un número limitado de sumas y productos sobre datos cifrados, manteniendo la posibilidad de descifrarlos con éxito.

### 3.2.3. Cifrado completamente homomórfico (FHE)

El avance decisivo en criptografía llegó con la construcción de esquemas **completamente homomórficos** (Fully Homomorphic Encryption, FHE), que permiten la evaluación de cualquier circuito finito sobre datos cifrados, sin límite en la profundidad o el número de operaciones. Esto implica que es posible computar cualquier función computable sobre datos cifrados, siempre y cuando el esquema asegure la *compacidad*, es decir, que el tamaño de los cifrados y el coste del descifrado no crecen con la complejidad del circuito.

El paso de SHE a FHE se logró mediante la técnica de **bootstrapping**: un mecanismo por el cual el propio esquema puede "refrescar" los cifrados para reducir el ruido acumulado, permitiendo así continuar evaluando circuitos sin perder la posibilidad de descifrar el resultado final.

### 3.2.4. Comparativa formal y operativa

Formalmente, la distinción entre los tipos de esquemas se expresa en función del conjunto de circuitos $C$ que pueden evaluarse homomórficamente:

- En un PHE, $C$ contiene solo una operación (suma o producto).
- En un SHE, $C$ es el conjunto de circuitos acotados en profundidad o en número de operaciones.
- En un FHE, $C$ es el conjunto de todos los circuitos booleanos o aritméticos polinómicos en tamaño.

| | PHE | SHE | FHE |
|---|---|---|---|
| Operaciones | Suma **o** producto | Suma y producto limitados | Suma y producto ilimitados |
| Ejemplo | RSA, ElGamal, Paillier | Gentry-Halevi-Smart | Gentry, BGV, CKKS, TFHE |
| Versatilidad | Baja | Media | Alta |

El desarrollo de esquemas FHE supuso un cambio de paradigma en criptografía, permitiendo aplicaciones avanzadas como la computación delegada en la nube, bases de datos cifradas o protocolos de privacidad diferencial. Sin embargo, los esquemas SHE y PHE siguen siendo relevantes por su eficiencia y simplicidad en contextos donde no se requiere computación arbitraria.

Estos esquemas permiten adaptar el nivel de funcionalidad requerido y la eficiencia según el contexto de aplicación: desde tareas sencillas como votaciones electrónicas (donde PHE es suficiente), hasta cálculos complejos en la nube o machine learning privado, donde la potencia del FHE es esencial.

# Capítulo 4

# Primeros Esquemas y sus Limitaciones

En este capítulo se presentan cuatro esquemas clásicos de cifrado parcialmente homomórfico (PHE) con gran importancia histórica, que constituyen la base para los desarrollos más avanzados que se abordarán en los siguientes capítulos. Estos sistemas permiten ilustrar los principios matemáticos fundamentales de la **criptografía homomórfica** y sirven como punto de partida para comprender la evolución hacia el cifrado completamente homomórfico (FHE). Cada esquema permite operar con datos cifrados bajo una única operación algebraica, y todos comparten limitaciones que motivaron la búsqueda de herramientas más generales y potentes.

A lo largo del capítulo, se expondrán la construcción, la propiedad homomórfica y las principales limitaciones de cada esquema.

## 4.1. El esquema de Goldwasser-Micali: cifrado homomórfico bit a bit

El esquema de Goldwasser-Micali [3], propuesto en 1982, fue el primer sistema criptográfico público que presentó una propiedad homomórfica. Es un esquema probabilístico diseñado para el cifrado de bits individuales, en el que la operación homomórfica que se permite es la suma módulo 2, es decir, la operación XOR.

El sistema se basa en la dificultad computacional del problema del residuo cuadrático. Sean $N = pq$ el producto de dos primos grandes y $x$ un número no residuo cuadrático módulo $N$ cuya raíz cuadrada tampoco sea residuo cuadrático módulo $N$. La clave pública es $(N, x)$, mientras que la clave secreta es $(p, q)$, la factorización de $N$.

- **KeyGen**: Elegir dos primos grandes $p, q$ y calcular $N = pq$. Elegir $x \in (\mathbb{Z}/N\mathbb{Z})^*$ que no sea un residuo cuadrático módulo $N$. La clave pública es $(N, x)$, y la secreta es $(p, q)$.

- **Enc**: Para cifrar un bit $m \in \{0,1\}$, elegir un $r \in (\mathbb{Z}/N\mathbb{Z})^*$ uniforme al azar y calcular
$$c = \begin{cases} r^2 \text{ mód } N & \text{si } m = 0, \\ xr^2 \text{ mód } N & \text{si } m = 1. \end{cases}$$

- **Dec**: Usando la clave secreta, decidir si $c$ es un residuo cuadrático módulo $N$ (entonces $m = 0$) o no (entonces $m = 1$).

En términos matemáticos, la distinción entre residuo y no residuo cuadrático se expresa mediante los símbolos de Legendre y Jacobi introducidos: el símbolo de Jacobi $\left(\frac{a}{N}\right)$ generaliza el de Legendre y se emplea para caracterizar la condición de residuo cuadrático cuando $N$ es un producto de dos primos. La seguridad del esquema radica en la dificultad computacional de decidir esta propiedad cuando la factorización de $N$ es desconocida.

El esquema es aditivo (XOR) homomórfico, ya que el producto de dos cifrados $c_1, c_2$ resulta en un cifrado del bit $m_1 \oplus m_2$:

$$c_1 \cdot c_2 \equiv \text{Enc}(m_1 \oplus m_2) \pmod{N}$$

Esto permite la computación bit a bit de sumas módulo 2 sobre datos cifrados.

**Ejemplo 4.1.** Supongamos $p = 7$, $q = 11$, de modo que $N = 77$. Se toma $x = 10$, que es un no residuo cuadrático módulo $N$. Para cifrar el bit $m_1 = 0$ se elige $r_1 = 3$ y se calcula $c_1 = 3^2 \text{ mód } 77 = 9$. Para cifrar el bit $m_2 = 1$, se elige $r_2 = 5$ y se calcula $c_2 = 10 \cdot 5^2 \text{ mód } 77 = 10 \cdot 25 \text{ mód } 77 = 250 \text{ mód } 77 = 19$.

Multiplicando los cifrados se obtiene $c_1 \cdot c_2 = 9 \cdot 19 = 171 \text{ mód } 77 = 17$. Este valor corresponde al cifrado de $m_1 \oplus m_2 = 0 \oplus 1 = 1$ (ya que 17 no es un residuo cuadrático módulo 77).

Así, se verifica la propiedad homomórfica:

$$\text{Enc}(m_1) \cdot \text{Enc}(m_2) = \text{Enc}(m_1 \oplus m_2)$$

En la práctica, con parámetros $p$ y $q$ mucho mayores, se garantiza la seguridad gracias a la dificultad computacional de determinar si un número es (o no) residuo cuadrático módulo N sin conocer la factorización de N.

Sin embargo, el espacio de mensajes se restringe a bits individuales, y no permite operaciones de multiplicación aritmética entre mensajes cifrados. Además, la eficiencia del esquema y su utilidad práctica están limitadas por el tamaño de los cifrados y el crecimiento lineal al cifrar cadenas de bits.

Aun así, este esquema supuso la base para el desarrollo de criptosistemas homomórficos más generales.

## 4.2. El esquema de ElGamal: cifrado homomórfico multiplicativo

El esquema de ElGamal [4], presentado en 1985, es uno de los sistemas de cifrado más conocidos y se basa en la dificultad del problema del logaritmo discreto. Permite la operación de producto sobre los mensajes cifrados, es decir, es multiplicativamente homomórfico.

El sistema se define sobre un grupo cíclico $G$ (en la práctica se toma $G = \mathbb{Z}_p^*$, donde $p$ es un número primo y el grupo tiene orden $p-1$) con generador $g$. La seguridad se apoya en la dificultad de calcular logaritmos discretos en $G$.

- **KeyGen**: Se elige un grupo cíclico $G$ de orden $p-1$, con $p$ primo, y un generador $g$ de $G$. El usuario selecciona un exponente secreto $a \in \mathbb{Z}_p^*$ y calcula $h = g^a$. La clave pública es $(G, p, g, h)$, y la clave secreta es $a$.
- **Enc**: Para cifrar un mensaje $m \in G$, elegir un $k \in \mathbb{Z}_p^*$ al azar y calcular el par

$$c = (c_1, c_2) = (g^k,\ m \cdot h^k)$$

- **Dec**: Dado el cifrado $(c_1, c_2)$, se recupera el mensaje original como

$$m = c_2 \cdot (c_1^a)^{-1}$$

La propiedad homomórfica del esquema se manifiesta en que, dados dos cifrados $c = (c_1, c_2)$ y $c' = (c_1', c_2')$ de $m$ y $m'$, el producto componente a componente

$$(c_1 c_1',\ c_2 c_2') = (g^{k+k'},\ mm' h^{k+k'})$$

es un cifrado válido de $mm'$. Por tanto, ElGamal permite multiplicar mensajes cifrados sin descifrarlos.

**Ejemplo 4.2.** Supongamos que trabajamos en el grupo multiplicativo $\mathbb{Z}_{17}^*$, que es cíclico de orden 16, con generador $g = 3$. Tomamos el exponente secreto $a = 6$, de modo que $h = 3^6 \bmod 17 = 729 \bmod 17 = 16 \bmod 17 = -1$.

Para cifrar el mensaje $m = 5$, elegimos $k = 7$ al azar, por lo que

$$c_1 = 3^7 \bmod 17 = 2187 \bmod 17 = 14;$$
$$c_2 = 5 \cdot (-1)^7 \bmod 17 = 5 \cdot 16 \bmod 17 = 80 \bmod 17 = 12$$

El cifrado es, por tanto, $(14, 12)$.

Para descifrar, calculamos $c_1^a = 14^6 \bmod 17$. Como $14 \equiv -3 \bmod 17$, $(-3)^6 = 729 \bmod 17 = 16$. El inverso de 16 módulo 17 es 16, ya que $16 \cdot 16 \equiv 1 \pmod{17}$, por lo que

$$m = 12 \cdot 16 \bmod 17 = 192 \bmod 17 = 5$$

Cifrando otro mensaje y haciendo la operación del producto para comprobar la propiedad homomórfica obtendríamos un resultado distinto al esperado. Esto es debido a la acumulación de factores aleatorios (ruido multiplicativo) y a que los parámetros elegidos son demasiado pequeños y no cumplen con los requisitos de seguridad.

En la práctica, para garantizar la seguridad y el éxito de la operación, el grupo y los parámetros deben ser mucho mayores, ya que la seguridad del esquema se basa en la dificultad computacional de calcular logaritmos discretos en $G$. Además, el factor aleatorio se puede "eliminar" al descifrar correctamente bajo ciertas condiciones y para algunos usos prácticos.

ElGamal es útil en contextos donde el producto entre mensajes es la operación principal a realizar. Sin embargo, no permite realizar operaciones aditivas sobre los cifrados, y el tamaño del cifrado es aproximadamente el doble que el del mensaje original, ya que cada mensaje se cifra como un par de elementos del grupo. Además, la seguridad del esquema depende del problema del logaritmo discreto, cuya dificultad puede variar según la elección del grupo.

A pesar de estas limitaciones, ElGamal constituye un referente clásico y sirvió de inspiración para posteriores desarrollos.

## 4.3. El esquema de Paillier: cifrado homomórfico aditivo

El esquema de Paillier [5], propuesto en 1999, es uno de los sistemas de cifrado asimétrico más conocidos que presenta homomorfismo con respecto a la suma. Su seguridad se basa en la dificultad de ciertos problemas relacionados con la factorización de enteros.

La construcción se realiza sobre el grupo multiplicativo de los enteros módulo $N^2$, donde $N = pq$ es el producto de dos primos grandes.

- **KeyGen**: Elegir dos primos grandes $p$ y $q$ y calcular $N = pq$. Calcular el valor $\lambda = \text{mcm}(p-1, q-1)$. Elegir $g \in \mathbb{Z}_{N^2}^*$ tal que $\text{mcd}(L(g^\lambda \bmod N^2), N) = 1$, donde $L(u) = \frac{u-1}{N}$ (suele usarse $g = N+1$, porque simplifica mucho los cálculos y siempre cumple las propiedades requeridas). La clave pública es $(N, g)$ y la secreta es $\lambda$.

- **Enc**: Para cifrar un mensaje $m \in \mathbb{Z}_N$ (debe ser un número entre 0 y $N-1$), elegir un $r \in \mathbb{Z}_N^*$ al azar y calcular

$$c = g^m \cdot r^N \bmod N^2$$

- **Dec**: Dado el cifrado $c$, calcular

$$m = \frac{L(c^\lambda \bmod N^2)}{L(g^\lambda \bmod N^2)} \bmod N$$

  donde $L(u) = \frac{u-1}{N}$.

La propiedad homomórfica se observa en que el producto de dos cifrados corresponde a la suma de los mensajes:

$$c_1 \cdot c_2 = (g^{m_1} r_1^N) \cdot (g^{m_2} r_2^N) \equiv g^{m_1+m_2} (r_1 r_2)^N \pmod{N^2}$$

lo que equivale al cifrado de $m_1 + m_2$.

**Ejemplo 4.3.** Supongamos $p = 3$, $q = 5$, de modo que $N = 15$ y $N^2 = 225$. Tomamos $g = 16$. Calculamos $\lambda = \text{mcm}(2, 4) = 4$. Para cifrar $m = 7$, elegimos $r = 2$. Entonces,

$$c = 16^7 \cdot 2^{15} \text{ mód } 225$$

Calculando, $16^7 = 268435456$, $2^{15} = 32768$, y $c = (268435456 \cdot 32768)$ mód 225. Esto da $c = 128$ mód $225 = 128$.

Para descifrar, calculamos $c^\lambda$ mód $N^2 = 128^4$ mód $225 = 16$, y $L(16) = \frac{16-1}{15} = 1$. Por otro lado, $g^\lambda$ mód $N^2 = 16^4$ mód $225 = 61$, $L(61) = \frac{61-1}{15} = 4$. Finalmente,

$$m = \frac{1}{4} \text{ mód } 15 = 4 \text{ mód } 15 = 4$$

En este caso el resultado es 4, que no coincide con el mensaje original 7 porque los parámetros elegidos son muy pequeños, y además no se cumple con los requisitos de seguridad del esquema. La seguridad y la corrección de Paillier dependen crucialmente de tener suficiente "entropía" y de que los grupos sean lo bastante grandes, pero en este ejemplo se han elegido números más pequeños para mostrar el comportamiento algebraico de forma cómoda.

En la práctica, los valores de $p$ y $q$ son mucho mayores, lo que garantiza un correcto funcionamiento del algoritmo, y así se cumplen las condiciones necesarias para garantizar la seguridad del esquema.

El esquema de Paillier es útil en aplicaciones donde se requiere sumar datos cifrados, como en votaciones electrónicas y agregación de datos privados. Sin embargo, no permite multiplicar mensajes cifrados y el tamaño del cifrado es mayor que el del mensaje.

A pesar de ello, Paillier es un referente fundamental en el estudio de criptosistemas homomórficos y su diseño ha inspirado numerosos esquemas posteriores.

## 4.4. El esquema RSA: cifrado homomórfico multiplicativo

El esquema RSA [6], propuesto en 1978, es también uno de los sistemas de cifrado más conocidos y ampliamente utilizados en la criptografía moderna. Su seguridad se basa en la dificultad de factorizar números enteros grandes. RSA es homomórfico respecto al producto: permite multiplicar mensajes cifrados, pero no realizar sumas.

El sistema RSA trabaja sobre el grupo multiplicativo de los enteros módulo $N = pq$, donde $p$ y $q$ son dos primos grandes.

- **KeyGen**: Elegir dos primos grandes $p$ y $q$ y calcular $N = pq$. Elegir un exponente $e$ tal que $\text{mcd}(e, (p-1)(q-1)) = 1$ y calcular $d$ como el inverso multiplicativo de $e$ módulo $(p-1)(q-1)$, es decir, $ed \equiv 1 \pmod{(p-1)(q-1)}$. La clave pública es $(N, e)$ y la secreta es $d$.

- **Enc**: Para cifrar un mensaje $m \in \mathbb{Z}_N^*$, se calcula el cifrado como

$$c = m^e \text{ mód } N$$

- **Dec**: Para descifrar, se calcula

$$m = c^d \text{ mód } N$$

La propiedad homomórfica de RSA se refleja en que el producto de dos cifrados es el cifrado del producto de los mensajes:

$$c_1 \cdot c_2 = (m_1^e \text{ mód } N) \cdot (m_2^e \text{ mód } N) \equiv (m_1 m_2)^e \text{ mód } N$$

Es decir, si multiplicamos dos cifrados, obtenemos el cifrado del producto de los mensajes originales.

**Ejemplo 4.4.** Supongamos $p = 5$, $q = 11$, así que $N = 55$. Elegimos $e = 3$, que es coprimo con $(p-1)(q-1) = 4 \cdot 10 = 40$. Calculamos $d$ como el inverso de 3 módulo 40, es decir, $d = 27$ (ya que $3 \cdot 27 = 81 \equiv 1 \pmod{40}$).

Para cifrar el mensaje $m = 3$:

$$c = 3^3 \text{ mód } 55 = 27 \text{ mód } 55 = 27$$

Cifremos otro mensaje $m' = 4$, el cifrado es $c' = 4^3 \text{ mód } 55 = 64 \text{ mód } 55 = 9$.

Multiplicando los dos cifrados:

$$c \cdot c' = 27 \cdot 9 = 243 \text{ mód } 55 = 23$$

Y de esta forma podemos ver lo siguiente:

$$(m \cdot m')^e = (3 \cdot 4)^3 = 12^3 = 1728 \text{ mód } 55 = 23$$

Es decir, el producto de los cifrados corresponde al cifrado del producto de los mensajes, cumpliéndose así la propiedad homomórfica.

Para descifrar el producto, elevamos el resultado al exponente privado:

$$(c \cdot c')^d \text{ mód } 55 = 23^{27} \text{ mód } 55 = 12$$

Por lo que el descifrado del producto de los cifrados coincide exactamente con el producto de los mensajes originales, como garantiza la propiedad homomórfica multiplicativa de RSA.

RSA es eficiente y ampliamente utilizado en comunicaciones seguras, aunque no es adecuado para cifrado homomórfico general, ya que solo permite la multiplicación. Además, para garantizar la seguridad, los parámetros deben ser mucho mayores que en este ejemplo. A pesar de sus limitaciones, el esquema RSA sentó las bases para el desarrollo de la criptografía de clave pública y su propiedad homomórfica parcial lo convierte en un ejemplo clásico de PHE.

## 4.5. Limitaciones de los esquemas parcialmente homomórficos

Los esquemas presentados en este capítulo —Goldwasser–Micali, ElGamal, Paillier y RSA— constituyen ejemplos fundamentales de **cifrado parcialmente homomórfico** (**PHE**). Es decir, cada uno de ellos permite realizar un número ilimitado de operaciones de un único tipo (suma, producto o XOR) sobre los mensajes cifrados, pero no operaciones mixtas ni la evaluación general de circuitos aritméticos. A pesar de su relevancia teórica e histórica, esta restricción matemática es el principal motivo por el que estos sistemas no permiten computación arbitraria sobre datos cifrados, y condiciona severamente su aplicabilidad en escenarios reales o generales.

### 4.5.1. Limitaciones generales

- **Operación única:** Cada esquema PHE permite únicamente una operación algebraica sobre los mensajes cifrados:
  - Goldwasser–Micali: suma módulo 2 (XOR).
  - ElGamal y RSA: producto.
  - Paillier: suma.

  En consecuencia, no es posible implementar funciones o circuitos que combinen operaciones (por ejemplo, realizar sumas y productos alternados) en un mismo proceso homomórfico.

- **Espacio de mensajes restringido:** Goldwasser–Micali cifra solo bits individuales, y el resto de esquemas cifran enteros en un espacio limitado ($\mathbb{Z}_N$ o $\mathbb{Z}_N^*$). El tamaño de los mensajes y el de los cifrados suele estar restringido por la estructura algebraica subyacente y los parámetros elegidos.

- **Reducción modular e información perdida:** En esquemas que operan sobre enteros módulo $N$, como RSA y Paillier, toda la aritmética se realiza módulo $N$. Si el resultado de la operación sobre los mensajes en claro supera $N$, el descifrado devolverá el valor módulo $N$, sin que sea posible saber si hubo reducción.

  No hay manera, ni siquiera para el poseedor de la clave secreta, de distinguir si el resultado tras descifrar una operación (como una multiplicación en RSA) es el producto real de los mensajes o el resultado tras reducción modular.

  Esta limitación, que se deriva de la propia naturaleza de la aritmética modular, es inherente a estos sistemas e impide, por ejemplo, hacer un seguimiento fiable de sumas acumulativas o productos crecientes. Para evitar ambigüedades, habría que restringir los mensajes para garantizar que las operaciones nunca superen $N$, lo que reduce aún más la funcionalidad.

- **Crecimiento del tamaño de los cifrados:** En el esquema de Goldwasser–Micali, cifrar una cadena de bits requiere cifrar cada bit individualmente,

lo que implica un crecimiento lineal en el tamaño total del cifrado. En otros esquemas como Paillier o ElGamal, los mensajes cifrados son números enteros, pero cada mensaje cifrado ocupa al menos el doble o más del tamaño del mensaje original (dos elementos del grupo en ElGamal, uno en un módulo cuadrado en Paillier).

- **Sensibilidad a la elección de parámetros:** El correcto funcionamiento y la seguridad dependen de una cuidadosa elección de parámetros (primos suficientemente grandes, generadores adecuados, etc.). Parámetros pequeños o mal elegidos pueden hacer que el esquema deje de ser seguro, incluso que el descifrado falle o que se pierda unicidad en la recuperación del mensaje. Además, los requisitos sobre los generadores (por ejemplo, en Paillier) pueden no cumplirse de forma trivial para valores pequeños.

- **Ruido y aleatoriedad:** En esquemas probabilísticos, cada cifrado incluye aleatoriedad. Aunque esto aporta seguridad, también puede hacer que la composición de varios cifrados (por ejemplo, mediante operaciones repetidas) acumule factores aleatorios (ruido multiplicativo o aditivo), dificultando el descifrado y limitando la profundidad práctica de las operaciones, especialmente en el caso de ElGamal.

- **Falta de compacidad en operaciones iteradas:** Aunque la longitud de los cifrados no crece con cada operación homomórfica individual, la necesidad de cifrar muchos datos por separado (por ejemplo, todos los bits en Goldwasser–Micali) o de manejar múltiples componentes por mensaje (como en ElGamal) incrementa el coste de almacenamiento y transmisión.

- **Dependencia fuerte de la dificultad de problemas clásicos:** La seguridad de estos esquemas depende exclusivamente de problemas clásicos concretos de teoría de números:

  - Goldwasser–Micali: residuo cuadrático.
  - ElGamal: logaritmo discreto.
  - Paillier, RSA: factorización de números enteros grandes.

  Además de requerir parámetros de gran tamaño para garantizar la seguridad, muchos de estos problemas podrían verse afectados en el futuro por avances en algoritmos clásicos o cuánticos, lo que plantea cuestiones sobre la robustez a largo plazo.

### 4.5.2. Comparativa entre esquemas y limitaciones específicas

| | **Goldwasser – Micali** | **ElGamal** | **Paillier** | **RSA** |
|---|---|---|---|---|
| Tipo de operación | XOR (suma mod 2) | Producto | Suma | Producto |
| Espacio del mensaje | Bits | $\mathbb{Z}_p^*$ | $\mathbb{Z}_N$ | $\mathbb{Z}_N^*$ |
| Expansión del cifrado | Muy alta (uno por bit) | x2 | x2 | Igual que el mensaje |
| Reducción modular | - | Sí | Sí | Sí |
| Ruido aleatorio | Sí | Sí | Sí | No |
| Compacidad circuital | No | No | No | No |
| Seguridad basada en | Residuo cuadrático | Logaritmo discreto | Factorización | Factorización |
| Operaciones mixtas | No | No | No | No |

### 4.5.3. Conclusión

En resumen, aunque los esquemas PHE marcan hitos en la historia de la criptografía y permiten ilustrar claramente la noción de homomorfismo sobre datos cifrados, presentan limitaciones matemáticas y prácticas fundamentales para la computación general. La imposibilidad de realizar operaciones mixtas, la reducción incontrolable módulo $N$, el crecimiento de los cifrados y las restricciones en el espacio de mensajes motivan la búsqueda de esquemas completamente homomórficos (FHE), capaces de soportar la evaluación de cualquier función computable de forma segura y general sobre datos cifrados.

# Capítulo 5

# Construcción matemática del FHE

En los capítulos anteriores se ha presentado el contexto teórico y matemático necesario para abordar el estudio de los esquemas de cifrado homomórfico, así como una revisión detallada de los primeros ejemplos clásicos: Goldwasser–Micali, ElGamal, Paillier y RSA. Todos estos sistemas comparten la propiedad de permitir la realización de ciertas operaciones algebraicas directamente sobre los textos cifrados, es decir, exhiben algún tipo de homomorfismo respecto a la suma, el producto o el operador XOR. Sin embargo, como hemos visto, tienen varias limitaciones inherentes, principalmente que su capacidad está restringida a operaciones concretas o a un número limitado de éstas, lo que impide el tratamiento general de funciones arbitrarias sobre datos cifrados.

El propósito de este capítulo es exponer el salto conceptual y matemático que conduce a la construcción de un esquema de cifrado completamente homomórfico (FHE), capaz de soportar la evaluación de cualquier circuito, es decir, de cualquier función computable, sobre datos encriptados. Analizaremos los fundamentos matemáticos que hacen posible este avance, veremos los retos asociados (como la gestión del ruido y los mecanismos como el *bootstrapping*), y sentaremos las bases para la descripción rigurosa de los principales esquemas de FHE en los capítulos posteriores. Todos los detalles se pueden consultar en [7], [8], [9] y [1].

## 5.1. Circuitos booleanos y su evaluación

### 5.1.1. Definición formal de circuitos booleanos

**Definición 5.1.** Un **circuito booleano** es una representación matemática de una función $f : \{0,1\}^n \to \{0,1\}$ construida a partir de la composición finita de puertas lógicas elementales e interconectadas, donde cada una recibe como entradas el resultado de otras puertas o las variables iniciales, y su salida puede servir a su vez como entrada para puertas posteriores.

Es habitual modelar un circuito booleano mediante un grafo dirigido sin ciclos, en el que los nodos representan las puertas lógicas y las aristas indican cómo se transmiten los valores desde las entradas hacia las salidas a través de las distintas puertas. De este modo, el circuito define una secuencia de operaciones lógicas que

transforma un vector de bits de entrada en un único bit de salida.

Las puertas lógicas básicas suelen ser AND ($\wedge$), OR ($\vee$), NOT ($\neg$) y, en algunos casos, XOR ($\oplus$). Mediante la combinación de estas operaciones, cualquier función booleana puede expresarse como un circuito finito. Además, resulta bien conocido que el conjunto formado por las puertas NAND o NOR es funcionalmente completo, es decir, basta una sola de ellas para generar cualquier circuito lógico.

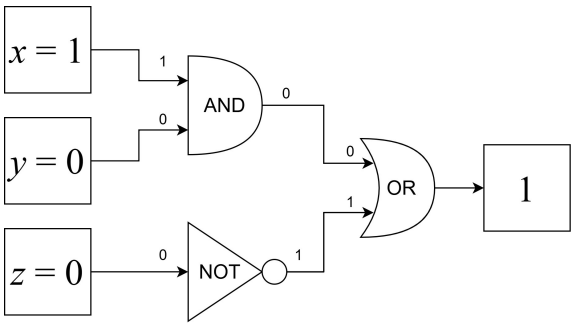


Figura 5.1: Ejemplo de circuito booleano para $f(x, y, z) = (x \wedge y) \vee (\neg z)$.

Como ya hemos visto en la sección 2.3, existe una traducción directa entre las operaciones lógicas y las operaciones algebraicas sobre el cuerpo finito $\mathbb{F}_2$:

$$\begin{aligned} \mathrm{NOT}(x) &= 1 - x \\ \mathrm{AND}(x, y) &= x \cdot y \\ \mathrm{OR}(x, y) &= x + y - x \cdot y \\ \mathrm{XOR}(x, y) &= x + y \pmod 2 \end{aligned}$$

Esta correspondencia permite representar cualquier función booleana como un polinomio sobre $\mathbb{F}_2$, lo que resulta fundamental en el contexto homomórfico: la operación sobre los cifrados refleja la estructura algebraica del circuito lógico subyacente.

**Ejemplo 5.1.** Consideremos la función *Majority* de tres variables, que toma valor 1 si al menos dos de sus entradas son 1. Es decir, el circuito booleano $f(x, y, z) = ((x \wedge y) \vee (x \wedge z)) \vee (y \wedge z)$. Su expresión polinómica sobre $\mathbb{F}_2$ es:

$$\mathrm{Majority}(x, y, z) = xy + xz + yz$$

La posibilidad de expresar cualquier función computable como un circuito booleano finito es el fundamento teórico sobre el que se asienta el cifrado completamente homomórfico (FHE). Esta propiedad de universalidad implica que, en principio, cualquier operación algorítmica puede ser representada como una secuencia finita de operaciones lógicas, y por tanto, puede ser trasladada al dominio cifrado siempre que el esquema homomórfico lo permita. En la práctica, esto significa que el cifrado homomórfico se convierte en una herramienta potencialmente universal para el procesamiento seguro de la información.

### 5.1.2. Profundidad y tamaño de circuitos

El análisis de la complejidad de los circuitos booleanos requiere distinguir entre dos parámetros fundamentales: la **profundidad** y el **tamaño** del circuito. Estos conceptos determinan la complejidad computacional de la función implementada y, en el contexto del FHE, condicionan la cantidad de operaciones homomórficas necesarias y el crecimiento del ruido, de forma que afectan directamente a la eficiencia y viabilidad de la evaluación sobre datos cifrados.

**Definición 5.2.** El **tamaño de un circuito** es el número total de puertas lógicas que lo componen, independientemente de cómo estén conectadas. El tamaño proporciona una medida global de la cantidad de operaciones elementales necesarias para implementar la función booleana asociada.

Este parámetro afecta al coste computacional total: circuitos muy grandes requieren más operaciones homomórficas, lo que incrementa el tiempo de cómputo y el consumo de recursos.

**Definición 5.3.** La **profundidad de un circuito** es la longitud máxima de un camino desde una entrada hasta la salida, es decir, es el máximo número de puertas lógicas que cualquier señal de entrada debe atravesar antes de contribuir al valor de salida. La profundidad cuantifica la dependencia secuencial: circuitos de profundidad pequeña pueden evaluarse con alta paralelización, mientras que una gran profundidad indica la necesidad de múltiples etapas secuenciales.

Desde el punto de vista de la teoría de la complejidad, la *profundidad de un circuito* está relacionada con el tiempo de cómputo en modelos paralelos: para algunas funciones clásicas, como la suma de $n$ bits, pueden construirse circuitos de tamaño $O(n)$ y profundidad $O(\log n)$ mediante estrategias como árboles binarios (que buscan realizar operaciones en paralelo en vez de hacerlas todas en serie). Sin embargo, existen funciones que requieren necesariamente una profundidad mínima de $\Omega(\log n)$, lo que impone límites fundamentales a la eficiencia de su evaluación. Es decir, existen límites teóricos a qué tan rápido se puede calcular la salida de un circuito, incluso aunque se sigan estrategias para construir el circuito agrupando, paralelizando y optimizando todas las operaciones lógicas posibles.

En el contexto del cifrado homomórfico, estos límites teóricos tienen consecuencias prácticas directas: la profundidad del circuito determina cuántas operaciones homomórficas (especialmente multiplicaciones) pueden aplicarse antes de que el crecimiento del ruido vuelva inviable el descifrado, restringiendo así la clase de funciones que pueden evaluarse de forma segura sobre datos cifrados.

Desde un punto de vista matemático, es posible diseñar circuitos que, a igualdad de función computada, optimicen uno u otro parámetro según las necesidades. El equilibrio entre ambos es un tema central en el diseño eficiente de algoritmos y en la viabilidad de su evaluación segura sobre datos cifrados.

### 5.1.3. Universalidad y conexión con los esquemas homomórficos clásicos

En los esquemas clásicos presentados (Goldwasser–Micali, ElGamal, Paillier y RSA) la evaluación homomórfica está limitada a operaciones concretas (XOR, producto o suma). Matemáticamente, esto equivale a restringir el conjunto de circuitos que pueden evaluarse a aquellos expresables con las operaciones soportadas por el esquema. Como resultado, la capacidad de computación de estos sistemas es muy limitada: no pueden, en general, evaluar circuitos arbitrarios ni implementar funciones complejas sobre datos cifrados.

El desafío fundamental del cifrado completamente homomórfico (FHE) es precisamente superar esa limitación y lograr que sea posible evaluar de manera eficiente y segura cualquier función computable sobre textos cifrados, es decir, **alcanzar la universalidad práctica**. Para alcanzar este objetivo, un esquema FHE debe satisfacer dos propiedades clave:

- **Capacidad de evaluación arbitraria:** el esquema debe permitir la evaluación de cualquier circuito booleano finito, es decir, de cualquier combinación de puertas lógicas elementales, sobre datos cifrados.
- **Compacidad:** tal y como se ha definido formalmente en el capítulo 3, el tamaño del texto cifrado resultante tras la evaluación de un circuito debe depender únicamente del parámetro de seguridad, y no de la complejidad o profundidad del circuito evaluado.

Cuando se cumplen ambas propiedades, se dice que el esquema es **universal** en el sentido homomórfico: permite trasladar cualquier cálculo algorítmico al dominio cifrado, manteniendo la viabilidad teórica (tipo de operaciones) y práctica (tamaño). La universalidad, por tanto, no es solo una propiedad aislada, sino la consecuencia matemática de alcanzar simultáneamente la evaluación arbitraria y la compacidad. Esta aspiración, que permaneció abierta durante décadas, es la que motiva el desarrollo de los esquemas FHE modernos, en los que retos como la gestión del ruido y la estructura algebraica subyacente son factores determinantes.

A partir de este marco conceptual, en los siguientes apartados trataremos las dificultades y soluciones matemáticas involucradas en la construcción efectiva de esquemas FHE.

## 5.2. Ruido y limitaciones del SHE

### 5.2.1. Origen y significado del ruido en esquemas homomórficos

Uno de los conceptos fundamentales en el estudio de los esquemas de cifrado homomórfico es el de **ruido**. En términos generales, el ruido es un valor numérico o algebraico que se añade intencionadamente al mensaje durante el cifrado, y cuya presencia permite garantizar la seguridad criptográfica del esquema.

Matemáticamente, en la mayoría de los esquemas homomórficos, el proceso de cifrado se modela mediante una función probabilística

$$c = \mathrm{Enc}(\mathrm{pk}, m; r),$$

donde $pk$ es la clave pública, $m$ es el mensaje y $r$ es un vector de aleatoriedad elegido en un conjunto adecuado.

Esta aleatoriedad se traduce, en la práctica, en la aparición de un término adicional: el ruido, que, aunque no impide la correcta recuperación del mensaje durante el descifrado (siempre que su magnitud se mantenga dentro de ciertos límites), dificulta enormemente que un atacante pueda extraer información del texto cifrado sin conocer la clave secreta.

El diseño de los esquemas homomórficos aprovecha el hecho de que, mientras el ruido se mantenga bajo ciertos umbrales, el descifrado seguirá funcionando correctamente. No obstante, cada vez que se realiza una operación homomórfica, el nivel de ruido en el resultado tiende a incrementarse. Este fenómeno, en última instancia, limita la potencia de los esquemas homomórficos y separa los esquemas "algo homomórficos" (SHE) de los completamente homomórficos (FHE).

Por tanto, el análisis del ruido y su evolución a lo largo de las operaciones homomórficas es esencial para comprender las posibilidades y limitaciones de un esquema de cifrado homomórfico.

### 5.2.2. Acumulación y crecimiento del ruido

Una de las características fundamentales de los esquemas de cifrado homomórfico es que el ruido presente en los textos cifrados no permanece constante a lo largo de las operaciones, sino que tiende a incrementarse cada vez que se realiza una operación homomórfica, especialmente en el caso de las multiplicaciones. Este fenómeno se conoce como *acumulación del ruido* y es la principal razón por la que, en los esquemas algo homomórficos (SHE), solo pueden evaluarse circuitos de profundidad limitada.

Veamos de forma más precisa cómo se comporta el ruido en un esquema homomórfico: sea $c_1 = \mathrm{Enc}(pk, m_1; r_1)$ y $c_2 = \mathrm{Enc}(pk, m_2; r_2)$ dos textos cifrados, donde el proceso de cifrado introduce en cada uno un término de ruido, denotados por $e_1$ y $e_2$, respectivamente. En la mayoría de los esquemas, el texto cifrado se puede escribir en la forma general

$$c = \mathrm{Enc}(\mathrm{pk}, m; r) = \alpha(m) + \beta(r) + e,$$

donde $\alpha$ y $\beta$ son funciones determinadas por el esquema, $r$ es la aleatoriedad empleada y $e$ representa el ruido inicial introducido por la aleatoriedad.

Supongamos ahora que el esquema soporta suma homomórfica. Si sumamos dos textos cifrados, el resultado es

$$c_{\mathrm{suma}} = c_1 + c_2 = \alpha(m_1) + \alpha(m_2) + \beta(r_1) + \beta(r_2) + (e_1 + e_2)$$

Como las funciones $\beta(r_1)$ y $\beta(r_2)$ suelen estar bien controladas,, a la hora de descifrar no supondrán un problema. Por lo tanto, el ruido tras la suma es simplemente $e_1+e_2$, es decir, crece de manera aditiva con el número de sumas.

En el caso de la multiplicación homomórfica, el análisis es más delicado. El producto de dos textos cifrados $c_1 \cdot c_2$ se comporta normalmente como

$$c_{\text{mult}} = \alpha(m_1)\alpha(m_2) + (\text{términos mixtos de } \beta(r_i) \text{ y } \alpha(m_i)) + (\text{términos de ruido con } e_i)$$

donde los términos de ruido combinados contienen productos con $e_1$ y $e_2$. Por ejemplo, en los esquemas sobre los enteros, tras cada multiplicación el nuevo ruido puede estimarse como

$$e_{\text{mult}} = \alpha(m_1)e_2 + \alpha(m_2)e_1 + e_1e_2$$

Los términos con $\beta(r_i)$ se desprecian porque, en general, quedan "absorbidos" durante el proceso de descifrado. El crecimiento del ruido está así determinado principalmente por el término $e_1e_2$, que se multiplica en cada operación y, tras $k$ multiplicaciones, puede crecer de forma exponencial con $k$. Los demás términos suelen crecer solo lineal o polinómicamente.

En términos generales, si partimos de un ruido inicial $e_0$ y aplicamos $k$ multiplicaciones consecutivas, el ruido puede alcanzar una cota del tipo

$$e_k \lesssim (e_0)^{2^k},$$

lo que ilustra su crecimiento potencialmente exponencial en función de la profundidad del circuito. Veamos la explicación formal con una cota real:

**Límite matemático para la profundidad evaluable.** Sea $e_0$ el ruido inicial introducido por el cifrado y $\tau$ el umbral máximo de ruido permitido para garantizar la corrección. Tras $d$ multiplicaciones sucesivas, el ruido puede aumentar hasta una cota

$$e_d \leq e_0 \cdot \gamma^d,$$

donde $\gamma > 1$ es la tasa de crecimiento del ruido característica del esquema. Para que el descifrado sea correcto, es necesario que $e_d < \tau$, lo que impone una cota para la profundidad máxima del circuito evaluable:

$$d_{\text{máx}} < \frac{\log(\tau/e_0)}{\log \gamma}.$$

Así, la profundidad de los circuitos que pueden evaluarse de manera segura en un esquema SHE está restringida por la relación entre el ruido inicial, el crecimiento por operación y el umbral de corrección del esquema (máximo ruido permitido).

En este punto es cuando la propiedad de la **compacidad** es fundamental. En los esquemas SHE, aunque el tamaño del cifrado y el coste de descifrado suelen crecer solo lineal o polinómicamente con cada operación individual, la acumulación de ruido implica que, al aumentar la profundidad del circuito, puede ser necesario aumentar los parámetros o incluso el tamaño de los cifrados para garantizar la corrección, perdiendo así la compacidad

Por esta razón, los esquemas SHE imponen una cota máxima a la profundidad de los circuitos evaluables, es decir, solo permiten una cantidad limitada de operaciones sobre datos cifrados, determinada por el umbral a partir del cual el ruido impide

el descifrado correcto. Esta acumulación de ruido constituye el principal cuello de botella matemático en la construcción de cifrados homomórficos potentes, pues la gestión del ruido y la compacidad están muy relacionados, y constituyen la principal diferencia práctica y teórica entre los esquemas SHE y FHE.

Por todo ello, el objetivo fundamental de los esquemas completamente homomórficos (FHE) es superar esta limitación y permitir la evaluación de circuitos de cualquier profundidad sobre datos cifrados, haciendo posible la computación arbitraria en el dominio cifrado.

## 5.3. El esquema de Gentry

### 5.3.1. Construcción general

El punto de partida de Gentry [10] es un esquema algo homomórfico (SHE) capaz de realizar sumas y productos sobre textos cifrados, pero sólo hasta una profundidad limitada debido al crecimiento acumulado del ruido en cada operación como hemos visto en la sección anterior.

En cualquier esquema SHE, el proceso de descifrado está implementado mediante un circuito aritmético o booleano que toma como entrada el texto cifrado y la clave secreta, y produce como salida el mensaje original. Formalmente, podemos expresar este proceso como

$$\mathrm{Dec}(\mathrm{sk}, c) = f(c, \mathrm{sk}) = m,$$

donde sk es la clave secreta y $f$ es una combinación de operaciones algebraicas. Sin embargo, la profundidad y complejidad de este circuito de descifrado actúan como una barrera: a medida que se realizan más operaciones homomórficas, el ruido acumulado en los textos cifrados aumenta, y llega un momento en que la corrección del descifrado ya no puede garantizarse al exceder un cierto umbral.

El avance de Gentry consiste en buscar un esquema SHE que sea *bootstrappable*, es decir, suficientemente potente como para evaluar, además de operaciones aritméticas básicas, el circuito de descifrado del propio esquema en el dominio cifrado. Si se logra esta propiedad, es posible aplicar una técnica denominada **bootstrapping**, cuyo objetivo es "reiniciar" el nivel de ruido en los textos cifrado, permitiendo así evaluar circuitos de cualquier profundidad.

La idea clave es que, mediante la evaluación homomórfica del circuito de descifrado sobre *el cifrado de un texto cifrado* (junto con una versión cifrada de la clave secreta), se obtiene un nuevo cifrado del mismo mensaje, pero con el ruido reducido a niveles similares a los de un cifrado fresco.

### 5.3.2. Bootstrapping y promoción a FHE

El **bootstrapping** es la técnica clave introducida por Gentry para transformar un esquema algo homomórfico (SHE) en un esquema completamente homomórfico (FHE). La idea fundamental es eliminar el ruido acumulado en un texto cifrado,

permitiendo así que el esquema soporte un número ilimitado de operaciones homomórficas.

Formalmente, decimos que un esquema SHE es **bootstrappable** si puede evaluar homomórficamente su propio circuito de descifrado, es decir, si para un texto cifrado $c$ y una clave secreta sk existe una función tal que

$$c' = \text{Eval}(\text{evk}, Dec, c^*, \text{sk}^*)$$

donde $c^*$ y $\text{sk}^*$ son versiones cifradas de $c$ y de la clave secreta, y $Dec$ representa el circuito de descifrado del esquema. El resultado $c'$ es un nuevo cifrado del mismo mensaje, pero con el ruido reducido a los niveles de un cifrado "fresco". Así, el proceso de bootstrapping consiste en evaluar el algoritmo de descifrado sobre datos cifrados, sin exponer en ningún momento la información en claro.

Este proceso necesita que el circuito de descifrado sea lo suficientemente sencillo para que el esquema SHE pueda evaluarlo antes de que el ruido acumulado supere el umbral permitido. Si esto se cumple, el esquema puede "refrescar" cualquier cifrado mediante bootstrapping, eliminando el ruido acumulado y permitiendo que la computación homomórfica continúe.

Gentry pone el foco en la profundidad de este circuito de descifrado, ya que representa la barrera matemática para extender un esquema SHE a un esquema FHE. Si esta profundidad excede la capacidad de evaluación segura del esquema ($d_{\text{máx}}$, como hemos visto anteriormente), el proceso de refresco del ruido no puede realizarse.

En resumen, la profundidad y complejidad del circuito de descifrado es el elemento central que determina si un esquema SHE puede ser promovido a un esquema completamente homomórfico.

### 5.3.3. Implementación y condiciones necesarias

La implementación del bootstrapping implica varios pasos y condiciones matemáticas que deben cumplirse para que el proceso de refresco del ruido sea efectivo y seguro.

El mecanismo fundamental consiste en realizar la operación de descifrado sobre datos que permanecen siempre cifrados. Para ello, el esquema debe poder codificar el texto cifrado $c$ (que descifra al mensaje $m$) y la clave secreta sk dentro del espacio de cifrado. Es decir, el evaluador dispone de $c^* = \text{Enc}(\text{pk}, c)$ y $\text{sk}^* = \text{Enc}(\text{pk}, \text{sk})$, y utiliza el algoritmo de evaluación homomórfica para computar

$$c' = \text{Eval}(\text{evk}, Dec, c^*, \text{sk}^*)$$

Para que el bootstrapping sea aplicable, el esquema debe ser capaz de:

- Representar el circuito de descifrado con una profundidad suficientemente baja para que el SHE pueda evaluarlo sin problemas.
- Garantizar la **corrección** de la evaluación: el resultado $c'$, al igual que $c$, debe descifrar correctamente al mensaje original $m$, sin introducir errores debidos al ruido.

- Preservar la **seguridad**: ni el proceso de evaluación ni la publicación de la clave secreta cifrada deben comprometer la confidencialidad del mensaje o la clave.

La principal restricción es que la profundidad del circuito de descifrado debe estar estrictamente por debajo de la profundidad máxima que el esquema SHE puede manejar, $d_{\text{máx}}$. Este requisito obliga a diseñar esquemas con operaciones de descifrado sencillas, o a introducir técnicas auxiliares (como el *squashing*, que veremos en la siguiente sección) para reducir la complejidad.

El éxito del bootstrapping depende, por tanto, de la estructura algebraica subyacente y de la optimización de los algoritmos internos de cifrado, evaluación y descifrado. Solo cuando se cumplen estas condiciones es posible transformar el SHE en un esquema FHE plenamente funcional.

### 5.3.4. Squashing del circuito de descifrado

En la práctica, uno de los principales retos para implementar el bootstrapping es que el circuito de descifrado de muchos esquemas SHE resulta demasiado profundo para ser evaluado homomórficamente. Para superar este obstáculo, Gentry introduce la técnica del *squashing* del circuito de descifrado, cuyo objetivo es reducir la profundidad de este circuito sin alterar su funcionalidad.

El **squashing** consiste en reescribir el proceso de descifrado como un circuito más sencillo, normalmente a costa de añadir información auxiliar al sistema. En la construcción original de Gentry, se incorpora una llamada "clave de ayuda" (*hint key*), que permite expresar una parte del descifrado (como un producto interno o una suma pesada) mediante una combinación de operaciones de menor profundidad. Así, el circuito de descifrado resultante puede ser evaluado dentro de los límites impuestos por el ruido.

Formalmente, se trata de transformar el circuito original de descifrado Dec, de profundidad $d_{dec}$, en un circuito equivalente $\text{Dec}'$ de profundidad reducida $d'_{dec}$, donde

$$d'_{dec} < d_{\text{máx}}$$

Pero esta reducción de profundidad no es gratuita: introducir información auxiliar puede afectar a la seguridad del sistema, por lo que debe garantizarse que no se filtra información sobre la clave secreta ni sobre los mensajes. Por ello, la técnica de *squashing* debe estudiarse cuidadosamente desde el punto de vista criptográfico, y así asegurar que el nuevo esquema mantiene la seguridad.

En resumen, el *squashing* del circuito de descifrado es un paso intermedio necesario para que el bootstrapping sea viable en la práctica, permitiendo aplicar el mecanismo de refresco del ruido incluso en esquemas donde el descifrado original sería demasiado complejo para ser evaluado homomórficamente.

### 5.3.5. Coste computacional y otras limitaciones

Además del desafío de la seguridad y de la profundidad del circuito de descifrado, tenemos otros obstáculos a la hora de aplicar el bootstrapping:

- El primer aspecto a considerar es el **coste de evaluación** del circuito de descifrado: la evaluación homomórfica de un circuito de descifrado sobre datos cifrados requiere calcular una función de profundidad $d_{dec}$ y tamaño $s_{dec}$. Como consecuencia, el tiempo necesario para llevar a cabo el bootstrapping, denotado $T_{bs}$, depende tanto del tamaño como de la complejidad aritmética del circuito:

  $$T_{bs} = O(s_{dec} \cdot t_{op})$$

  donde $t_{op}$ es el tiempo necesario para una operación homomórfica elemental (como una suma o una multiplicación).

  Sin embargo, en la práctica, tanto el tamaño como la profundidad de los circuitos de descifrado suelen ser muy superiores a los de las operaciones básicas. Esto provoca que el proceso de bootstrapping sea *la operación más costosa* dentro de un esquema FHE. En la construcción original de Gentry, el bootstrapping dominaba el tiempo de cómputo total, representando un verdadero cuello de botella para la eficiencia.

- Otro aspecto fundamental es el **aumento temporal del ruido** durante la evaluación: aunque el resultado final es un cifrado "fresco", durante la computación intermedia el ruido puede crecer de forma importante debido a la cantidad de operaciones homomórficas realizadas. Esto obliga a ajustar con mucho detalle los parámetros del esquema para preservar la corrección del descifrado.

- Finalmente, la **seguridad** puede verse afectada por el uso de técnicas como el *squashing*: para reducir la profundidad del circuito, suele ser necesario añadir información auxiliar (como una *hint key*) a la clave pública. Si este proceso no se diseña con cuidado puede introducir vulnerabilidades inesperadas.

En resumen, el coste computacional del bootstrapping justifica la necesidad de encontrar esquemas que lo optimicen o incluso lo eviten parcialmente, y motiva la búsqueda de construcciones más eficientes, como las que presentaremos en los capítulos siguientes.

## 5.4. Observaciones finales

La construcción de Gentry fue un hito en la historia de la criptografía moderna, al demostrar que un cifrado completamente homomórfico es posible. Su gran idea fue usar el *bootstrapping* para refrescar el ruido y así poder hacer cualquier número de operaciones sobre datos cifrados, algo que antes se consideraba imposible.

Pero la propuesta de Gentry tiene varios problemas prácticos. El proceso de bootstrapping es muy costoso en tiempo y recursos, y para poder llevarlo a cabo debemos transformar el circuito de descifrado mediante herramientas como el

*squashing*, que complican aún más el proceso tanto por la eficiencia como por la seguridad.

Por eso, después de la construcción de Gentry han surgido otras variantes que buscan superar estos obstáculos. En los siguientes capítulos veremos dos de estas líneas principales: por un lado, los esquemas sobre los enteros, como DGHV, que permiten entender mejor cómo se comporta el ruido y simplifican la estructura algebraica; y por otro lado, los esquemas basados en retículos (como LWE y RLWE), que suponen grandes mejoras en eficiencia y son la base de la mayoría de los FHE modernos y actuales.

En definitiva, la idea de Gentry abrió un nuevo campo en la criptografía y sigue siendo el punto de partida para entender cómo calcular de forma segura con datos cifrados.

# Capítulo 6

# FHE sobre los enteros

## 6.1. Introducción

El desarrollo de esquemas de cifrado completamente homomórfico (FHE) representa uno de los mayores avances recientes de la criptografía. La construcción de FHE permitió, por primera vez, realizar operaciones arbitrarias sobre datos cifrados, sin necesidad de acceder a la información original. Esta propiedad tiene gran relevancia para la privacidad y la seguridad en contextos como el almacenamiento en la nube, el procesamiento externo de datos sensibles o el diseño de sistemas criptográficos avanzados .

En este capítulo se presenta el esquema de cifrado completamente homomórfico sobre los enteros, conocido como DGHV. A diferencia de otros enfoques más abstractos, el esquema DGHV parte de construcciones aritméticas elementales, utilizando operaciones sobre los números enteros y la aritmética modular.

Primero veremos el funcionamiento general del esquema, con la estructura de claves, el proceso de cifrado y descifrado, y las operaciones permitidas. A continuación, se explorarán los fundamentos matemáticos con especial atención al papel del ruido y su evolución. Luego, se analizarán las limitaciones inherentes al crecimiento del ruido y la solución del *bootstrapping*. Por último, estudiaremos la seguridad del esquema, basada en el problema del máximo común divisor aproximado, y se expondrá un ejemplo completo.

## 6.2. Esquema DGHV

El esquema DGHV, propuesto por van Dijk, Gentry, Halevi y Vaikuntanathan en 2010 [8], es un ejemplo de cifrado completamente homomórfico que opera sobre los números enteros mediante aritmética modular y sumas y productos elementales. A continuación se describen sus componentes y notación principales.

### 6.2.1. Algoritmo

#### Parámetros y notación

Sea $p$ un entero impar suficientemente grande que actúa como clave secreta. El tamaño de $p$ (de $\eta$ bits, normalmente cientos o miles de bits) determina la seguridad y la tolerancia al ruido. Por otro lado, la clave pública es una colección de enteros $\mathbf{x} = (x_0, x_1, \ldots, x_\tau)$.

Se consideran también estos parámetros:

- $\rho$ y $\rho'$, cotas del ruido introducido en cada cifrado y en cada operación respectivamente.
- $\gamma$, tamaño (en bits) de los $x_i$.
- $\tau$ es el número de elementos públicos $x_i$ (además de $x_0$) que se generan.

El mensaje a cifrar será siempre un bit $m \in \{0, 1\}$.

#### Generación de claves

- **Clave secreta:** Se elige un entero impar aleatorio $p$ de tamaño $\eta$ bits.
- **Clave pública:** Se genera una colección de enteros $\mathbf{x} = (x_0, x_1, \ldots, x_\tau)$, donde cada uno se construye como

$$x_i = q_i p + 2r_i,$$

con $q_i$ un entero grande aleatorio, y $r_i$ un entero pequeño acotado que representa el ruido, $|r_i| < 2^\rho$. Se fuerza que $x_0$ sea el mayor y que $x_0 \bmod p$ sea par.

La razón de elegir $p$ impar es que, a la hora de descifrar, se debe garantizar que se pueda distinguir el bit menos significativo tras reducir módulo $p$. Los $x_i$ deben ser grandes y "aleatoriamente desplazados" una distancia par respecto a múltiplos de $p$ para dificultar cualquier intento de deducir $p$. Finalmente, $x_0 \bmod p$ debe ser par porque en el cifrado y las operaciones reduciremos el resultado módulo $x_0$, y debemos evitar que se introduzca ruido impar "oculto" al hacer esta reducción.

#### Cifrado

Para cifrar un mensaje $m \in \{0, 1\}$:

1. Se escoge un subconjunto de elementos de la clave pública con $S \subset \{1, \ldots, \tau\}$ de forma aleatoria.
2. Se elige un ruido $r$ aleatorio con $|r| < 2^{\rho'}$ para cierto $\rho' > \rho$.
3. El cifrado se define como

$$c = m + 2r + 2\sum_{i \in S} x_i \pmod{x_0}.$$

El mensaje queda "escondido" dentro de $c$, y la suma sobre los $x_i$ asegura aleatoriedad adicional.

El subconjunto $S$ elegido en cada cifrado permite que, incluso cifrando el mismo mensaje varias veces, el resultado sea diferente. Sumar $2\sum_{i\in S} x_i$ añade muchos posibles valores de "cero cifrado" y hace que el esquema sea semánticamente seguro. El término $2r$ introduce ruido adicional, dificultando la obtención del mensaje original sin la clave. Finalmente, todo el resultado se toma módulo $x_0$ para evitar que el valor cifrado sea par o impar dependiendo de $m$, y además así el resultado es siempre más pequeño que $x_0$ y se evita un crecimiento descontrolado.

**Descifrado**

El descifrado se basa en la reducción módulo $p$ y la extracción del bit menos significativo:

$$m = (c \bmod p) \bmod 2$$

Es decir, primero se reduce el cifrado módulo la clave secreta $p$, y luego se extrae el bit menos significativo del resultado.

Este procedimiento recupera el mensaje original siempre que el término de ruido no sea demasiado grande en valor absoluto, es decir, $|c - kp| < p/2$ para algún $k \in \mathbb{Z}$ (el cifrado está "cerca" de algún múltiplo de $p$). Esta condición será fundamental en las siguientes secciones.

### 6.2.2. Aritmética modular en DGHV

El esquema está construido sobre congruencias entre enteros. El proceso de cifrado puede verse como la ocultación del bit $m$ dentro de una suma de múltiplos de la clave secreta $p$, de la forma $q_i p$ , a los que se añade un pequeño "desplazamiento" (el ruido) y una reducción módulo $x_0$. Dado que el atacante no conoce $p$, no puede distinguir el mensaje del ruido, y la aleatoriedad introducida por la elección del subconjunto $S$ asegura que cada cifrado es prácticamente independiente del anterior. Tras la reducción módulo $p$, desaparezcan las contribuciones de los términos que sean múltiplos de $p$, quedando únicamente el mensaje y el ruido. Así, si el cifrado es

$$c = m + 2r + 2\sum_{i\in S} x_i \pmod{x_0},$$

y recordando que cada $x_i = q_i p + 2r_i$, se tiene

$$c = m + 2r + 2\sum_{i\in S}(q_i p + 2r_i) \pmod{x_0}.$$

Expandiendo,

$$c = m + 2r + 2\sum_{i\in S} q_i p + 4\sum_{i\in S} r_i \pmod{x_0}.$$

Al reducir módulo $p$, todos los términos con factor $p$ desaparecen, y el resultado es

$$c \bmod p = m + 2r + 4\sum_{i\in S} r_i \bmod p.$$

El descifrado extrae el bit menos significativo:

$$m = (c \bmod p) \bmod 2.$$

### 6.2.3. El papel del ruido

El término de ruido, en la expresión anterior,

$$\text{ruido total} = 2r + 4\sum_{i \in S} r_i,$$

debe permanecer controlado para que el proceso de descifrado funcione correctamente. La condición fundamental es que el valor absoluto del ruido total sea significativamente menor que $p/2$. Así se asegura que, al reducir módulo $p$, no se produce un "desbordamiento" que pueda alterar el resultado del bit menos significativo.

Este límite es crucial: si el ruido crece demasiado (por la acumulación debida a operaciones homomórficas sucesivas), el mensaje original podría quedar "oculto" por el propio ruido, provocando errores en el descifrado. Esta restricción motiva la necesidad de técnicas para eliminar o reducir el ruido, como el *bootstrapping*, que se abordarán en secciones posteriores.

**Lema 6.1** (Corrección del descifrado)**.** El algoritmo de descifrado recupera el mensaje original si el *ruido total* $|2r + 4\sum_{i \in S} r_i| < p/2$.

*Demostración.* Sea $x_i = q_i p + 2r_i$. Entonces,

$$c = m + 2r + 2\sum_{i \in S} x_i \pmod{x_0} = m + 2r + 2\sum_{i \in S}(q_i p + 2r_i) \pmod{x_0}$$

$$= m + 2r + 2\sum_{i \in S} q_i p + 4\sum_{i \in S} r_i \pmod{x_0}$$

Al reducir módulo $p$, desaparecen los múltiplos de $p$:

$$c \bmod p = m + 2r + 4\sum_{i \in S} r_i \bmod p$$

Sea $e = 2r + 4\sum_{i \in S} r_i$ el ruido total. Si $|e| < p/2$, la reducción módulo $p$ da exactamente $m + e$, y por lo tanto:

$$(c \bmod p) \bmod 2 = m$$

No hay "desbordamientos" circulares, y de esta forma el ruido no impide recuperar el mensaje original. □

### 6.2.4. Operaciones homomórficas

**Definición 6.1** (Operaciones homomórficas)**.** Sean $c_1$ y $c_2$ dos cifrados, correspondientes a $m_1, m_2 \in \{0,1\}$, definimos las siguientes operaciones:

- Suma: $c_{\text{suma}} = c_1 + c_2 \pmod{x_0}$.
- Producto: $c_{\text{prod}} = c_1 \cdot c_2 \pmod{x_0}$.

Ambos resultados son de nuevo elementos del espacio de los cifrados, y pueden ser descifrados de la misma manera que un cifrado original, siempre que el ruido acumulado se mantenga bajo control. Estas operaciones permiten la evaluación de cualquier circuito aritmético o booleano sobre bits cifrados, pero solo si la profundidad del circuito (y, por tanto, el crecimiento del ruido) es limitada.

**Lema 6.2.** La suma y el producto de dos cifrados corresponden, tras el descifrado, a la suma y producto de los bits originales, mientras el ruido total esté acotado.

*Demostración.* Supongamos que $c_1$ y $c_2$ son cifrados de $m_1, m_2 \in \{0,1\}$, construidos así:

$$c_1 = m_1 + 2r_1 + 2\sum_{i \in S_1} x_i \pmod{x_0}$$

$$c_2 = m_2 + 2r_2 + 2\sum_{j \in S_2} x_j \pmod{x_0}$$

donde $S_1, S_2 \subset \{1, \ldots, \tau\}$ son subconjuntos aleatorios y $r_1, r_2$ son ruidos independientes.

**Suma homomórfica:**

$$c_{\text{suma}} = c_1 + c_2 \pmod{x_0}$$

Sustituyendo,

$$\begin{aligned} c_{\text{suma}} &= \left( m_1 + m_2 + 2(r_1 + r_2) + 2\sum_{i \in S_1} x_i + 2\sum_{j \in S_2} x_j \right) \pmod{x_0} \\ &= (m_1 + m_2) + 2(r_1 + r_2) + 2\sum_{k \in S_1 \cup S_2} x_k \pmod{x_0} \end{aligned}$$

Al reducir $c_{\text{suma}}$ módulo $p$:

$$\begin{aligned} c_{\text{suma}} \bmod p &= (m_1 + m_2) + 2(r_1 + r_2) + 2\sum_{k \in S_1 \cup S_2} (q_k p + 2r_k) \bmod p \\ &= (m_1 + m_2) + 2(r_1 + r_2) + 2\sum_{k \in S_1 \cup S_2} q_k p + 4\sum_{k \in S_1 \cup S_2} r_k \bmod p \\ &= (m_1 + m_2) + 2(r_1 + r_2) + 4\sum_{k \in S_1 \cup S_2} r_k \bmod p \end{aligned}$$

Finalmente, al tomar el bit menos significativo:

$$m' = (c_{\text{suma}} \bmod p) \bmod 2 = (m_1 + m_2) \bmod 2$$

**Producto homomórfico:**

$$c_{\text{prod}} = c_1 \cdot c_2 \pmod{x_0}$$

Expandimos:

$$c_{\text{prod}} = (m_1 + 2r_1 + 2\sum_{i \in S_1} x_i) \cdot (m_2 + 2r_2 + 2\sum_{j \in S_2} x_j) \pmod{x_0}$$

Al multiplicar y desarrollar, el término principal es:

$$c_{\text{prod}} = m_1 m_2 + \text{términos pares} \pmod{x_0}$$

Reducimos módulo $p$, los términos con $x_k$ pasan a ser $2r_k$ y los múltiplos de $p$ desaparecen, obteniéndose:

$$c_{\text{prod}} \bmod p = m_1 m_2 + \text{ruido total (par)} \bmod p$$

Al tomar el bit menos significativo, se obtiene el producto de los mensajes:

$$m'' = (c_{\text{prod}} \bmod p) \bmod 2 = (m_1 m_2) \bmod 2$$

siempre que el ruido total esté acotado y no provoque desbordamiento modular.

□

Sin embargo, la acumulación de ruido tras varias operaciones limita la profundidad de circuitos evaluables. Se usan técnicas de reducción de ruido, como el *bootstrapping*, las cuales permiten superar esta barrera, como se verá en el siguiente apartado.

Esta construcción muestra cómo la aritmética modular y la combinación de ruido y suma de múltiplos de la clave secreta para "ocultar" el mensaje hacen posible un esquema de cifrado completamente homomórfico sobre los enteros, sencillo en apariencia pero con una base matemática profunda.

## 6.3. Limitaciones del esquema

Aunque el esquema DGHV logra implementar la propiedad homomórfica sobre los enteros, a la hora de llevar a cabo la aplicación práctica nos encontramos obstáculos importantes, principalmente relacionados con el crecimiento del ruido y la gestión de los parámetros.

### 6.3.1. Crecimiento del ruido y profundidad de circuitos

La acumulación de ruido es la limitación matemática más importante. Sea $c$ un cifrado, cuyo ruido total denotamos por $e$. Tras cada suma, el ruido crece de manera aditiva, y tras cada producto, puede crecer de forma multiplicativa.

**Lema 6.3.** Sea $c_1, c_2$ cifrados con ruidos $e_1, e_2$ respectivamente.

- La suma $c_1 + c_2$ tiene ruido $e_{\text{suma}} = e_1 + e_2$.
- El producto $c_1 \cdot c_2$ tiene ruido

$$e_{\text{prod}} = m_1 e_2 + m_2 e_1 + e_1 e_2$$

donde $m_1, m_2$ son los mensajes originales (bits).

*Demostración.* La suma es inmediata, pues los ruidos simplemente se suman. En el producto, expandiendo y considerando sólo los términos que no desaparecen al reducir módulo $p$, el ruido final incluye productos cruzados entre los ruidos y los mensajes. Si $m_1, m_2 \in \{0, 1\}$, los coeficientes son controlados pero el término $e_1 e_2$ puede crecer rápidamente. □

Como consecuencia, tras $d$ operaciones (especialmente multiplicaciones), el ruido puede alcanzar una magnitud que supere el umbral seguro para el descifrado, es decir, el límite $|e| < p/2$. Esta es la razón por la que DGHV, en su versión básica, solo permite evaluar circuitos de profundidad limitada.

### 6.3.2. Restricciones en la elección de parámetros

El tamaño del primo secreto $p$ debe ser suficiente para garantizar que, incluso tras llevar a cabo el máximo número permitido de operaciones, el ruido total siga cumpliendo $|e| < p/2$. Esto crea una relación directa entre los parámetros de seguridad y la funcionalidad del esquema. Por ejemplo, si se desean circuitos más profundos, $p$ debe ser mayor, lo que incrementa el tamaño de las claves y el coste computacional.

Además, el número de elementos públicos $\tau$ y el tamaño de los ruidos iniciales $\rho$ y $\rho'$ deben elegirse cuidadosamente para mantener tanto la seguridad como la corrección del cifrado. No existe una fórmul única, pero todas las propuestas prácticas requieren equilibrio entre profundidad máxima y eficiencia.

### 6.3.3. Eficiencia y escalabilidad

Desde un punto de vista algorítmico, la longitud de los cifrados y el tamaño de la clave pública crecen rápidamente con los parámetros de seguridad. Por ejemplo, cada $x_i$ y cada $c$ puede tener miles de bits. La complejidad de las operaciones aritméticas sobre números tan grandes limita el uso práctico del esquema, especialmente en escenarios que requieren un gran número de operaciones homomórficas.

En resumen, la matemática del crecimiento del ruido y la relación entre parámetros impone un límite natural a la potencia y eficiencia del esquema DGHV. Para superar estas barreras y obtener un FHE genuino, es necesario recurrir a técnicas de reducción de ruido, como el *bootstrapping*, que se presenta a continuación.

## 6.4. Bootstrapping y refresco de cifrados

La técnica del *bootstrapping*, introducida por Craig Gentry, es la clave para transformar un esquema SHE en uno FHE. En el contexto de DGHV, permite eliminar el principal obstáculo matemático: el crecimiento ilimitado del ruido tras operaciones sucesivas.

### 6.4.1. Profundidad de circuitos y capacidad de autoevaluación

La propiedad fundamental para que el bootstrapping funcione es que el propio esquema sea capaz de evaluar, de manera homomórfica, un circuito suficientemente complejo: el que implementa el algoritmo de descifrado.

**Definición 6.2** (Circuito de descifrado)**.** Sea $C_{\mathrm{Dec}}$ el circuito booleano/arimético que, dados como entrada un cifrado $c$ y la clave secreta $p$, computa $m = (c \bmod p) \bmod 2$.

El tamaño y la profundidad de $C_{\mathrm{Dec}}$ dependen de la representación binaria de $p$ y de la operación de módulo, que es computacionalmente costosa si $p$ es grande. Por tanto, la profundidad $d_{\mathrm{Dec}}$ de este circuito es crítica: si el esquema puede evaluar circuitos de al menos esa profundidad antes de que el ruido supere el umbral de descifrado, entonces es posible refrescar cualquier cifrado.

**Definición 6.3** (Esquema bootstrappable)**.** Un esquema SHE se dice *bootstrappable* si puede evaluar, de manera homomórfica, su propio circuito de descifrado $C_{\mathrm{Dec}}$.

### 6.4.2. Bootstrapping: procedimiento formal

Supongamos que el usuario dispone de la clave secreta $p$ y que la clave pública permite cifrar cualquier bit. Además, consideremos que existe una función de evaluación homomórfica $\mathsf{Eval}$.

El proceso de refresco se puede describir así:

1. Se cifra la clave secreta $p$ bit a bit, obteniendo un cifrado por cada bit, y todos ellos conforman $\mathsf{Enc}(p)$.

2. Se usa la función de evaluación homomórfica para calcular

$$\mathsf{Eval}(\mathrm{evk}, C_{\mathrm{Dec}}, (c, \mathsf{Enc}(p)))$$

   obteniendo como resultado un cifrado del bit $m = \mathrm{Dec}_p(c)$, pero con el ruido "reiniciado" al nivel de un cifrado fresco. De esta forma no se revela en ningún momento el bit $m$, ya que como estamos evaluando $C_{\mathrm{Dec}}$ de forma homomórfica el resultado sigue estando cifrado.

El bootstrapping permite reducir el ruido de cualquier cifrado al nivel inicial, tras un número arbitrario de operaciones, siempre que se refresque antes de superar el umbral seguro. Así, se pueden "encadenar" bloques de operaciones y refrescos sucesivos, y por inducción, evaluar circuitos de cualquier tamaño.

### 6.4.3. Squashing del circuito de descifrado en DGHV

Como se ha visto, el principal obstáculo técnico para el bootstrapping es la profundidad del circuito de descifrado $C_{\text{Dec}}$, cuya operación central es el cálculo del residuo $c$ mód $p$. Para claves $p$ suficientemente grandes, implementar este circuito en forma homomórfica requiere una profundidad superior a la que el esquema básico puede soportar. Para superar esta barrera, se introduce la técnica conocida como *squashing*, cuyo objetivo es reducir la profundidad del circuito de descifrado.

El esquema DGHV aplica esta técnica ampliando la clave pública con un vector $\vec{y} = (y_1, ..., y_\Theta)$. Estos $y_i$ son fracciones racionales en el intervalo $[0, 2)$ (con un cierto número de bits de precisión), tales que existe un subconjunto $S \subset \{1, \ldots, \Theta\}$ (con un cierto tamaño) que cumple

$$\sum_{i \in S} y_i \approx \frac{1}{p} \pmod 2.$$

Así, en lugar de calcular $c$ mód $p$ directamente, el nuevo circuito de descifrado tiene la siguiente estructura:

1. Se publica en la clave pública el vector de racionales $\vec{y}$.

2. Para un cifrado $c$, se calculan los bits $b_i$, que son los bits menos significativos de los productos $c \cdot y_i \pmod 2$, para cada $i = 1, ..., \Theta$.

3. El descifrado consiste en calcular el bit menos significativo de

$$c - \sum_{i=1}^{\Theta} b_i y_i$$

Este procedimiento permite reemplazar el cálculo de $c$ mód $p$ por una combinación de sumas, multiplicaciones y extracciones de bits menos significativos, todas ellas operaciones de baja profundidad de circuito. Para consultar todos los detalles sobre la construcción de los $y_i$ y el procedimiento, véase el artículo "Fully Homomorphic Encryption over the Integers", por van Dijk, Gentry, Halevi y Vaikuntanathan.

**Lema 6.4** (Reducción de profundidad mediante squashing)**.** Con el vector de racionales $\vec{y}$ incluido en la clave pública, el circuito de descifrado puede ser implementado con profundidad polinómica en el tamaño de la clave, permitiendo así su evaluación homomórfica antes de que el ruido alcance el límite crítico.

*Demostración.* La suma y la extracción de bits menos significativos pueden implementarse como circuitos booleanos poco profundos. El diseño concreto de $\vec{y}$ asegura que el cálculo es correcto mientras el ruido se mantenga dentro de los parámetros establecidos. □

**Nota**: La publicación de $\vec{y}$ implica que la seguridad del esquema no depende solo del problema del máximo común divisor aproximado, sino también de la dificultad del *Sparse Subset Sum Problem* (SSSP).

En resumen, la técnica de *squashing* en DGHV, tal como se describe en el artículo original, permite reemplazar la reducción modular por un circuito de baja profundidad basado en operaciones sobre el vector de racionales públicos $\vec{y}$ y extracciones de bits menos significativos. De esta forma, se reduce la profundidad del circuito de descifrado, haciendo factible el bootstrapping y, por tanto, el paso de SHE a FHE, pero introduce nuevos desafíos en cuanto a la gestión de la clave pública y la seguridad matemática del sistema

### 6.4.4. Conclusión del bootstrapping

En términos matemáticos, el bootstrapping constituye el mecanismo que permite superar la barrera del ruido acumulado siempre que se pueda evaluar $C_{\text{Dec}}$ de forma homomórfica:

$$\text{profundidad máxima de evaluación} \geq \text{profundidad de } C_{\text{Dec}}$$

Si esta condición se cumple, el esquema DGHV se convierte en FHE: es posible evaluar cualquier circuito computable sobre datos cifrados, lo que supone un salto conceptual que resolvió un problema abierto durante décadas y sentó las bases para toda la investigación posterior en criptografía homomórfica.

Sin embargo, el coste computacional del bootstrapping es muy elevado; en la práctica, representa el principal cuello de botella de los sistemas FHE sobre los enteros. Por ello, muchos de los desarrollos recientes en FHE, incluidos los esquemas basados en retículos o el cifrado aproximado, derivan de la idea original del bootstrapping.

Actualmente, el bootstrapping sigue siendo objeto de investigación matemática y computacional, y constituye un puente entre la teoría algebraica y la posibilidad de aplicaciones prácticas en computación segura, privacidad en la nube y criptografía post-cuántica.

## 6.5. Ejemplo práctico

Para ilustrar el funcionamiento del esquema DGHV, consideramos un ejemplo numérico sencillo, adaptado de los ejemplos presentados en los artículos de referencia.

### Parámetros

Elegimos parámetros pequeños para simplificar los cálculos (no son seguros en la práctica):

- Clave secreta: $p = 17$ (primo y pequeño; en la práctica debe ser mucho mayor).

- Parámetro de ruido: $\rho = 2$, de modo que los $r_i$ pueden ser $-3 \leq r_i \leq 3$ (con $|r_i| < 2^\rho$).
- Generamos tres valores públicos:

$$x_0 = q_0 p + 2r_0 = 11 \cdot 17 + 2 \cdot 2 = 187 + 4 = 191$$
$$x_1 = q_1 p + 2r_1 = 8 \cdot 17 + 2 \cdot (-1) = 136 - 2 = 134$$
$$x_2 = q_2 p + 2r_2 = 6 \cdot 17 + 2 \cdot 1 = 102 + 2 = 104$$

  Así, la clave pública es $(x_0, x_1, x_2)$.

## Cifrado y descifrado

Supongamos que queremos cifrar el bit $m = 1$.

- Elegimos aleatoriamente un subconjunto $S \subset \{1, 2\}$; por ejemplo, $S = \{1\}$.
- Elegimos un ruido $r = -1$.
- Calculamos el cifrado:

$$c = m + 2r + 2\sum_{i \in S} x_i \pmod{x_0} = 1 + 2 \cdot (-1) + 2 \cdot 134 \pmod{191} = ... = 76$$

  Así, el cifrado es $c = 76$.

Para descifrar, calculamos:

$$c \text{ mód } p = 76 \text{ mód } 17 = 76 - 4 \cdot 17 = 76 - 68 = 8$$

y el bit menos significativo:

$$8 \text{ mód } 2 = 0$$

En este caso, el descifrado ha fallado. Analicemos el motivo: el ruido total (en valor absoluto) debe ser menor que $p/2 = 8{,}5$. Volvamos a calcular el ruido:

$$e = 2r + 4\sum_{i \in S} r_i = 2 \cdot (-1) + 4 \cdot (-1) = -2 - 4 = -6$$

A pesar de que el ruido está dentro del rango permitido, la suma final no conserva el mensaje. Esto sucede a veces al usar parámetros pequeños como en nuestro caso. Con parámetros reales, la probabilidad de que el bit menos significativo no coincida con el mensaje original cuando $|e| < p/2$ es prácticamente nula.

Volvamos a intentarlo cambiando los parámetros ligeramente:

- Elegimos el subconjunto $S = \{2\}$ (es decir, usamos $x_2$).
- Elegimos un ruido $r = 1$.

- Calculamos el cifrado:

$$c = m + 2r + 2x_2 \pmod{x_0} = 1 + 2 \cdot 1 + 2 \cdot 104 \pmod{191} = ... = 20$$

Así, el cifrado es $c = 20$.

Para descifrar $c$:

$$c \bmod p = 20 \bmod 17 = 20 - 17 = 3$$

y el bit menos significativo:

$$3 \bmod 2 = 1$$

En este caso el descifrado es correcto. Ahora comprobamos el ruido:

$$e = 2r + 4\sum_{i \in S} r_i = 2 \cdot 1 + 4 \cdot 1 = 2 + 4 = 6$$

$$m + e = 1 + 6 = 7 < p/2 = 8,5$$

por lo que la condición de corrección se cumple.

## Suma homomórfica

Supongamos que ciframos dos bits $m_1 = 1, m_2 = 1$, obteniendo $c_1 = 20$ y $c_2 = 18$. Calculamos la suma homomórfica:

$$c_{\text{suma}} = c_1 + c_2 \pmod{191} = 20 + 18 = 38$$

Desciframos ahora el resultado:

$$38 \bmod 17 = 38 - 2 \cdot 17 = 38 - 34 = 4$$

$$4 \bmod 2 = 0$$

El bit resultante es 0, que corresponde a $m_1 + m_2 = 0$ (módulo 2).

## Producto homomórfico

Calculamos el producto homomórfico:

$$c_{\text{prod}} = c_1 \cdot c_2 \pmod{191} = 20 \cdot 18 = 360 \pmod{191} = 169$$

Ahora desciframos:

$$169 \bmod 17 = 169 - 9 \cdot 17 = 169 - 153 = 16$$

$$16 \bmod 2 = 0$$

Sin embargo, $m_1 \cdot m_2 = 1$. De nuevo, al igual que en el ejemplo anterior erróneo, esto sucede con mucha frecuencia al usar pequeños ejemplos.

## 6.6. Resumen y perspectiva: comparación y motivación hacia retículos

El esquema DGHV, como hemos visto, constituye avance clave en la historia de la criptografía homomórfica: es el primer esquema práctico que permite operar sobre datos cifrados de forma completamente general. Frente a sistemas como Paillier o Goldwasser-Micali, que solo admiten sumas o productos, DGHV logra la homomorfía total gracias a la gestión modular del ruido y el bootstrapping.

Sin embargo, DGHV presenta limitaciones importantes: los tamaños de clave y de cifrado crecen rápidamente, y el procedimiento de bootstrapping —aunque matemáticamente brillante— introduce un coste computacional muy elevado. Además, el tamaño de los parámetros necesarios para una seguridad realista dificulta su uso práctico. Todo eso hace que DGHV no sea viable en aplicaciones reales donde la eficiencia es crítica.

Por eso, la comunidad ha buscado alternativas más eficientes. Los esquemas FHE basados en retículos, como los construidos sobre LWE o RLWE, ofrecen mejoras claras: claves y cifrados más compactos, operaciones más rápidas, y fundamentos de seguridad resistentes incluso frente a futuros ataques cuánticos.

En resumen, DGHV es un punto de partida fundamental, pero la evolución hacia FHE basado en retículos es un paso natural tanto por motivos matemáticos como prácticos. El próximo capítulo estará dedicado precisamente a entender estos nuevos esquemas, sus ventajas y las ideas matemáticas que los sustentan.

# Capítulo 7

# FHE sobre Retículos: LWE y RLWE

## 7.1. Introducción

El desarrollo reciente de la criptografía completamente homomórfica ha puesto el foco en problemas de retículos como fundamento matemático para la construcción de esquemas eficientes y seguros. Tras el éxito del esquema DGHV basado en la aritmética de los enteros, se observó que la eficiencia y escalabilidad de estos sistemas estaban limitadas (crecimiento del ruido, complejidad de las operaciones...). Como respuesta, surgió una nueva generación de esquemas FHE que explotan la estructura algebraica de los retículos, en particular los problemas conocidos como *Learning With Errors* (LWE) y su versión en anillos, *Ring Learning With Errors* (RLWE).

El esquema basado en LWE se apoya en la dificultad computacional de problemas como el *Shortest Vector Problem* (SVP) en retículos. Este planteamiento proporciona garantías sólidas incluso frente a adversarios con capacidades cuánticas.

Por otro lado, la transición de LWE a RLWE corresponde a razones de eficiencia y aprovechamiento de estructuras algebraicas como los anillos de polinomios módulo ideales ciclotómicos. Esta adaptación hace posible optimizar el rendimiento y reducir el tamaño de las claves y los cifrados sin debilitar la seguridad del esquema.

En este capítulo abordamos los fundamentos matemáticos de los esquemas FHE basados en LWE y RLWE. Se expondrán las definiciones formales de los problemas subyacentes, la relación entre LWE y los problemas clásicos de retículos, y la construcción de esquemas FHE sobre estos fundamentos.

## 7.2. LWE y relación con los problemas de retículos

Los esquemas criptográficos basados en retículos se apoyan en la dificultad de ciertos problemas clásicos, especialmente el *Shortest Vector Problem* (SVP), y en la formulación del problema *Learning With Errors* (LWE), que constituye la piedra angular de la criptografía post-cuántica moderna.

### 7.2.1. Definición del problema LWE

El problema *Learning With Errors* (LWE), introducido por Regev en el año 2005 [11], puede describirse de la siguiente manera: Sean $n \in \mathbb{N}$, $q$ un entero positivo y $\chi$ una distribución de probabilidad sobre $\mathbb{Z}_q$. Se elige un vector secreto $\mathbf{s} \in \mathbb{Z}_q^n$, y se obtienen muestras de la forma

$$(\mathbf{a},\ b = \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q,$$

donde $\mathbf{a} \in \mathbb{Z}_q^n$ es aleatorio y $e \in \mathbb{Z}_q$ se toma según la distribución $\chi$. El objetivo es recuperar el secreto $\mathbf{s}$ (problema de búsqueda), o bien distinguir estas muestras de pares $(\mathbf{a}, b)$ elegidos uniformemente al azar en $\mathbb{Z}_q^n \times \mathbb{Z}_q$ (problema de decisión).

**Definición 7.1.** Dado $n, q, \chi$ y muestras $(\mathbf{a}_i, b_i)$, donde $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i$, con $\mathbf{a}_i \in \mathbb{Z}_q^n$ uniformes y $e_i \sim \chi$, el **problema de decisión LWE** consiste en distinguir entre muestras generadas mediante este proceso y muestras completamente aleatorias en $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

La seguridad de los esquemas criptográficos basados en LWE se basa en que, para ciertos parámetros (tamaño adecuado del error, dimensión y módulo), los cuales además permiten un esquema funcional, no existen algoritmos eficientes (clásicos ni cuánticos) conocidos capaces de resolver el problema en el caso general.

### 7.2.2. Relación con problemas clásicos de retículos

La relevancia de LWE para la criptografía se debe a que su dificultad está relacionada con problemas clásicos de retículos como el *Shortest Vector Problem* (SVP) y el *Shortest Independent Vector Problem* (SIVP). Regev demostró que resolver LWE para ciertos parámetros permite resolver SVP y SIVP en el peor caso sobre retículos de alta dimensión, lo que garantiza la seguridad teórica de los sistemas basados en LWE.

Formalmente, existen reducciones polinómicas que conectan la dificultad de problemas como SVP y SIVP con la de LWE: esto implica que la seguridad de los esquemas de cifrado basados en LWE puede basarse en la dureza de estos problemas clásicos bien estudiados en teoría de retículos.

#### Importancia de la base

En la práctica, la elección de la base del retículo es crucial: existen bases "mal condicionadas" (con vectores muy largos o muy próximos entre sí) y bases "reducidas", que facilitan la resolución de problemas sobre el retículo. La eficiencia de los algoritmos utilizados depende fuertemente de la base elegida, por lo que la dificultad de reducir una base "mala" a una base "buena" es precisamente lo que da seguridad a muchos esquemas criptográficos.

## 7.3. RLWE: extensión y motivación

### 7.3.1. Motivación para RLWE

El problema *Ring Learning With Errors* (RLWE) generaliza LWE al contexto de anillos de polinomios y permite representar y operar con bloques de datos de forma eficiente. Esta extensión responde tanto a necesidades de eficiencia computacional como al aprovechamiento de la estructura algebraica de los anillos, que permite reducir el tamaño de claves y cifrados sin debilitar la seguridad. Además, la aritmética en anillos facilita implementaciones rápidas mediante técnicas como la transformada de Fourier sobre anillos finitos (NTT). El artículo fundamental para consultar todos los detalles del RLWE es [12].

### 7.3.2. Definición formal del problema RLWE

Sea $R = \mathbb{Z}_q[x]/\langle f(x)\rangle$, donde $f(x)$ es un polinomio irreducible, normalmente ciclotómico. El secreto, el error y las muestras de RLWE son elementos de $R$. Una muestra es un par $(a(x), b(x)) \in R \times R$ con

$$b(x) = a(x)s(x) + e(x)$$

donde $a(x)$ es aleatorio, $s(x)$ es el secreto, y $e(x)$ es un error pequeño según una distribución discreta sobre $R$.

**Definición 7.2.** El **problema RLWE** consiste en distinguir entre muestras $(a(x), b(x))$, con $b(x) = a(x)s(x) + e(x)$ y $e(x)$ pequeño, y pares completamente aleatorios en $R \times R$.

La seguridad de RLWE se fundamenta en la dificultad de problemas de retículos ideales, y existen reducciones formales que justifican su solidez teórica.

El problema RLWE puede verse como una generalización natural de LWE: si en RLWE se toma un polinomio de grado uno, se recupera el esquema original de LWE. Sin embargo, trabajar sobre el anillo $R$ permite representar vectores largos de manera compacta, aplicar operaciones algebraicas más ricas y reducir notablemente el tamaño de las claves y los textos cifrados, manteniendo la dificultad computacional del problema subyacente. Por este motivo, los esquemas FHE actuales se construyen casi exclusivamente sobre RLWE, aprovechando tanto su eficiencia como la solidez de su base matemática.

## 7.4. Construcción de esquemas FHE basados en LWE y RLWE

La construcción de esquemas de cifrado homomórfico sobre LWE y RLWE sigue una estructura común en lo que respecta a la generación de claves, el cifrado y descifrado, y la definición de las operaciones homomórficas. La seguridad de estos esquemas se apoya en la dificultad computacional de los problemas LWE y RLWE, como se ha expuesto anteriormente.

### 7.4.1. Esquema básico de cifrado sobre LWE

Sea $q$ un módulo, $n$ la dimensión y $\chi$ la distribución del error. El esquema consta de los siguientes algoritmos:

- **KeyGen**: Elige $\mathbf{s} \in \mathbb{Z}_q^n$ uniformemente al azar como clave secreta.
- **Enc**: Para un mensaje $m \in \mathbb{Z}_q$, elige $\mathbf{a} \in \mathbb{Z}_q^n$ uniformemente y $e \sim \chi$, y calcula el cifrado como

$$c = (\mathbf{a},\ b = \langle \mathbf{a}, \mathbf{s} \rangle + e + m) \in \mathbb{Z}_q^n \times \mathbb{Z}_q.$$

- **Dec**: Dado un cifrado $c = (\mathbf{a}, b)$ y la clave secreta $\mathbf{s}$, se recupera el mensaje calculando

$$m = [b - \langle \mathbf{a}, \mathbf{s} \rangle]_q,$$

suponiendo que el error $e$ es suficientemente pequeño.

### 7.4.2. Esquema de cifrado sobre RLWE

En el caso de RLWE, los elementos viven en el anillo $R = \mathbb{Z}_q[x]/\langle f(x) \rangle$. El proceso es análogo:

- **KeyGen**: Elige $s(x) \in R$ uniformemente al azar como clave secreta.
- **Enc**: Para $m(x) \in R$, toma $a(x) \in R$ aleatorio y $e(x)$ pequeño, y calcula

$$c = (a(x),\ b(x) = a(x)s(x) + e(x) + m(x)) \in R \times R.$$

- **Dec**: Dados $c = (a(x), b(x))$ y la clave secreta $s(x)$, el mensaje se recupera como

$$m(x) = [b(x) - a(x)s(x)]_q,$$

siempre que el error $e(x)$ sea suficientemente pequeño.

En ambos casos, el crecimiento del ruido tras cada operación limita la cantidad de operaciones posibles antes de que la decodificación falle, motivando el desarrollo de técnicas de reducción de ruido (bootstrapping), que se tratarán en la siguiente sección.

### 7.4.3. Operaciones homomórficas en esquemas LWE y RLWE

Ambos esquemas permiten operaciones homomórficas sobre los textos cifrados. Sea $c_1 = (\mathbf{a}_1, b_1)$ y $c_2 = (\mathbf{a}_2, b_2)$ dos cifrados bajo LWE de los mensajes $m_1, m_2 \in \mathbb{Z}_q$:

- **Suma:**

$$c_1 + c_2 = (\mathbf{a}_1 + \mathbf{a}_2,\ b_1 + b_2)$$

El resultado es un cifrado de $m_1 + m_2$ con ruido igual a la suma de los ruidos originales.

- **Producto escalar:** Dado $\alpha \in \mathbb{Z}_q$,

$$\alpha \cdot c_1 = (\alpha \mathbf{a}_1,\ \alpha b_1)$$

que cifra el mensaje $\alpha m_1$.

El producto de dos cifrados LWE, sin modificaciones adicionales, no produce un cifrado válido bajo el mismo esquema (salvo en variantes más avanzadas).

En el caso de RLWE, sea $c_1 = (a_1(x), b_1(x))$ y $c_2 = (a_2(x), b_2(x))$ cifrados de $m_1(x), m_2(x) \in R$:

- **Suma:**

$$c_1 + c_2 = (a_1(x) + a_2(x),\ b_1(x) + b_2(x))$$

cifra $m_1(x) + m_2(x)$.

- **Multiplicación:**

$$c_1 * c_2 = (a_1(x)a_2(x),\ b_1(x)b_2(x))$$

El resultado cifra $m_1(x)m_2(x)$, aunque con un incremento significativo del ruido. Es necesario controlar este crecimiento para mantener la corrección del esquema.

### 7.4.4. Análisis del ruido

Como ocurre en todos los esquemas FHE, el principal obstáculo práctico es el crecimiento del ruido tras cada operación. En los esquemas basados en LWE y RLWE, al cifrar un mensaje, el error o ruido inicial, que proviene de la distribución $\chi$ elegida en el esquema, es pequeño y controlado. Sin embargo, cada operación homomórfica incrementa el nivel de ruido presente.

Consideremos el caso LWE. Si $c_1 = (\mathbf{a}_1, b_1)$ y $c_2 = (\mathbf{a}_2, b_2)$ son dos cifrados de $m_1$ y $m_2$ con errores $e_1$ y $e_2$ respectivamente, entonces:

- **Suma:** El nuevo cifrado $c = c_1 + c_2$ cifra $m_1 + m_2$ y su error es $e_{\text{sum}} = e_1 + e_2$. Después de $k$ sumas, el ruido total es, como máximo,

$$|e_{\text{sum}}| \leq k \cdot \text{máx}\{|e_1|, |e_2|, \ldots\}$$

Por lo que el crecimiento es lineal con el número de sumas.

- **Producto escalar:** Multiplicar un cifrado $c_1$ por un escalar $\alpha$ multiplica también el error: $|e_{\text{esc}}| = |\alpha||e_1|$.

- **Multiplicación (caso extendido):** Si el esquema permite multiplicación, el ruido del producto es más complejo: incluye términos como $m_2e_1 + m_1e_2 + e_1e_2$, junto con posibles nuevos errores introducidos por la estructura del esquema. De forma general, el ruido crece aproximadamente como

$$|e_{\text{prod}}| \leq |m_2|\,|e_1| + |m_1|\,|e_2| + |e_1e_2|$$

El crecimiento aquí es mucho más rápido, y se multiplica tras cada operación.

En esquemas RLWE, la situación es similar, pero el error es un polinomio en $R$ y se combina según la aritmética del anillo. Tras cada suma o multiplicación, el término dominante del ruido sigue patrones similares, aunque puede aprovecharse la estructura algebraica para optimizar su manejo.

En cualquier caso, para asegurar la corrección del esquema es necesario que el ruido total tras todas las operaciones homomórficas permanezca acotado. En el caso general del LWE, el ruido total tras todas las operaciones debe cumplir

$$|e_{\text{total}}| < \frac{q}{2},$$

y para el RLWE general, que la norma $\|e(x)\|_\infty < q/2$. Si esta cota se supera, se pierde la corrección del esquema (aunque las cotas pueden variar según las variantes del esquema).

Por tanto, el número de operaciones, especialmente multiplicaciones, que puede realizar un esquema FHE antes de necesitar un refresco del ruido (bootstrapping) está acotado.

La elección de los parámetros iniciales (dimensión $n$, módulo $q$ y distribución de error) determina el margen disponible para la acumulación de ruido y, por tanto, la funcionalidad práctica del sistema. El equilibrio entre seguridad y eficiencia depende críticamente de este análisis.

### 7.4.5. Comparación entre LWE y RLWE

El esquema RLWE puede verse como una extensión algebraica eficiente de LWE. Ambos permiten suma homomórfica y, con modificaciones, multiplicación. Sin embargo, RLWE representa los mensajes y operaciones en un anillo polinómico, lo que permite trabajar de forma más compacta y eficiente sobre bloques de datos y soportar multiplicación homomórfica de manera natural. Además, los esquemas basados en RLWE presentan mejor escalabilidad y un menor tamaño relativo de las claves y los textos cifrados. Por todo esto, la gran mayoría de las implementaciones actuales se llevan a cabo con esquemas RLWE.

Para detalles sobre la eficiencia y propiedades algebraicas de RLWE frente a LWE, véase [7], y [12].

## 7.5. Bootstrapping y FHE completo

Al igual que en otros esquemas de cifrado homomórfico, en aquellos basados en LWE y RLWE el principal obstáculo práctico es el crecimiento del ruido. Si el ruido acumulado supera cierto umbral, el descifrado ya no funciona correctamente. El bootstrapping es la técnica que permite superar este límite y lograr un cifrado completamente homomórfico.

### 7.5.1. Implementación del circuito de descifrado

El descifrado en estos esquemas es una función sencilla de los componentes del cifrado y de la clave secreta. Por ejemplo, en RLWE basta con calcular

$$m(x) = [b(x) - a(x)s(x)]_q,$$

donde $c = (a(x), b(x))$ es el cifrado, $s(x)$ la clave secreta y $[\cdot]_q$ indica la reducción módulo $q$. Sin embargo, cuando se quiere evaluar este descifrado homomórficamente, es necesario implementar cada una de sus operaciones como circuitos sobre textos cifrados. Esto implica:

- **Multiplicación homomórfica por el secreto:** la multiplicación por $s(x)$ debe realizarse usando una versión cifrada de la clave, por lo que la clave pública incluye una codificación de $s(x)$ preparada para este uso (key switching).
- **Reducción módulo $q$ y redondeo:** para obtener el mensaje es necesario aplicar operaciones de reducción módulo $q$ o bien redondear los coeficientes de los polinomios. Estas operaciones deben representarse como circuitos aritméticos de baja profundidad.
- **Optimización de la representación:** las variantes modernas emplean claves secretas pequeñas o binarias y técnicas algebraicas para reducir la complejidad y la profundidad del circuito de descifrado.

Además, la clave pública debe contener una codificación cifrada de la clave secreta para que la evaluación sea posible sobre textos cifrados. Finalmente, al hacer la evaluación se obtiene un nuevo cifrado del mismo mensaje pero con el ruido reinicializado:

$$c' = \mathsf{Eval}(\mathcal{C}_{\text{dec}}, c, \mathsf{Enc}(s)),$$

donde $c'$ es un nuevo cifrado de $m$ con ruido reducido, y $\mathsf{Enc}(s)$ es la encriptación de la clave secreta.

### 7.5.2. Profundidad de circuito y viabilidad

Para que el bootstrapping sea posible, el esquema debe permitir la evaluación homomórfica del circuito de descifrado antes de que el ruido supere el umbral de corrección. La profundidad de este circuito, denotada $D(\mathcal{C})$, depende del esquema concreto, pero gracias a la aritmética polinómica de RLWE suele ser relativamente baja.

En los esquemas iniciales como DGHV, el bootstrapping era lento y costoso. La estructura de RLWE ha cambiado este panorama: gracias a la aritmética en anillos y al uso de la transformada de Fourier sobre anillos (NTT), el coste computacional ha disminuido drásticamente. Además, los esquemas modernos emplean nuevas técnicas como el *modulus switching* y el *key switching* para controlar del crecimiento del ruido:

- El modulus switching consiste en reducir progresivamente el módulo del cifrado a medida que avanzan las operaciones, manteniendo el error bajo control.
- El key switching permite transformar un cifrado bajo una clave de evaluación intermedia en otro cifrado bajo la clave original, evitando que el tamaño de las claves y de los cifrados crezca de forma incontrolada.

Ambas ideas han resultado fundamentales para incrementar la eficiencia y la profundidad de los circuitos que pueden evaluarse sin necesidad de recurrir inmediatamente al bootstrapping [13].

Todo esto ha hecho posible el desarrollo de esquemas FHE prácticos, como BGV, BFV y CKKS, capaces de manejar operaciones sobre datos cifrados en tiempos razonables y permitiendo aplicaciones en cloud computing, privacidad en aprendizaje automático y análisis de datos sensibles.

## 7.6. Ejemplo práctico de RLWE

Consideramos el anillo $R = \mathbb{Z}_7[x]/\langle x^2+1\rangle$, con módulo $q = 7$. Usaremos mensajes constantes $m_1, m_2 \in \mathbb{Z}_7$ para simplificar la notación y facilitar los cálculos.

### Generación de claves

Elegimos una clave secreta sencilla, por ejemplo:

$$s(x) = 2$$

### Cifrado de dos mensajes

Supongamos que queremos cifrar $m_1 = 3$ y $m_2 = 4$.

Para el cifrado de $m_1 = 3$, primero elegimos $a_1(x) = 1$ y $e_1(x) = 1$. Después, calculamos

$$b_1(x) = a_1(x) \cdot s(x) + e_1(x) + m_1 = 1 \cdot 2 + 1 + 3 = 2 + 1 + 3 = 6$$

$$c_1 = (a_1(x), b_1(x)) = (1, 6)$$

Ahora, para el cifrado de $m_2 = 4$, elegimos $a_2(x) = 3$, $e_2(x) = 2$, y calculamos

$$b_2(x) = a_2(x) \cdot s(x) + e_2(x) + m_2 = 3 \cdot 2 + 2 + 4 = 6 + 2 + 4 = 12$$

Reducimos módulo 7:

$$12 \equiv 5 \pmod 7$$

$$c_2 = (a_2(x), b_2(x)) = (3, 5)$$

### Suma homomórfica

La suma homomórfica de los dos cifrados es

$$c_{\text{sum}} = (a_1(x) + a_2(x),\ b_1(x) + b_2(x)) = (1 + 3,\ 6 + 5) = (4, 11)$$

Reducimos módulo 7, y $11 \equiv 4 \pmod 7$, así que

$$c_{\text{sum}} = (4, 4)$$

### Descifrado de la suma

Calculamos:

$$b_{\text{sum}}(x) - a_{\text{sum}}(x) \cdot s(x) = 4 - 4 \cdot 2 = 4 - 8 = -4 \equiv 3 \pmod 7$$

El resultado esperado sería la suma $m_1 + m_2 = 3 + 4 = 7 \equiv 0 \pmod 7$. Sin embargo, el descifrado produce 3 en vez de 0. Esto se debe a que, en los esquemas RLWE, el descifrado recupera el mensaje sumado al error total acumulado:

$$\text{descifrado} = (m_1 + m_2) + (e_1 + e_2) = 0 + (1 + 2) = 3 \pmod 7$$

En nuestro caso no hay problema, ya que se cumple que

$$|e_1 + e_2| = 3 < q/2 = 7/2 = 3,5$$

Por lo tanto, podemos asegurar que el descifrado no ha fallado: simplemente el resultado ha sido desplazado por el error.

En general, mientras el error total $(e_1 + e_2)$ sea pequeño, el esquema es seguro y funcional: el mensaje puede recuperarse correctamente mediante redondeo o mapeo a la representación original. Además, en aplicaciones prácticas muchas veces se acepta que el mensaje recuperado pueda diferir por un error pequeño, ya que no afecta demasiado al resultado, y el esquema sigue siendo seguro.

En resumen, este ejemplo muestra que el descifrado recupera la suma de los mensajes desplazada por el error acumulado. Mientras este desplazamiento sea pequeño comparado con $q$, el esquema es funcional y seguro, y así se garantiza en aplicaciones reales mediante la elección adecuada de parámetros.

## 7.7. Resumen y perspectiva

### 7.7.1. Comparativa con DGHV y esquemas anteriores

El desarrollo del FHE comenzó con propuestas como el esquema DGHV, basado en la aritmética de los enteros. Aunque supuso un gran avance conceptual, sus limitaciones prácticas son claras: los textos cifrados y las claves son grandes, el crecimiento del ruido es muy rápido, y la técnica de bootstrapping resulta prohibitiva para la mayoría de las aplicaciones prácticas.

Actualmente, los esquemas basados en retículos, en particular LWE y RLWE, cambian este panorama. El uso de la estructura algebraica de los retículos y, especialmente, de los anillos de polinomios, permite reducir el tamaño de las claves y de los cifrados, y hace mucho más eficiente tanto el cifrado como la evaluación homomórfica. Las técnicas algebraicas modernas, como la transformada de Fourier (NTT), han hecho posible la aplicación real de FHE.

Pero, más allá de la eficiencia, la diferencia más profunda está en el fundamento matemático: la seguridad de LWE y RLWE se apoya en problemas clásicos de retículos (como SVP y SIVP), aportando una garantía matemática robusta incluso frente a adversarios cuánticos.

### 7.7.2. Ventajas y retos de los esquemas basados en retículos

Como principales ventajas de los esquemas basados en LWE y RLWE destacan:

- **Fundamento matemático sólido:** Su seguridad se apoya en la dificultad computacional de problemas clásicos de retículos, para los que no se conocen algoritmos eficientes ni siquiera en el modelo cuántico.
- **Eficiencia y escalabilidad:** La estructura de anillo y el uso de técnicas algebraicas permiten cifrados y claves de tamaño polinómico, y hacen viables implementaciones eficientes de operaciones homomórficas, incluidas suma y producto.
- **Viabilidad del bootstrapping:** La evaluación homomórfica del circuito de descifrado es más eficiente y menos profunda que en esquemas previos, permitiendo sistemas FHE utilizables en la práctica.
- **Criptografía post-cuántica:** LWE y RLWE constituyen la base de muchas de las propuestas actuales para la estandarización de criptografía resistente a ataques cuánticos.

Sin embargo, estos esquemas presentan retos abiertos:

- **Crecimiento del ruido:** Aunque mucho más controlado que en DGHV, el ruido sigue limitando la profundidad de los circuitos evaluables y la eficiencia práctica del bootstrapping.
- **Elección de parámetros:** El equilibrio entre seguridad, tamaño de clave, módulo $q$ y error sigue siendo un tema de investigación, especialmente ante posibles ataques avanzados.
- **Implementación segura:** La implementación eficiente y resistente a ataques de canal lateral sigue siendo un desafío.

En conclusión, los esquemas basados en retículos han cambiado la perspectiva de la criptografía homomórfica. El enfoque algebraico y las garantías matemáticas hacen de RLWE el paradigma dominante, y los avances técnicos recientes permiten vislumbrar aplicaciones prácticas que, hace solo una década, parecían fuera de alcance.

# Capítulo 8

# Conclusión y perspectivas futuras

## 8.1. Resumen de la evolución matemática del FHE

La criptografía completamente homomórfica (FHE) surge del reto matemático de diseñar sistemas de cifrado que permitan realizar operaciones aritméticas directamente sobre datos cifrados, de forma que el resultado, una vez descifrado, coincida exactamente con la operación realizada en claro. Formalmente, se exige que el esquema de cifrado sea homomórfico respecto a una estructura algebraica concreta, normalmente la suma y el producto en un anillo.

En sus inicios, los esquemas existentes solo permitían una única operación, ya fuera la suma (como en Goldwasser-Micali) o el producto (como en RSA). Esta limitación impulsó el desarrollo de los esquemas algo homomórficos (SHE), que permiten realizar ambas operaciones, pero únicamente un número limitado de veces. Esta restricción viene impuesta por el crecimiento del ruido en los textos cifrados: al superar cierto umbral, el descifrado deja de ser fiable y el esquema pierde funcionalidad.

El avance fundamental se produce con la propuesta de Craig Gentry [10], que introduce el concepto de *bootstrapping*: una técnica matemática que permite reducir el ruido acumulado mediante la evaluación homomórfica del algoritmo de descifrado del propio esquema. Esta idea revolucionaria inaugura la primera construcción plenamente homomórfica (FHE), capaz de soportar circuitos de profundidad arbitraria manteniendo la seguridad y la corrección.

La evolución posterior ha sido principalmente algebraica. Inicialmente, los esquemas se construían sobre los enteros utilizando operaciones modulares, como en DGHV [8], pero estos esquemas eran muy poco funcionales debido a su altísimo costo computacional. Sin embargo, los avances más significativos se han producido al reformular los sistemas sobre anillos de polinomios y módulos de alta dimensión, lo que ha permitido dejar atrás esquemas excesivamente costosos y abrir la puerta a aplicaciones prácticas. Esta nueva estructura algebraica vincula la seguridad del cifrado a problemas fundamentales de teoría de retículos, como *Learning With Errors* (LWE) y *Ring Learning With Errors* (RLWE). En consecuencia, la solidez de estos esquemas se basa en la dificultad computacional de problemas clásicos como el Shortest Vector Problem (SVP) y el Closest Vector Problem (CVP), considerados inabordables incluso para algoritmos cuánticos [13, 7, 1].

En resumen, la evolución matemática del FHE refleja una transición desde construcciones elementales basadas en operaciones sobre los enteros, hacia esquemas más actuales basados en la teoría de retículos, en los que la interacción entre la estructura algebraica y la seguridad computacional constituye el motor principal de los avances recientes.

## 8.2. Estado actual y variantes principales

En los últimos años, el campo de la criptografía homomórfica ha dado lugar a varios esquemas fundamentales, cada uno con características propias y pensados para distintas aplicaciones matemáticas y prácticas. Entre los más destacados se encuentran:

- **BGV** (*Brakerski-Gentry-Vaikuntanathan*) [13]: Este esquema está construido sobre la dificultad del problema *Ring Learning With Errors* (RLWE) y utiliza anillos de polinomios para estructurar los mensajes. Utiliza técnicas como el *modulus switching* y el *key switching*, que permiten controlar el crecimiento del ruido y facilitan la evaluación eficiente de circuitos. En la actualidad, BGV es una de las bases teóricas más estudiadas y aplicadas.

- **CKKS** (*Cheon-Kim-Kim-Song*) [14]: A diferencia de otros esquemas, CKKS permite operaciones homomórficas aproximadas sobre números reales y complejos. Su principal aportación es la posibilidad de cifrar y manipular directamente vectores de valores en punto flotante, lo que facilita aplicaciones en aprendizaje automático, análisis estadístico y otras áreas donde la precisión absoluta no es imprescindible. Al igual que BGV, la seguridad de CKKS se basa en la dificultad computacional de RLWE.

- **TFHE** (*Fast Fully Homomorphic Encryption over the Torus*) [15]: Este esquema está especialmente optimizado para la evaluación eficiente de circuitos lógicos y operaciones bit a bit. Su implementación del *bootstrapping* es especialmente rápida y permite refrescar el ruido después de cada operación lógica, posibilitando un procesamiento ágil sobre datos cifrados. TFHE ilustra cómo el diseño algebraico puede adaptarse a distintas necesidades funcionales dentro del marco general de la FHE.

Estos y otros esquemas, como B/FV o HEAAN, muestran una tendencia clara: aprovechar la estructura algebraica rica de los anillos de polinomios y vincular la seguridad a problemas difíciles de teoría de retículos, como LWE y RLWE. Gracias a estas ideas, la criptografía homomórfica está dejando de ser solo un interés teórico para acercarse cada vez más a aplicaciones reales en la práctica [7, 1, 9].

## 8.3. Ventajas y retos actuales

El desarrollo de la criptografía completamente homomórfica ha supuesto una revolución conceptual, tanto por las posibilidades matemáticas que ofrece como por los desafíos técnicos que aún presenta. A continuación se resumen los principales logros y dificultades que marcan la investigación en este campo:

**Ventajas matemáticas y criptográficas:**

- La seguridad de los esquemas modernos está fundamentada en problemas de teoría de retículos, como LWE y RLWE, cuya dificultad es resistente incluso frente a algoritmos cuánticos.

- La estructura algebraica de los anillos de polinomios permite definir operaciones homomórficas generales y construir cifrados compactos, con técnicas avanzadas como *modulus switching* y *key switching* para controlar el ruido.

- Existen esquemas versátiles (como CKKS o BGV) que se adaptan a operaciones aritméticas exactas o aproximadas, facilitando aplicaciones en distintos ámbitos.

**Retos y limitaciones:**

- *Crecimiento del ruido:* Aunque las técnicas modernas permiten gestionar el ruido generado, el bootstrapping sigue siendo costoso en términos computacionales. La eficiencia de los esquemas depende en gran medida de la gestión del ruido y de la elección de parámetros adecuados.

- *Coste computacional y tamaño de los cifrados:* A pesar de los avances, los requerimientos de tiempo y espacio son elevados en comparación con otros sistemas criptográficos. El tamaño de las claves y de los textos cifrados, así como el coste de las operaciones, son un obstáculo para aplicaciones a gran escala.

- *Ataques de implementación y elección de parámetros:* La seguridad práctica depende no solo de la dificultad matemática subyacente, sino también de la elección de los parámetros y de la resistencia a ataques laterales, que pueden aprovechar vulnerabilidades en implementaciones concretas.

- *Compatibilidad y estandarización:* La diversidad de esquemas y parámetros dificulta la interoperabilidad y la adopción generalizada. Actualmente se están desarrollando propuestas de estandarización y bibliotecas eficientes y seguras [9].

En conjunto, el campo avanza hacia una mayor eficiencia y robustez, aunque la frontera entre la teoría matemática y la viabilidad práctica sigue siendo el principal reto a resolver en los próximos años.

## 8.4. Aplicaciones prácticas y futuro

El desarrollo de la criptografía completamente homomórfica (FHE) ha abierto la puerta a aplicaciones que, hasta hace pocos años, solo podían plantearse de forma teórica. Gracias a la posibilidad de operar directamente sobre datos cifrados sin necesidad de descifrarlos, el FHE permite resolver problemas donde la confidencialidad de la información es esencial. Entre las aplicaciones más relevantes descritas en la literatura [9], se encuentran:

- **Privacidad del consumidor en publicidad personalizada:** El uso de FHE permite ofrecer recomendaciones o anuncios basados en preferencias del usuario, sin que el proveedor tenga acceso directo a datos personales como la ubicación o históricos de consumo.
- **Aplicaciones médicas:** Los datos clínicos de los pacientes pueden procesarse (por ejemplo, para análisis predictivo o diagnóstico) sin exponer en ningún momento información sensible, facilitando la colaboración entre centros médicos y el análisis de datos a gran escala bajo restricciones de privacidad. De hecho, ya existen modelos que son capaces de predecir un infarto de un paciente sin revelar sus datos personales.
- **Minería de datos y análisis estadístico:** Es posible realizar operaciones complejas sobre bases de datos cifradas, extrayendo patrones o generando estadísticas sin comprometer la confidencialidad de los datos originales.
- **Privacidad financiera:** FHE posibilita la externalización de cálculos financieros a la nube o a servicios externos, manteniendo la seguridad tanto de los datos financieros como de los algoritmos utilizados para analizarlos. Algunos ejemplos son el intercambio seguro de información entre bancos, evaluaciones de riesgo con datos cifrados, consultas privadas sobre transacciones...
- **Votación electrónica:** Los esquemas FHE permiten construir sistemas de voto en los que el recuento se realiza directamente sobre votos cifrados, preservando tanto la privacidad del votante como la integridad del proceso electoral.
- **Reconocimiento forense de imágenes:** Se pueden comparar huellas digitales, imágenes o archivos cifrados con bases de datos sensibles, sin que las partes tengan acceso al contenido real, muy útil en ámbitos policiales.
- **Herramientas criptográficas avanzadas:** Los esquemas FHE sirven como bloques de construcción para otros sistemas como firmas homomórficas, autenticadores de mensajes y computación multiparte segura.

Aunque el despliegue masivo de FHE todavía está condicionado por limitaciones computacionales y de eficiencia, los avances de la última década permiten vislumbrar aplicaciones prácticas en ámbitos como la computación en la nube, la sanidad, la banca, o incluso procesos electorales electrónicos. El desafío futuro reside en seguir perfeccionando las bases matemáticas y en trasladar estos avances a soluciones robustas y accesibles, donde la privacidad y la seguridad sigan garantizadas por la dificultad intrínseca de los problemas algebraicos subyacentes.

# Bibliografía


[1] Rebecca M. Meissen. A mathematical approach to fully homomorphic encryption. Technical report, Worcester Polytechnic Institute, 100 Institute Road, Worcester MA 01609-2280 USA, April 2012.

[2] J.W.S. Cassels. *An Introduction to the Geometry of Numbers*. Springer, 1971.

[3] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[4] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[5] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[6] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[7] Alice Silverberg. Fully homomorphic encryption for mathematicians. *Notices of the American Mathematical Society*, 60(7):844–852, 01 2013.

[8] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 24–43, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[9] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Reuter, and Martin Strand. A guide to fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2015:1192, 2015.

[10] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery.

[11] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), September 2009.

[12] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6), November 2013.

[13] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 309–325, New York, NY, USA, 2012. Association for Computing Machinery.

[14] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. Cryptology ePrint Archive, Paper 2016/421, 2016.

[15] Masahiro Yagisawa. Improved fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Paper 2017/763, 2017.